NATURAL EXPERIMENTS IN NLP AND WHERE TO FIND THEM

Pietro Lesci
University of Cambridge

X: @pietro_lesci
LinkedIn: /pietrolesci
Page: pietrolesci.com
Mail: pietrolesci@outlook.com

CAUSAL ESTIMATION OF MEMORISATION PROFILES

ACL 2024 (Best Paper Award 🏆)

Pietro Lesci

Clara Meister

Thomas Hofmann

Andreas Vlachos

Tiago Pimentel

Why study memorisation?

Why study memorisation?

Language models can reproduce entire sequences from their training set verbatim (Carlini, 2021; 2023).

Why study memorisation?

Language models can reproduce entire sequences from their training set verbatim (Carlini, 2021; 2023).

Memorisation has implications for:

- Copyright and data protection
- How models encode factual information
- Our understanding of models' training dynamics

Why study memorisation?

Language models can reproduce entire sequences from their training set verbatim (Carlini, 2021; 2023).

Memorisation has implications for:

- Copyright and data protection
- How models encode factual information
- Our understanding of models' training dynamics

In order to be able to study memorisation we need methods to quickly, cheaply, and accurately measure it!

Tl;dr: Estimate memorisation directly from the data!

Tl;dr: Estimate memorisation directly from the data!

We want to estimate **counterfactual memorisation**
(as defined in Feldman, 2020):

Tl;dr: Estimate memorisation directly from the data!

We want to estimate **counterfactual memorisation**
(as defined in Feldman, 2020):

"The causal effect of observing an instance during training on a model's ability to correctly predict that instance"

Tl;dr: Estimate memorisation directly from the data!

We want to estimate **counterfactual memorisation**
(as defined in Feldman, 2020):

"The causal effect of observing an instance during training on a model's ability to correctly predict that instance"

Tl;dr: Estimate memorisation directly from the data!

We want to estimate **counterfactual memorisation**
(as defined in Feldman, 2020):

"The causal effect of observing an instance during training on a model's ability to correctly predict that instance"

🤔 Problem: Current methods require re-training the model multiple times, which is infeasible for recent language models and datasets

Tl;dr: Estimate memorisation directly from the data!

We want to estimate **counterfactual memorisation**
(as defined in Feldman, 2020):

"The causal effect of observing an instance during training on a model's ability to correctly predict that instance"

🤔 Problem: Current methods require re-training the model multiple times, which is infeasible for recent language models and datasets

😃 Solution: Use econometrics to estimate causal effects directly from data (i.e., model evaluations)!

Tl;dr: Estimate memorisation directly from the data!

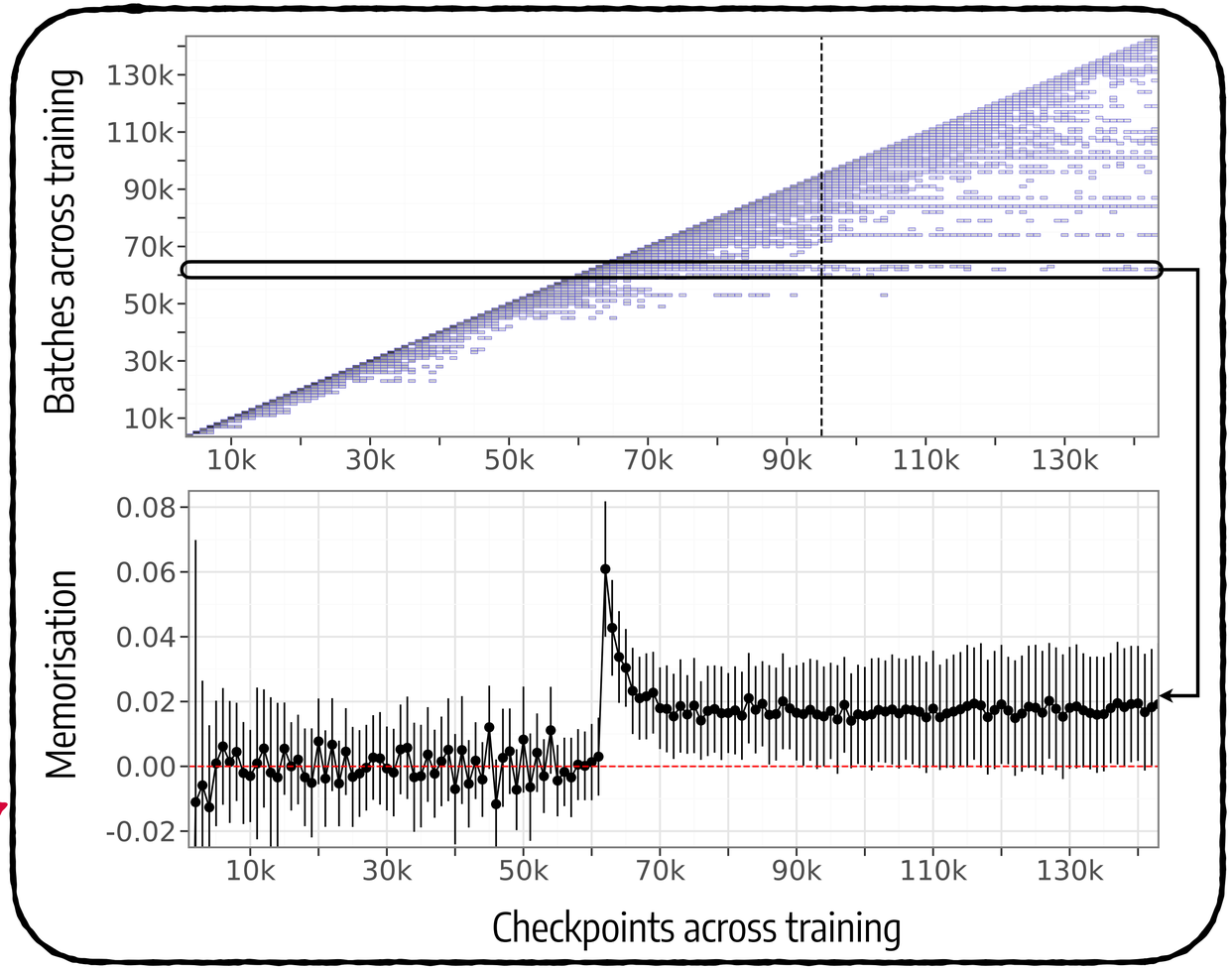
We want to estimate **counterfactual memorisation** (as defined in Feldman, 2020):

"The causal effect of observing an instance during training on a model's ability to correctly predict that instance"

🤔 Problem: Current methods require re-training the model multiple times, which is infeasible for recent language models and datasets

😀 Solution: Use econometrics to estimate causal effects directly from data (i.e., model evaluations)!

The output of our method: the Memorisation Profile



Memorisation as a difference of potential outcomes

Memorisation as a difference of potential outcomes

$$\tau_{\boldsymbol{x},c} \overset{\text{def}}{=} \underbrace{Y_c(\boldsymbol{x}; g)} - \underbrace{Y_c(\boldsymbol{x}; \infty)}$$

Memorisation as a difference of potential outcomes

$$\tau_{\boldsymbol{x},c} \overset{\text{def}}{=} \underbrace{Y_c(\boldsymbol{x}; g)}_{\downarrow} - \underbrace{Y_c(\boldsymbol{x}; \infty)}_{}$$

Memorisation as a difference of potential outcomes

$$\tau_{\boldsymbol{x},c} \overset{\text{def}}{=} \underbrace{Y_c(\boldsymbol{x}; g)}_{\downarrow} - \underbrace{Y_c(\boldsymbol{x}; \infty)}_{}$$

Memorisation as a difference of potential outcomes

$$\tau_{\boldsymbol{x},c} \overset{\text{def}}{=} \underbrace{Y_c(\boldsymbol{x}; g)}_{\text{Performance of model on } \boldsymbol{x} \text{ when trained with } \boldsymbol{x} \text{ at step } g} - \underbrace{Y_c(\boldsymbol{x}; \infty)}$$

Memorisation as a difference of potential outcomes

$$\tau_{\boldsymbol{x},c} \overset{\text{def}}{=} \underbrace{Y_c(\boldsymbol{x}; g)}_{\text{Performance of model on } \boldsymbol{x} \text{ when trained with } \boldsymbol{x} \text{ at step } g} - \underbrace{Y_c(\boldsymbol{x}; \infty)}_{\downarrow}$$

Memorisation as a difference of potential outcomes

$$\tau_{\boldsymbol{x},c} \overset{\text{def}}{=} \underbrace{Y_c(\boldsymbol{x}; g)}_{\text{Performance of model on } \boldsymbol{x} \text{ when trained with } \boldsymbol{x} \text{ at step } g} - \underbrace{Y_c(\boldsymbol{x}; \infty)}_{\text{Performance of model on } \boldsymbol{x} \text{ when } \textbf{not} \text{ trained with } \boldsymbol{x}}$$

Memorisation as a difference of potential outcomes

$$\tau_{\boldsymbol{x},c} \overset{\text{def}}{=} \underbrace{Y_c(\boldsymbol{x}; g)}_{\text{Performance of model on } \boldsymbol{x} \text{ when trained with } \boldsymbol{x} \text{ at step } g} - \underbrace{Y_c(\boldsymbol{x}; \infty)}_{\text{Performance of model on } \boldsymbol{x} \text{ when } \textbf{not} \text{ trained with } \boldsymbol{x}}$$

Memorisation as a difference of potential outcomes

$$\tau_{\boldsymbol{x},c} \overset{\text{def}}{=} \underbrace{Y_c(\boldsymbol{x}; g)}_{\text{Performance of model on } \boldsymbol{x} \text{ when trained with } \boldsymbol{x} \text{ at step } g} - \underbrace{Y_c(\boldsymbol{x}; \infty)}_{\text{Performance of model on } \boldsymbol{x} \text{ when } \textbf{not} \text{ trained with } \boldsymbol{x}}$$

For each training run, we observe only one of the two potential outcomes!

Estimating the counterfactual outcome $Y_c(x; \infty)$

Estimating the counterfactual outcome $Y_c(\boldsymbol{x}; \infty)$

(k,l)-extractable memorisation (Carlini et al., 2023):

Assume the counterfactual performance is negligible in the absence of training

Estimating the counterfactual outcome $Y_c(\boldsymbol{x}; \infty)$

(k,l)-extractable memorisation (Carlini et al., 2023):

Assume the counterfactual performance is negligible in the absence of training

$$\tau^{\texttt{extr}}_{\boldsymbol{x},c} = Y_c(\boldsymbol{x}; g)$$

Estimating the counterfactual outcome $Y_c(\boldsymbol{x}; \infty)$

(k,l)-extractable memorisation (Carlini et al., 2023):

Assume the counterfactual performance is negligible in the absence of training

$$\tau^{\texttt{extr}}_{\boldsymbol{x},c} = Y_c(\boldsymbol{x}; g)$$

- Strong assumption
- Does not account for a “baseline” predictability

Estimating the counterfactual outcome $Y_c(\boldsymbol{x}; \infty)$

(k,l)-extractable memorisation (Carlini et al., 2023):

Assume the counterfactual performance is negligible in the absence of training

$$\tau^{\texttt{extr}}_{\boldsymbol{x},c} = Y_c(\boldsymbol{x}; g)$$

- Strong assumption
- Does not account for a “baseline” predictability

Architectural memorisation (Feldman, 2020; and later work):

Re-train the model multiple times and average across runs

Estimating the counterfactual outcome $Y_c(\boldsymbol{x}; \infty)$

(k,l)-extractable memorisation (Carlini et al., 2023):

Assume the counterfactual performance is negligible in the absence of training

$$\tau^{\texttt{extr}}_{\boldsymbol{x},c} = Y_c(\boldsymbol{x}; g)$$

- Strong assumption
- Does not account for a “baseline” predictability

Architectural memorisation (Feldman, 2020; and later work):

Re-train the model multiple times and average across runs

$$\tau^{\texttt{arch}}_{\boldsymbol{x},p(\psi)} = \mathop{\mathbb{E}}_{\boldsymbol{\psi}}\left[Y_T(\boldsymbol{x}; G(\boldsymbol{x})) \mid G(\boldsymbol{x}) \neq \infty\right] - \mathop{\mathbb{E}}_{\boldsymbol{\psi}}\left[Y_T(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = \infty\right]$$

Estimating the counterfactual outcome $Y_c(\boldsymbol{x}; \infty)$

(k,l)-extractable memorisation (Carlini et al., 2023):

Assume the counterfactual performance is negligible in the absence of training

$$\tau^{\texttt{extr}}_{\boldsymbol{x},c} = Y_c(\boldsymbol{x}; g)$$

- Strong assumption
- Does not account for a "baseline" predictability

Architectural memorisation (Feldman, 2020; and later work):

Re-train the model multiple times and average across runs

$$\tau^{\texttt{arch}}_{\boldsymbol{x},p(\psi)} = \mathop{\mathbb{E}}_{\boldsymbol{\psi}}\left[Y_T(\boldsymbol{x}; G(\boldsymbol{x})) \mid G(\boldsymbol{x}) \neq \infty\right] - \mathop{\mathbb{E}}_{\boldsymbol{\psi}}\left[Y_T(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = \infty\right]$$

- Computationally expensive
- Estimates memorisation for a model architecture rather than a specific model
- Does not account for data order
- No insights on the training dynamics

From instance-level to batch-level memorisation

$$\tau_{\boldsymbol{x},c} \overset{\text{def}}{=} Y_c(\boldsymbol{x}; g) - Y_c(\boldsymbol{x}; \infty)$$

From instance-level to batch-level memorisation

$$\tau_{\boldsymbol{x},c} \overset{\text{def}}{=} Y_c(\boldsymbol{x}; g) - Y_c(\boldsymbol{x}; \infty)$$

$$\tau_{g,c} \overset{\text{def}}{=} \underbrace{\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; g) \mid G(\boldsymbol{x}) = g\right]} - \underbrace{\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = g\right]}$$

From instance-level to batch-level memorisation

$$\tau_{\boldsymbol{x},c} \overset{\text{def}}{=} Y_c(\boldsymbol{x}; g) - Y_c(\boldsymbol{x}; \infty)$$

$$\tau_{g,c} \overset{\text{def}}{=} \underbrace{\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; g) \mid G(\boldsymbol{x}) = g\right]}_{\text{Average performance on batch when trained on batch at step } g} - \underbrace{\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = g\right]}$$

From instance-level to batch-level memorisation

$$\tau_{\boldsymbol{x},c} \overset{\text{def}}{=} Y_c(\boldsymbol{x}; g) - Y_c(\boldsymbol{x}; \infty)$$

$$\tau_{g,c} \overset{\text{def}}{=} \underbrace{\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; g) \mid G(\boldsymbol{x}) = g\right]}_{\text{Average performance on batch when trained on batch at step } g} - \underbrace{\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = g\right]}_{\text{Average performance on batch when **not** trained on batch given it is selected for training at step } g}$$

From instance-level to batch-level memorisation

Training Batch | Validation Batch

$$\tau_{\boldsymbol{x},c} \overset{\text{def}}{=} Y_c(\boldsymbol{x}; g) - Y_c(\boldsymbol{x}; \infty)$$

$$\tau_{g,c} \overset{\text{def}}{=} \underbrace{\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; g) \mid G(\boldsymbol{x}) = g\right]}_{\text{Average performance on batch when trained on batch at step } g} - \underbrace{\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; \infty) \mid G(\boldsymbol{x}) = g\right]}_{\text{Average performance on batch when **not** trained on batch given it is selected for training at step } g}$$

From instance-level to batch-level memorisation

Training Batch Validation Batch

$$\tau_{\boldsymbol{x},c} \overset{\text{def}}{=} Y_c(\boldsymbol{x}; g) - Y_c(\boldsymbol{x}; \infty)$$

$$\tau_{g,c} \overset{\text{def}}{=} \underbrace{\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; g) \mid \blacksquare\right]}_{\text{Average performance on batch when trained on batch at step } g} - \underbrace{\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; \infty) \mid \blacksquare\right]}_{\text{Average performance on batch when \textbf{not} trained on batch given it is selected for training at step } g}$$

From instance-level to batch-level memorisation

Training Batch

Validation Batch

✅ g and 🟨 = Observable

∞ and 🟦 = Observable

❌ g and 🟦 = Not Observable

∞ and 🟨 = Not Observable

$$\tau_{x,c} \overset{\text{def}}{=} Y_c(x; g) - Y_c(x; \infty)$$

$$\tau_{g,c} \overset{\text{def}}{=} \underbrace{\mathop{\mathbb{E}}_{x}\left[Y_c(x; g) \mid \blacksquare\right]}_{\text{Average performance on batch when trained on batch at step } g} - \underbrace{\mathop{\mathbb{E}}_{x}\left[Y_c(x; \infty) \mid \blacksquare\right]}_{\text{Average performance on batch when **not** trained on batch given it is selected for training at step } g}$$

From instance-level to batch-level memorisation

Training Batch

Validation Batch

✅ g and 🟨 = Observable

∞ and 🟦 = Observable

❌ g and 🟦 = Not Observable

∞ and 🟨 = Not Observable

$$\tau_{x,c} \overset{\text{def}}{=} Y_c(x; g) - Y_c(x; \infty)$$

$$\tau_{g,c} \overset{\text{def}}{=} \underbrace{\mathop{\mathbb{E}}_{x}\left[Y_c(x; g) \mid \text{🟨} \text{✅}\right]}_{\text{Average performance on batch when trained on batch at step } g} - \underbrace{\mathop{\mathbb{E}}_{x}\left[Y_c(x; \infty) \mid \text{🟨}\right]}_{\text{Average performance on batch when \textbf{not} trained on batch given it is selected for training at step } g}$$

From instance-level to batch-level memorisation

Training Batch

Validation Batch

✅ g and 🟨 = Observable
∞ and 🟦 = Observable

❌ g and 🟦 = Not Observable
∞ and 🟨 = Not Observable

$$\tau_{x,c} \overset{\text{def}}{=} Y_c(x; g) - Y_c(x; \infty)$$

$$\tau_{g,c} \overset{\text{def}}{=} \underbrace{\mathop{\mathbb{E}}_{x}\left[Y_c(x; g) \mid \text{🟨} \text{✅}\right]}_{\text{Average performance on batch when trained on batch at step } g} - \underbrace{\mathop{\mathbb{E}}_{x}\left[Y_c(x; \infty) \mid \text{🟨} \text{❌}\right]}_{\text{Average performance on batch when \textbf{not} trained on batch given it is selected for training at step } g}$$

On the way to estimating the counterfactual

Training Batch

Validation Batch

✅ g and 🟨 = Observable

✅ ∞ and 🟦 = Observable

❌ g and 🟦 = Not Observable

❌ ∞ and 🟨 = Not Observable

$$\tau_{g,c} = \mathop{\mathbb{E}}_{\boldsymbol{x}}[Y_c(\boldsymbol{x}; g) \mid \square] - \mathop{\mathbb{E}}_{\boldsymbol{x}}[Y_c(\boldsymbol{x}; \infty) \mid \square]$$

On the way to estimating the counterfactual

Training Batch

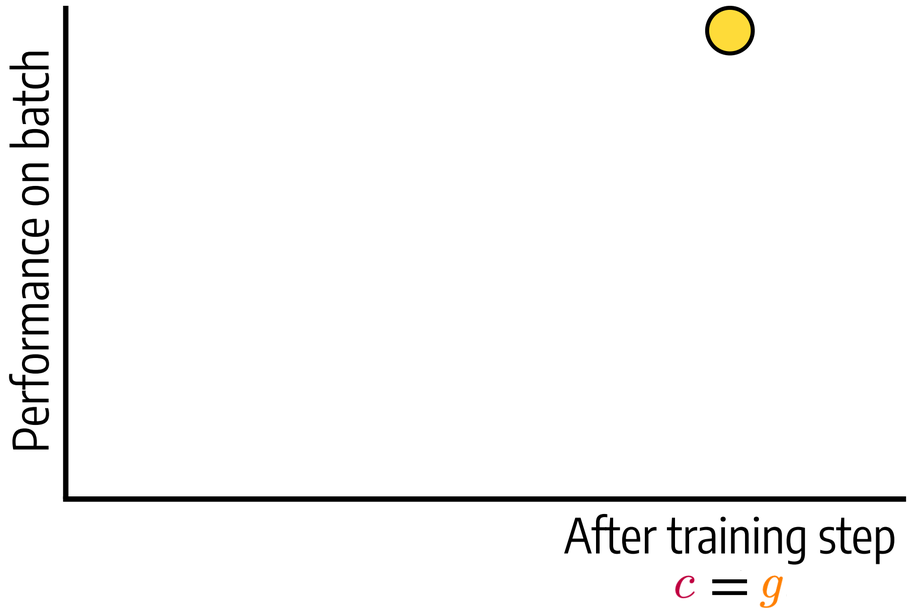
Validation Batch

✅ g and 🟨 = Observable

✅ ∞ and 🟦 = Observable

❌ g and 🟦 = Not Observable

❌ ∞ and 🟨 = Not Observable



$$\mathop{\mathbb{E}}_{x}\left[Y_c(x; g) \mid \text{🟨}\right] \text{✅}$$

$$\tau_{g,c} = \mathop{\mathbb{E}}_{x}\left[Y_c(x; g) \mid \text{🟨}\right] \text{✅} - \mathop{\mathbb{E}}_{x}\left[Y_c(x; \infty) \mid \text{🟨}\right]$$

On the way to estimating the counterfactual

Training Batch

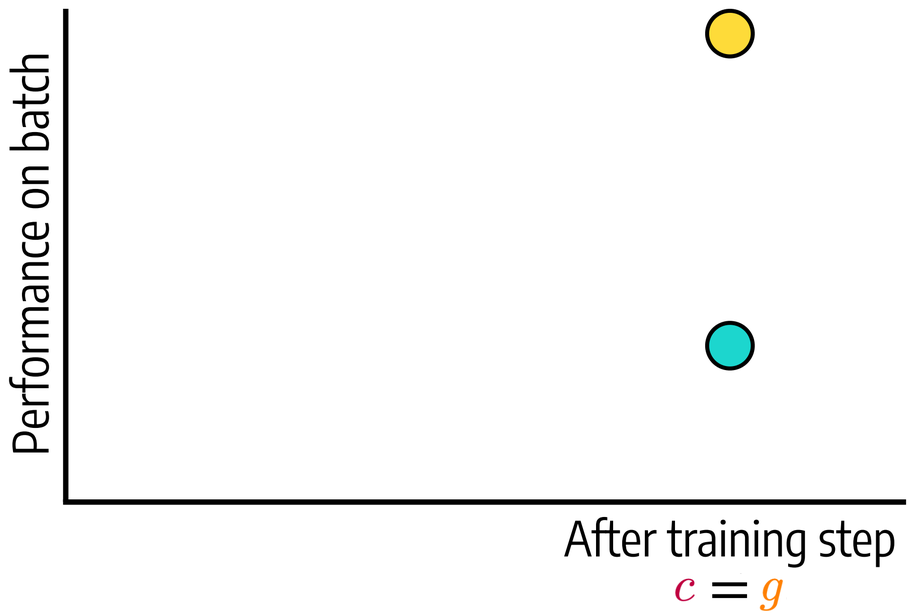
Validation Batch

✅ g and 🟨 = Observable

✅ ∞ and 🟦 = Observable

❌ g and 🟦 = Not Observable

❌ ∞ and 🟨 = Not Observable



$$\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; g) \mid \text{🟨}\right]$$

$$\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; \infty) \mid \text{🟦}\right]$$

$$\tau_{g,c} = \mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; g) \mid \text{🟨}\right] - \mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; \infty) \mid \text{🟨}\right]$$

On the way to estimating the counterfactual

Training Batch

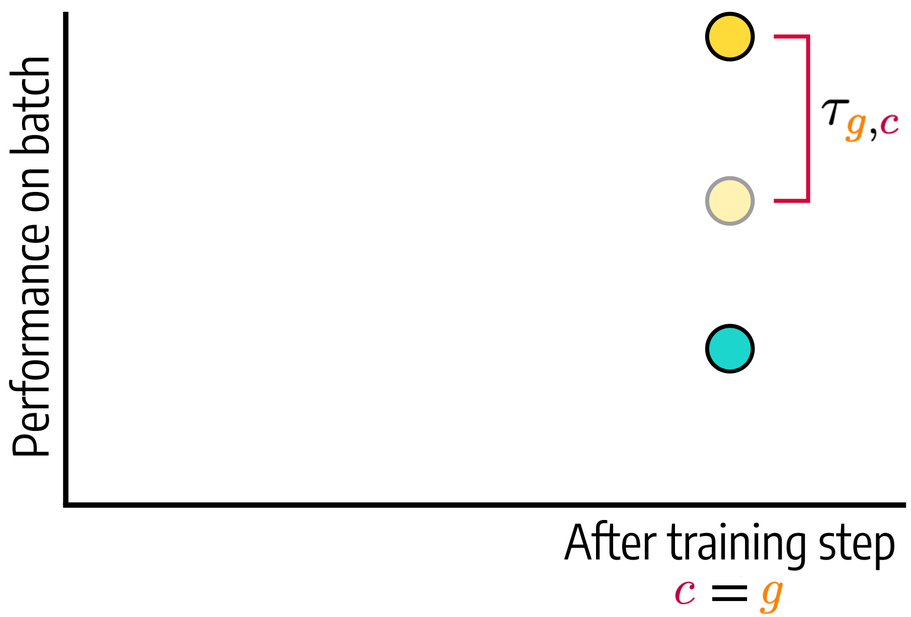
Validation Batch

g and = Observable

∞ and = Observable

g and = Not Observable

∞ and = Not Observable



$$\mathop{\mathbb{E}}_{x}\left[Y_c(x; g) \mid \right]$$

$$\mathop{\mathbb{E}}_{x}\left[Y_c(x; \infty) \mid \right]$$

$$\mathop{\mathbb{E}}_{x}\left[Y_c(x; \infty) \mid \right]$$

$$\tau_{g,c} = \mathop{\mathbb{E}}_{x}\left[Y_c(x; g) \mid \right] - \mathop{\mathbb{E}}_{x}\left[Y_c(x; \infty) \mid \right]$$

On the way to estimating the counterfactual

Training Batch

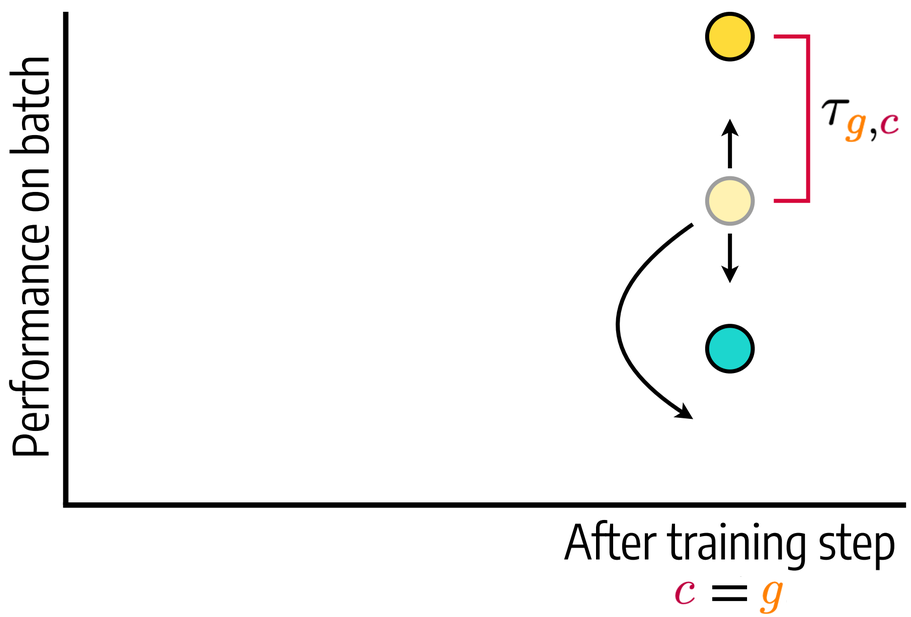
Validation Batch

✅ g and 🟨 = Observable

✅ ∞ and 🟦 = Observable

❌ g and 🟦 = Not Observable

❌ ∞ and 🟨 = Not Observable



$$\mathop{\mathbb{E}}_{x}\left[Y_c(x; g) \mid \text{🟨} \checkmark\right]$$

$$\mathop{\mathbb{E}}_{x}\left[Y_c(x; \infty) \mid \text{🟨} \times\right]$$

$$\mathop{\mathbb{E}}_{x}\left[Y_c(x; \infty) \mid \text{🟦} \checkmark\right]$$

$$\tau_{g,c} = \mathop{\mathbb{E}}_{x}\left[Y_c(x; g) \mid \text{🟨} \checkmark\right] - \mathop{\mathbb{E}}_{x}\left[Y_c(x; \infty) \mid \text{🟨} \times\right]$$

The Difference estimator

Training Batch

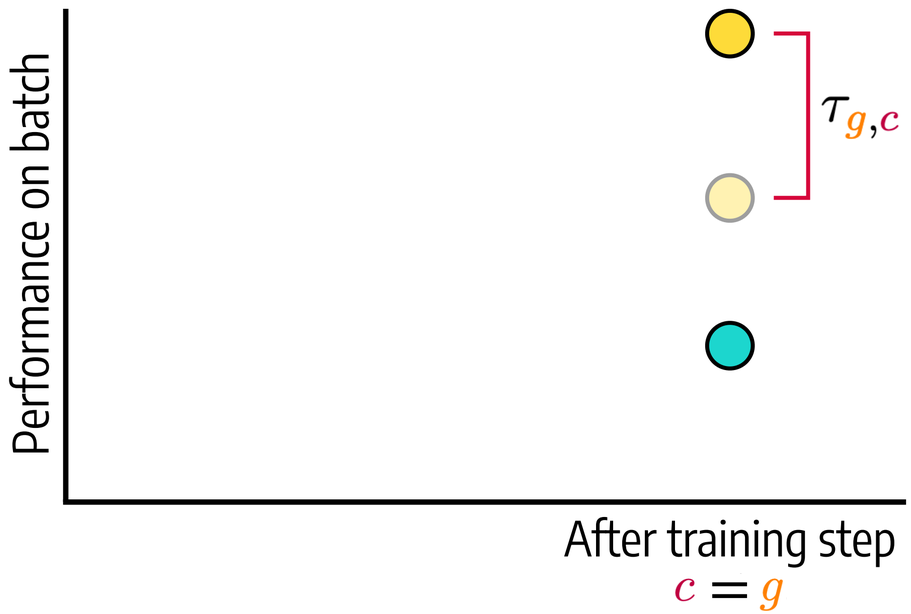
Validation Batch

✅ g and 🟨 = Observable

✅ ∞ and 🟦 = Observable

❌ g and 🟦 = Not Observable

❌ ∞ and 🟨 = Not Observable



$$\mathop{\mathbb{E}}_{x}\left[Y_c(x; g) \mid \text{🟨}\right] \quad ✅$$

$$\mathop{\mathbb{E}}_{x}\left[Y_c(x; \infty) \mid \text{🟨}\right]$$

$$\mathop{\mathbb{E}}_{x}\left[Y_c(x; \infty) \mid \text{🟦}\right] \quad ✅$$

$$\tau_{g,c} = \mathop{\mathbb{E}}_{x}\left[Y_c(x; g) \mid \text{🟨}\right] - \mathop{\mathbb{E}}_{x}\left[Y_c(x; \infty) \mid \text{🟨}\right]$$ ✅ ❌

The Difference estimator

Training Batch

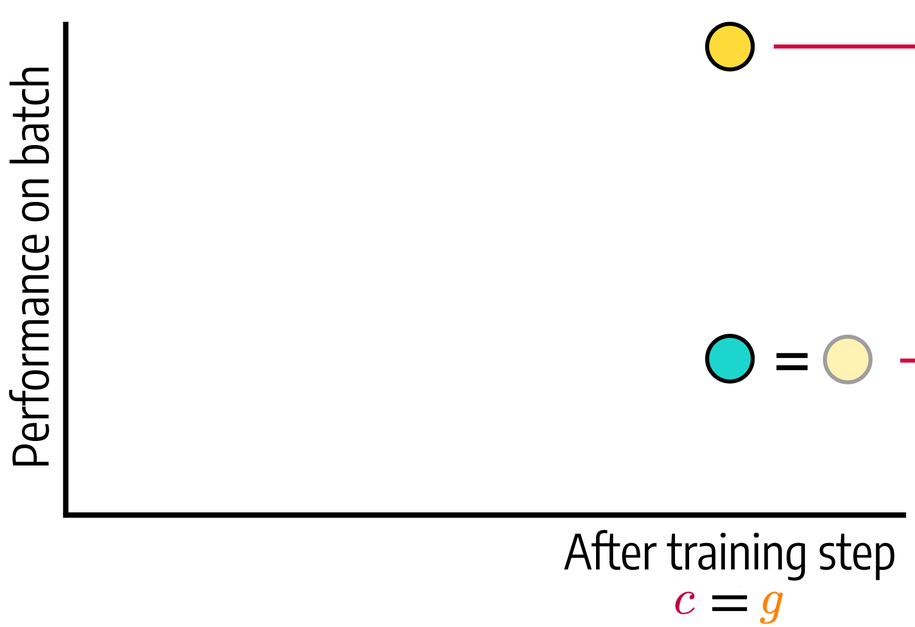
Validation Batch

g and = Observable

∞ and = Observable

g and = Not Observable

∞ and = Not Observable



$$\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; g) \mid \right]$$

$\tau_{g,c}$

$$\mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; \infty) \mid \right] = \mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; \infty) \mid \right]$$

IID Assumption

$$\tau_{g,c} = \mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; g) \mid \right] - \mathop{\mathbb{E}}_{\boldsymbol{x}}\left[Y_c(\boldsymbol{x}; \infty) \mid \right]$$

The Difference estimator

Training Batch

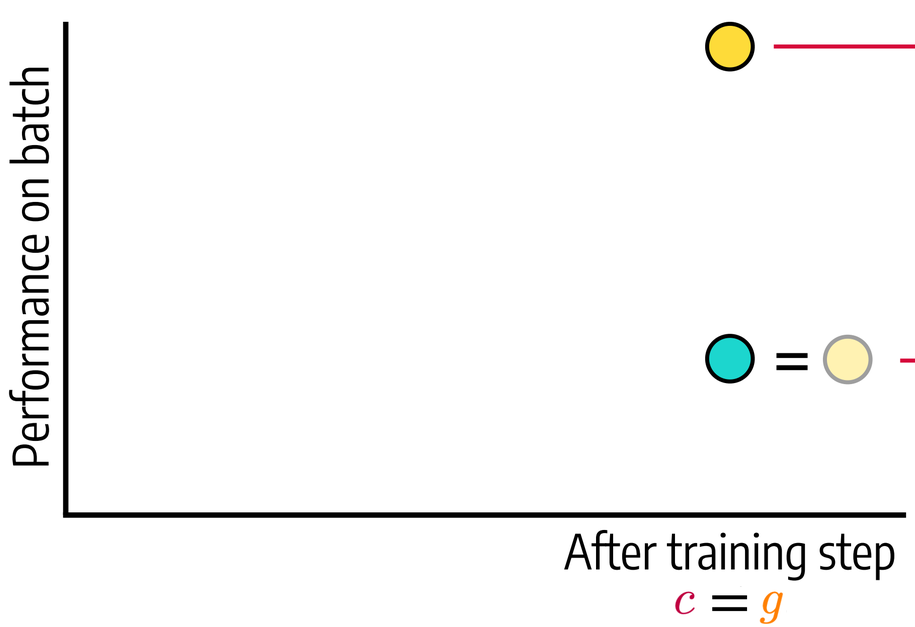
Validation Batch

✓ g and 🟨 = Observable

✓ ∞ and 🟦 = Observable

✗ g and 🟦 = Not Observable

✗ ∞ and 🟨 = Not Observable



$\tau_{g,c}$

$\mathop{\mathbb{E}}_{\boldsymbol{x}}[Y_c(\boldsymbol{x}; g) \mid$ 🟨 ✓ $]$

$\mathop{\mathbb{E}}_{\boldsymbol{x}}[Y_c(\boldsymbol{x}; \infty) \mid$ 🟦 ✓ $] = \mathop{\mathbb{E}}_{\boldsymbol{x}}[Y_c(\boldsymbol{x}; \infty) \mid$ 🟨 $]$

IID Assumption

$\tau^{\texttt{diff}}_{g,c} = \mathop{\mathbb{E}}_{\boldsymbol{x}}[Y_c(\boldsymbol{x}; g) \mid$ 🟨 ✓ $] - \mathop{\mathbb{E}}_{\boldsymbol{x}}[Y_c(\boldsymbol{x}; \infty) \mid$ 🟦 ✓ $]$

The Difference-in-Differences estimator (1/2)

Training Batch

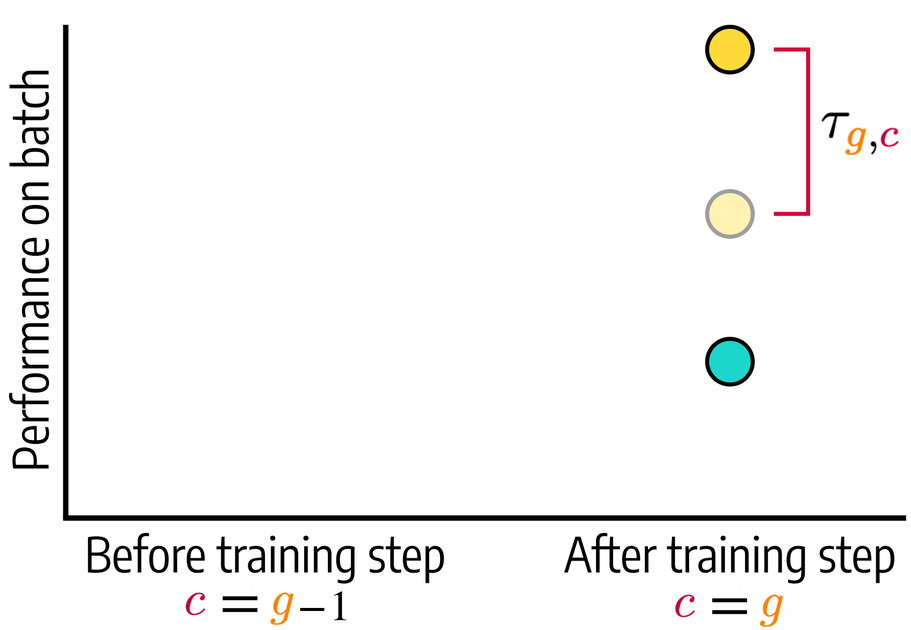
Validation Batch

✅ g and = Observable

✅ ∞ and = Observable

❌ g and = Not Observable

❌ ∞ and = Not Observable



$$\mathop{\mathbb{E}}_{x}\left[Y_c(x; g) \mid \right]$$ ✅

$$\mathop{\mathbb{E}}_{x}\left[Y_c(x; \infty) \mid \right]$$ ✅

The Difference-in-Differences estimator (1/2)

Training Batch

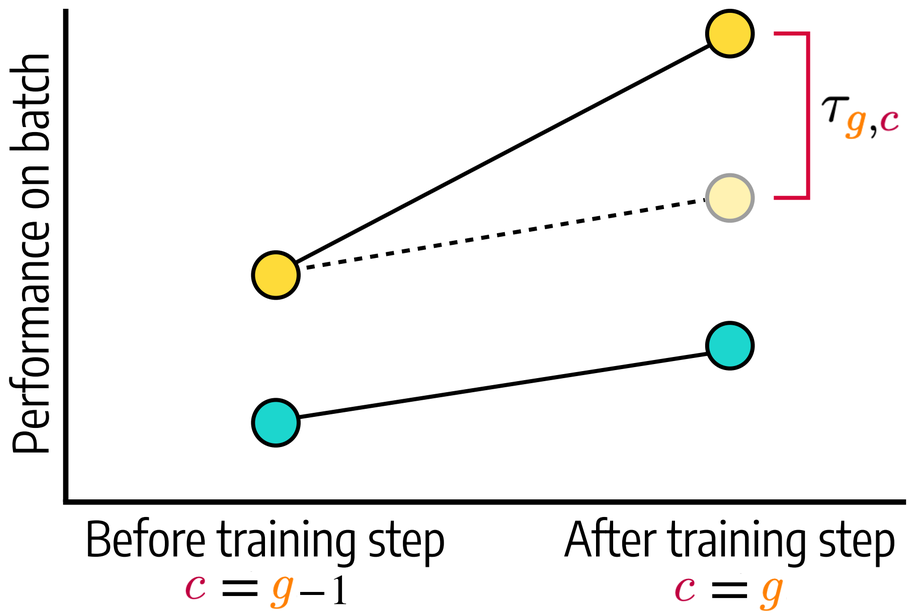
Validation Batch

✅ g and 🟨 = Observable

✅ ∞ and 🟦 = Observable

❌ g and 🟦 = Not Observable

❌ ∞ and 🟨 = Not Observable



$$\mathop{\mathbb{E}}_{x}\left[Y_c(x; g) \mid \text{🟨} \checkmark\right]$$

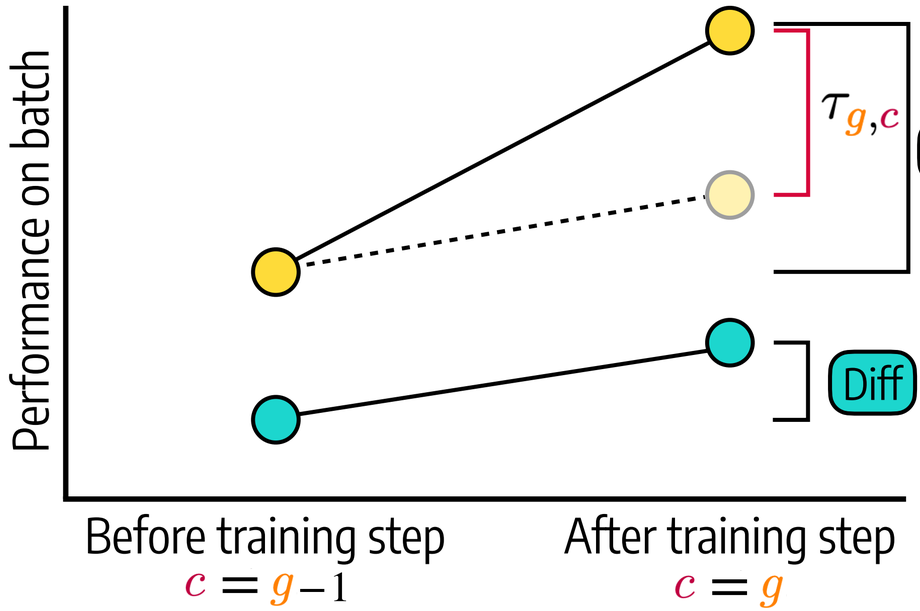
$$\mathop{\mathbb{E}}_{x}\left[Y_c(x; \infty) \mid \text{🟦} \checkmark\right]$$

The Difference-in-Differences estimator (1/2)

Training Batch | Validation Batch

✅ g and 🟨 = Observable
✅ ∞ and 🟦 = Observable

❌ g and 🟦 = Not Observable
❌ ∞ and 🟨 = Not Observable



Diff = $\mathop{\mathbb{E}}_{x}\left[Y_c(x; g) \mid \text{🟨}\right] - \mathop{\mathbb{E}}_{x}\left[Y_{g-1}(x; g) \mid \text{🟨}\right]$

Diff = $\mathop{\mathbb{E}}_{x}\left[Y_c(x; \infty) \mid \text{🟦}\right] - \mathop{\mathbb{E}}_{x}\left[Y_{g-1}(x; \infty) \mid \text{🟦}\right]$

The Difference-in-Differences estimator (2/2)

Training Batch

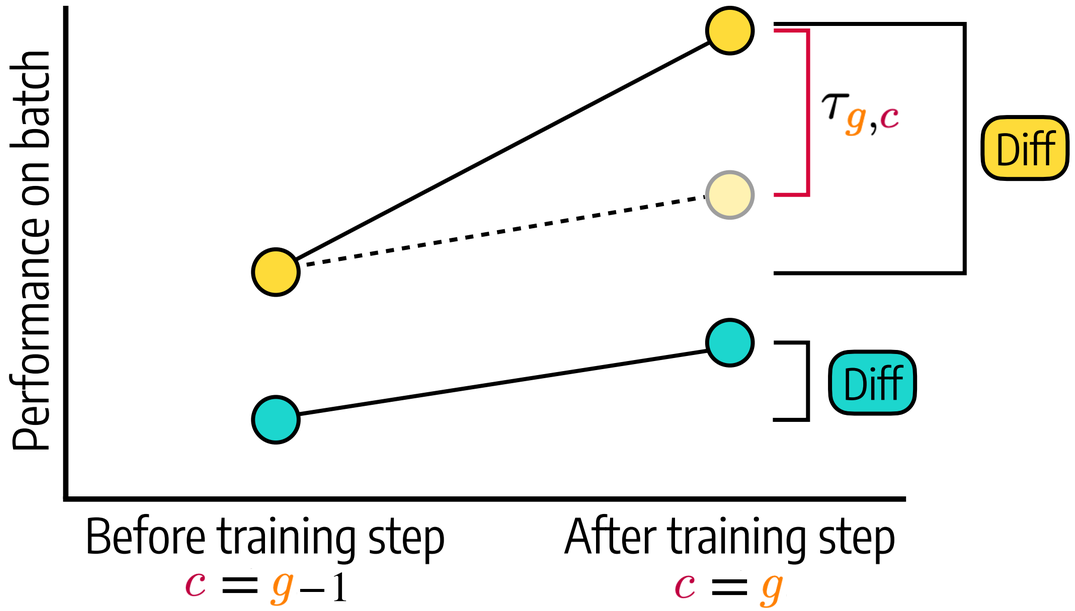
Validation Batch

✅ g and = Observable

∞ and = Observable

❌ g and = Not Observable

∞ and = Not Observable



The Difference-in-Differences estimator (2/2)

Training Batch

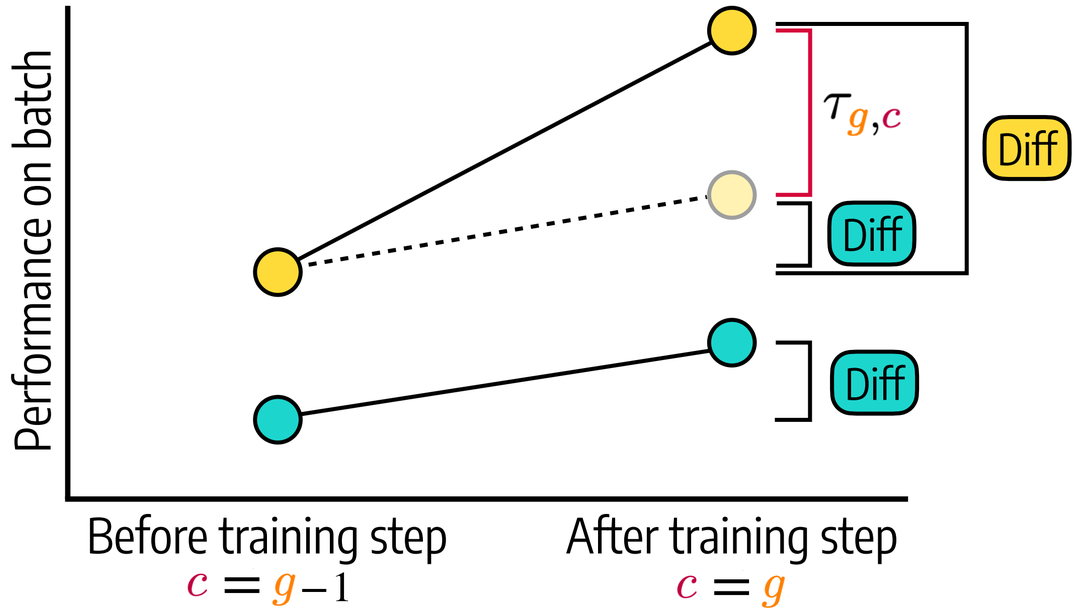
Validation Batch

✅ g and = Observable

∞ and = Observable

❌ g and = Not Observable

∞ and = Not Observable



Parallel Trend Assumption

The Difference-in-Differences estimator (2/2)

Training Batch

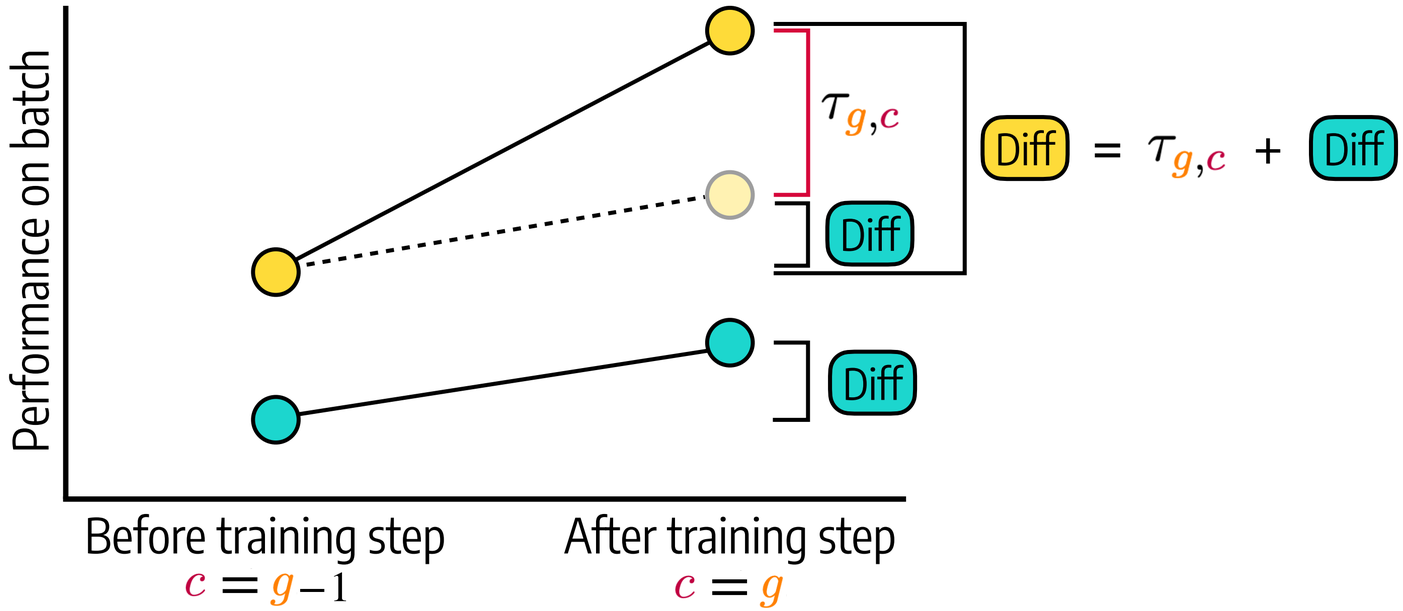
Validation Batch

✅ g and = Observable

∞ and = Observable

❌ g and = Not Observable

∞ and = Not Observable



Diff = $\tau_{g,c}$ + Diff

Parallel Trend Assumption

The Difference-in-Differences estimator (2/2)

Training Batch

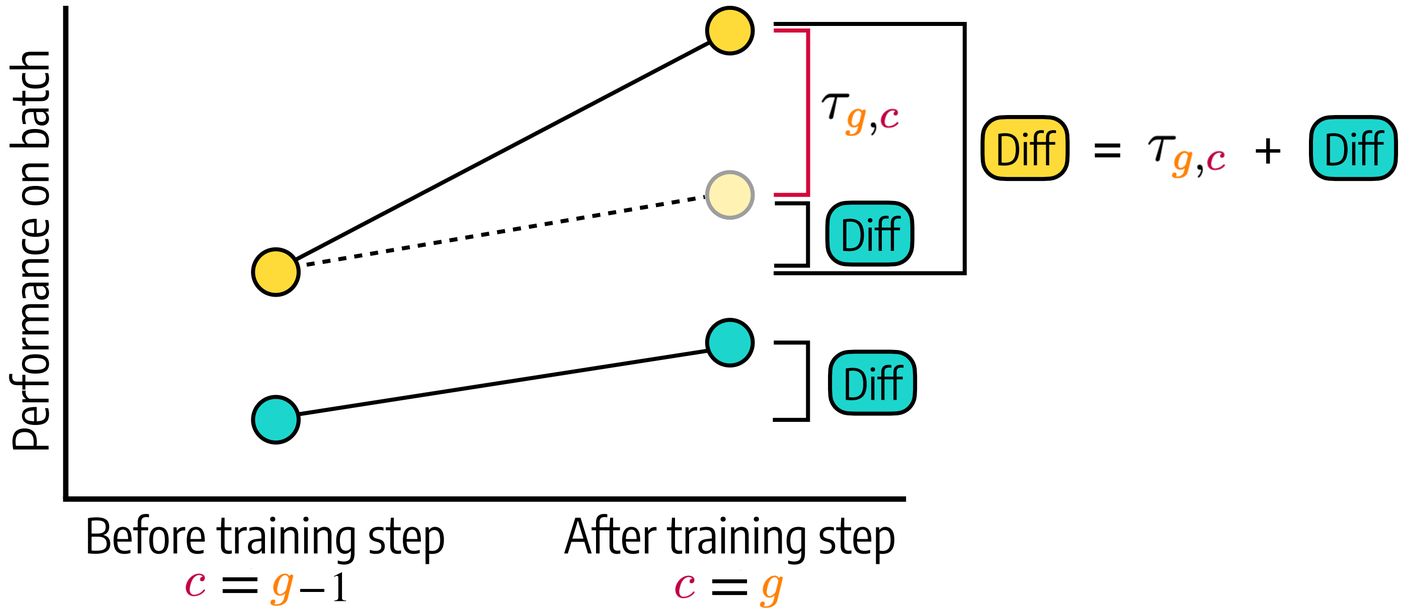
Validation Batch

✅ g and = Observable

∞ and = Observable

❌ g and = Not Observable

∞ and = Not Observable



Diff = $\tau_{g,c}$ + Diff

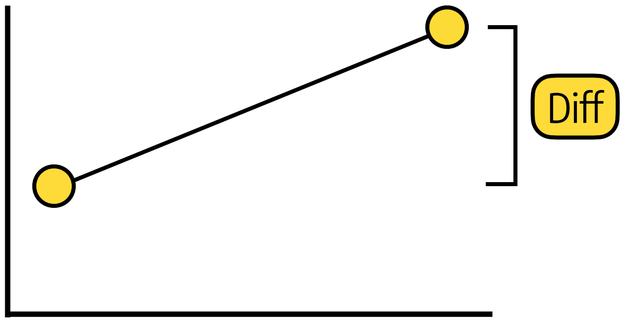
Parallel Trend Assumption

$$\tau^{\texttt{did}}_{g,c} = \text{Diff} - \text{Diff}$$

Difference-in-Differences in practice

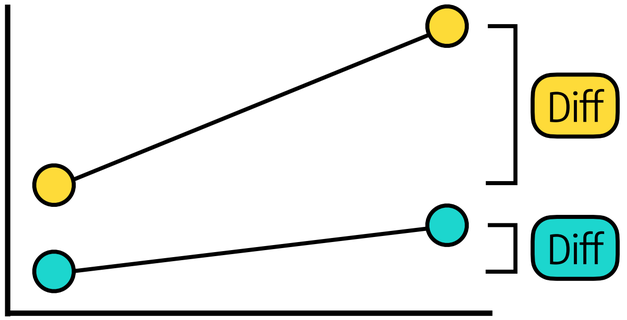
Difference-in-Differences in practice

$$\tau^{\texttt{did}}_{g,c} = \mathop{\mathbb{E}}_{\boldsymbol{x}}[Y_c(\boldsymbol{x}; g) - Y_{g-1}(\boldsymbol{x}; g) \,|\, \square]$$



Difference-in-Differences in practice

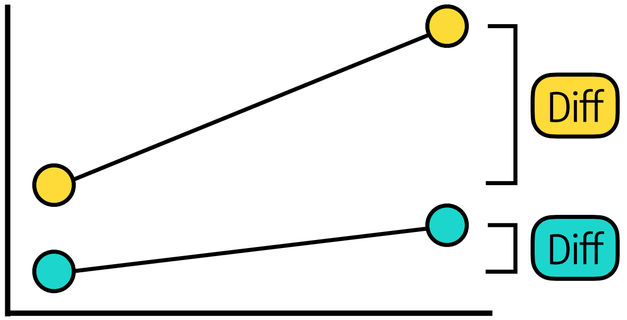
$$\tau^{\texttt{did}}_{g,c} = \mathop{\mathbb{E}}_{\boldsymbol{x}}[Y_c(\boldsymbol{x}; g) - Y_{g-1}(\boldsymbol{x}; g) \mid \square\!\checkmark] - \mathop{\mathbb{E}}_{\boldsymbol{x}}[Y_c(\boldsymbol{x}; \infty) - Y_{g-1}(\boldsymbol{x}; \infty) \mid \square\!\checkmark]$$



Difference-in-Differences in practice

$$\tau^{\texttt{did}}_{g,c} = \mathop{\mathbb{E}}_{\boldsymbol{x}}[Y_c(\boldsymbol{x}; g) - Y_{g-1}(\boldsymbol{x}; g) \mid \boxed{}] - \mathop{\mathbb{E}}_{\boldsymbol{x}}[Y_c(\boldsymbol{x}; \infty) - Y_{g-1}(\boldsymbol{x}; \infty) \mid \boxed{}]$$

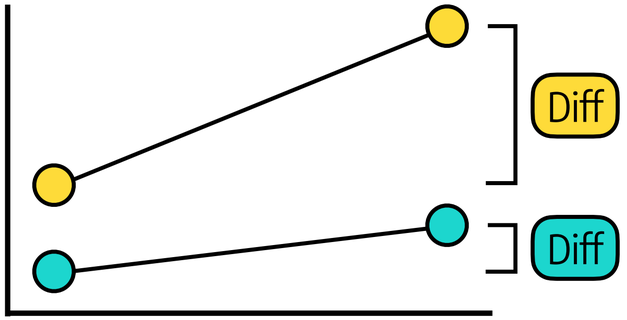
$$\widehat{\tau}^{\texttt{did}}_{g,c} = \underbrace{\left(\overline{Y}_c(g) - \overline{Y}_{g-1}(g)\right)}_{\text{Performance difference on training batches}}$$



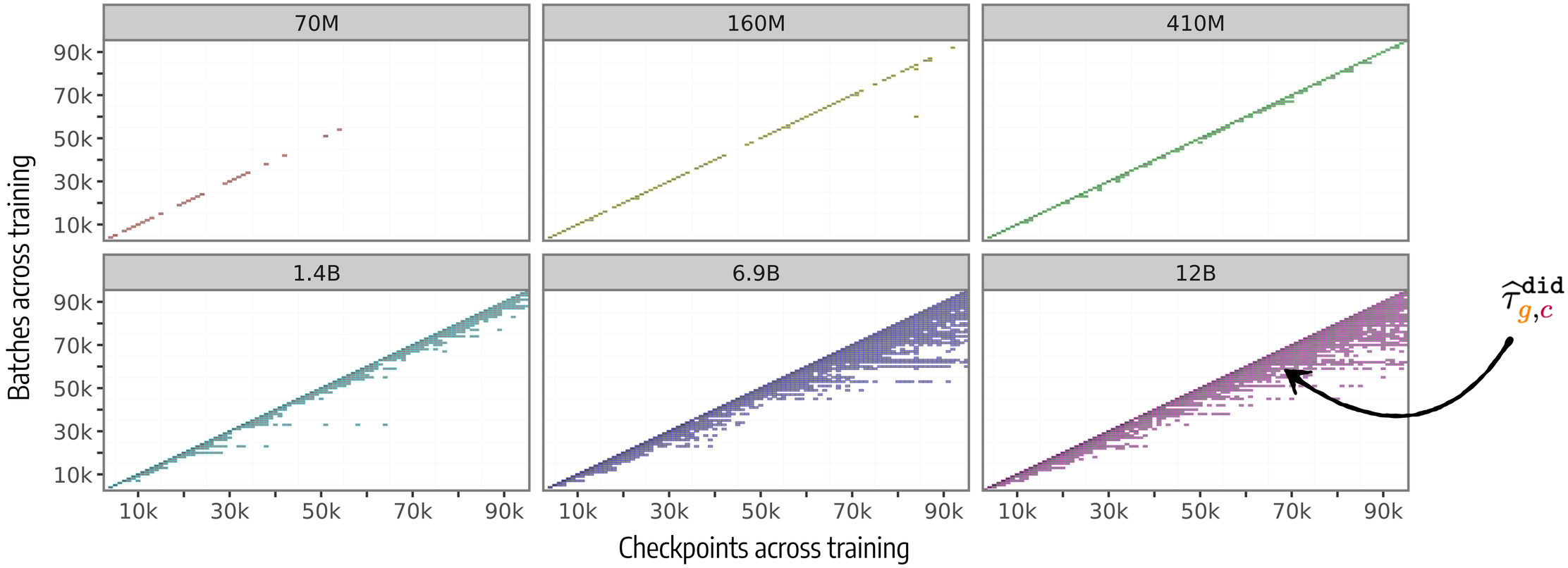
Difference-in-Differences in practice

$$\tau^{\texttt{did}}_{g,c} = \mathop{\mathbb{E}}_{\boldsymbol{x}}[Y_c(\boldsymbol{x}; g) - Y_{g-1}(\boldsymbol{x}; g) \mid \boxed{}] - \mathop{\mathbb{E}}_{\boldsymbol{x}}[Y_c(\boldsymbol{x}; \infty) - Y_{g-1}(\boldsymbol{x}; \infty) \mid \boxed{}]$$

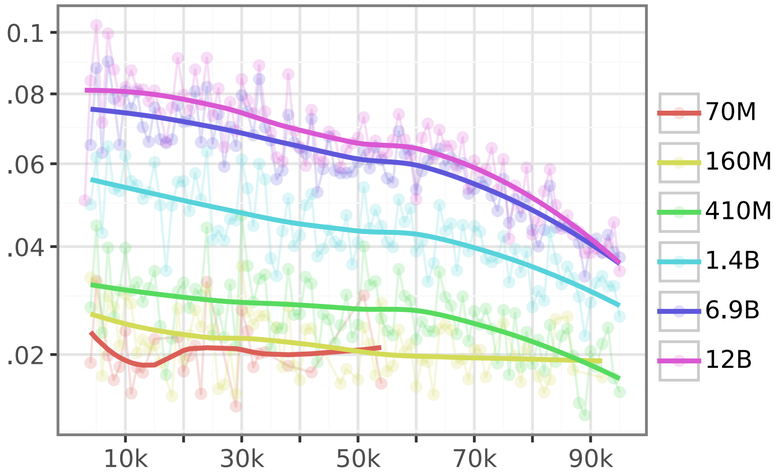
$$\widehat{\tau}^{\texttt{did}}_{g,c} = \underbrace{\left(\overline{Y}_c(g) - \overline{Y}_{g-1}(g)\right)}_{\text{Performance difference on training batches}} - \underbrace{\left(\overline{Y}_c(\infty) - \overline{Y}_{g-1}(\infty)\right)}_{\text{Performance difference on validation batches}}$$



Constructing the memorisation profile

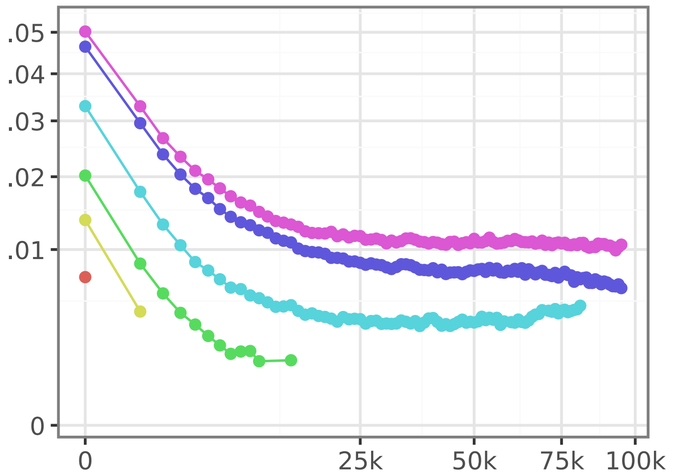


Aggregating the memorisation profile



Checkpoints/Batches across training ($g = c$)

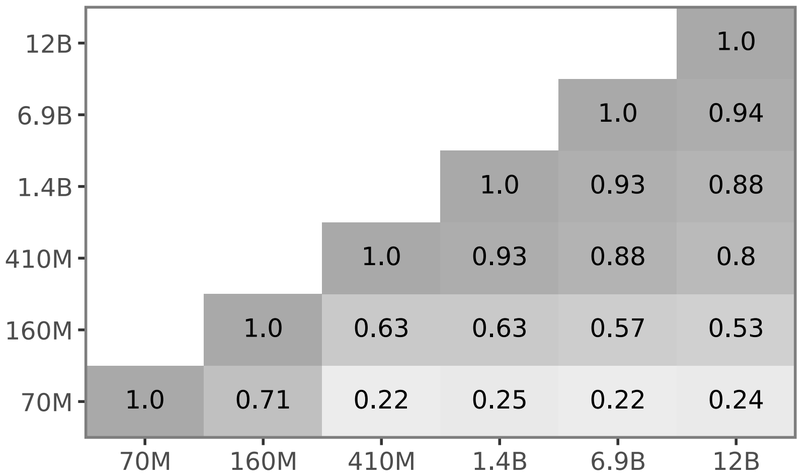
Instantaneous memorisation of a batch at the checkpoint is first seen



Checkpoints after batch is seen ($g > c$)

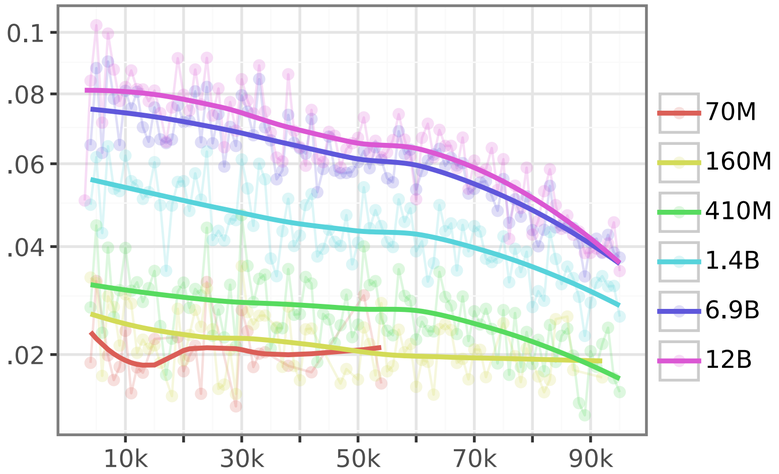
Average persistent memorisation of a batch per step after it has been seen*

*(only averaging across batches that have been trained that much)



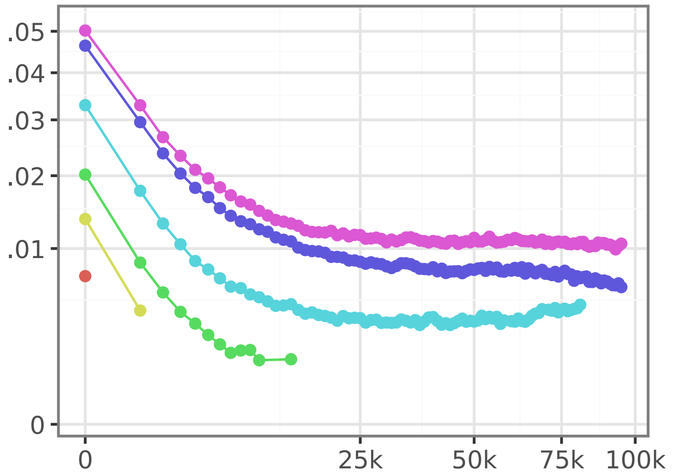
Pearson correlation between the memorisation profiles of different model sizes

Aggregating the memorisation profile



Checkpoints/Batches across training ($g = c$)

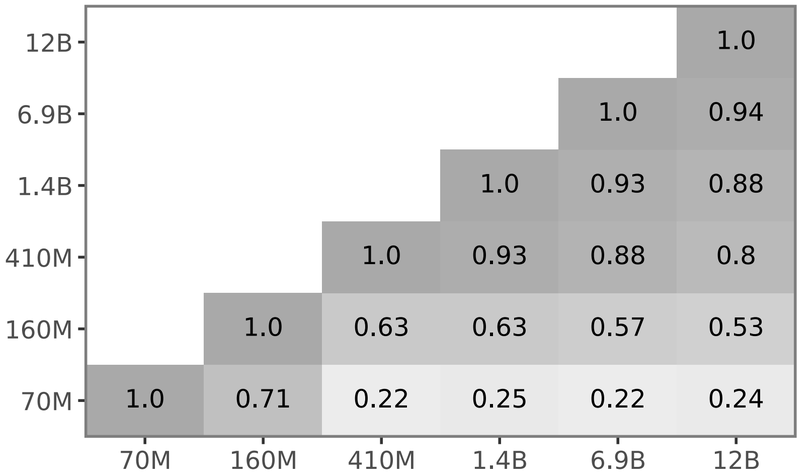
Instantaneous memorisation of a batch at the checkpoint is first seen



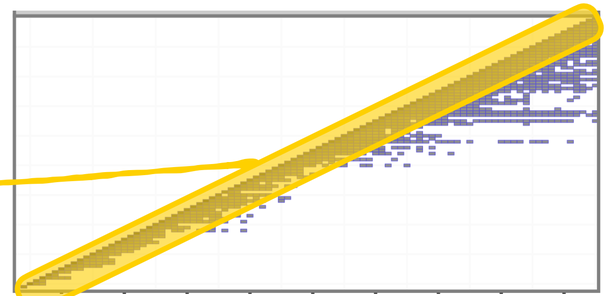
Checkpoints after batch is seen ($g > c$)

Average persistent memorisation of a batch per step after it has been seen*

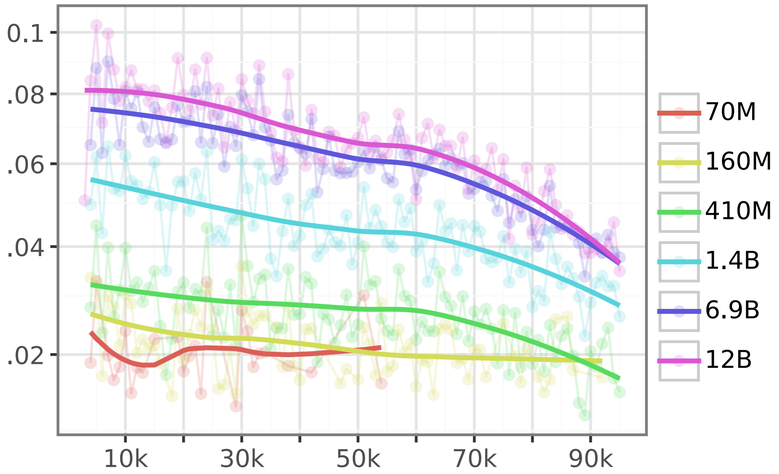
*(only averaging across batches that have been trained that much)



Pearson correlation between the memorisation profiles of different model sizes

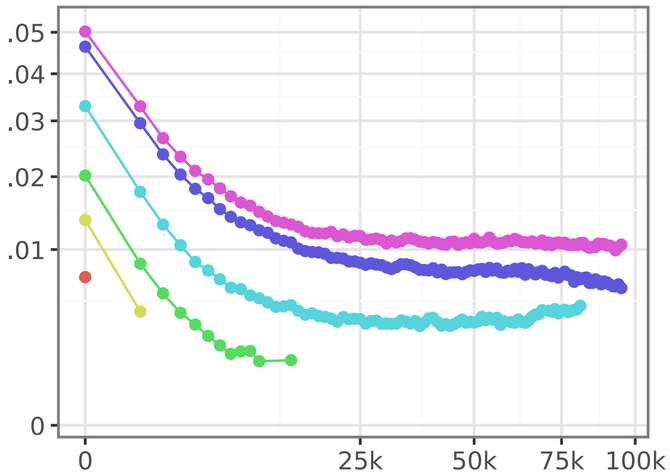


Aggregating the memorisation profile



Checkpoints/Batches across training ($g = c$)

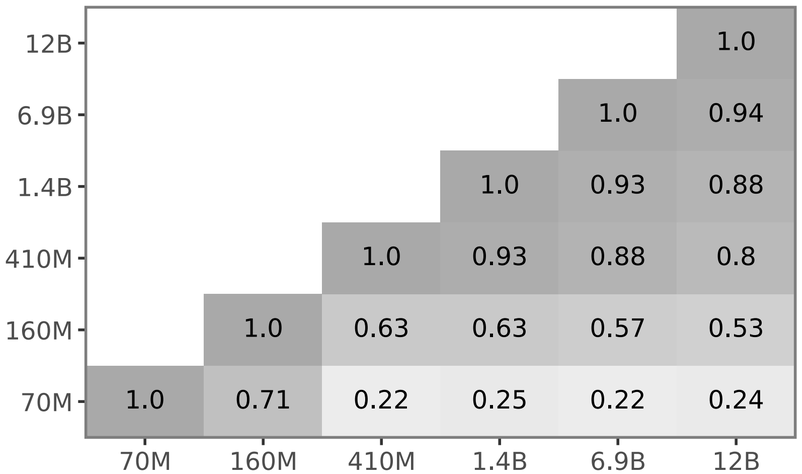
Instantaneous memorisation of a batch at the checkpoint is first seen



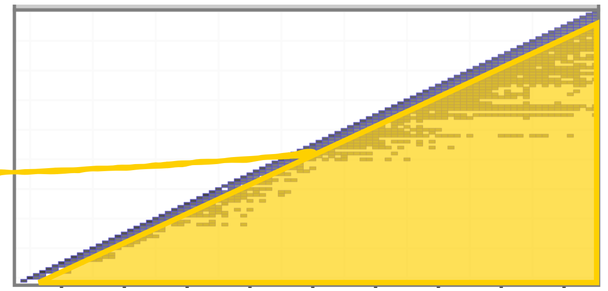
Checkpoints after batch is seen ($g > c$)

Average persistent memorisation of a batch per step after it has been seen*

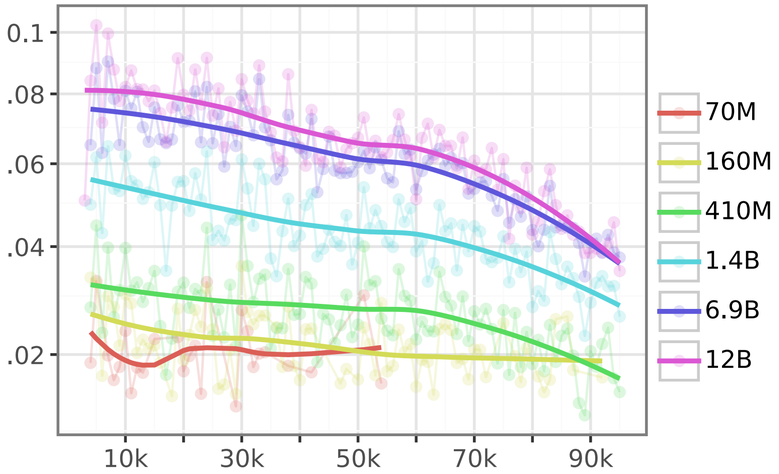
*(only averaging across batches that have been trained that much)



Pearson correlation between the memorisation profiles of different model sizes

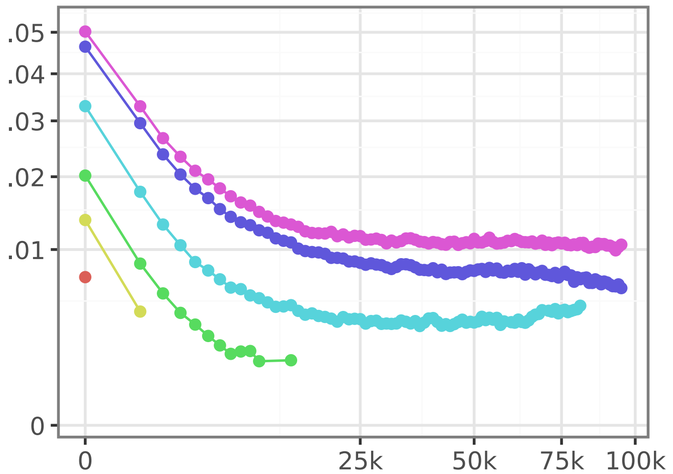


Aggregating the memorisation profile



Checkpoints/Batches across training ($g = c$)

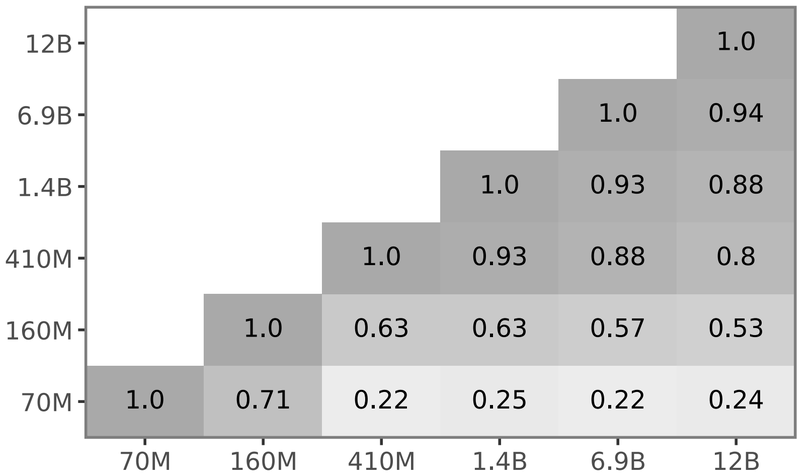
Instantaneous memorisation of a batch at the checkpoint is first seen



Checkpoints after batch is seen ($g > c$)

Average persistent memorisation of a batch per step after it has been seen*

*(only averaging across batches that have been trained that much)



Pearson correlation between the memorisation profiles of different model sizes

Causal Estimation of Tokenisation Bias

ACL 2025


Pietro Lesci


Clara Meister


Thomas Hofmann


Andreas Vlachos


Tiago Pimentel

What is tokenisation? And why do we need it?

What is tokenisation? And why do we need it?

“Today is a sunny day!”

What is tokenisation? And why do we need it?

“Today is a sunny day!”

Unicode: [U+0054, U+006F, U+0064, U+0061, U+0079, U+0020, …]

UTF-8: [54, 6F, 64, 61, 79, 20, 69, 73, 20, 61, 20, 73, 75, 6E, 6E, 79, 20, 64, 61, 79, 21]

Graphemes: [“T” , “o”, “d”, “a”, “y”, “_”, “i”, “s”, “_”, “a”, “_”, “s”, “u”, “n”, “n”, “y”, “_”, “d”, “a”, “y”, “!”]

What is tokenisation? And why do we need it?

“Today is a sunny day!”

Small vocabulary
Long sequences

Unicode: [U+0054, U+006F, U+0064, U+0061, U+0079, U+0020, …]

UTF-8: [54, 6F, 64, 61, 79, 20, 69, 73, 20, 61, 20, 73, 75, 6E, 6E, 79, 20, 64, 61, 79, 21]

Graphemes: [“T” , “o”, “d”, “a”, “y”, “_”, “i”, “s”, “_”, “a”, “_”, “s”, “u”, “n”, “n”, “y”, “_”, “d”, “a”, “y”, “!”]

What is tokenisation? And why do we need it?

"Today is a sunny day!"

Small vocabulary
Long sequences

Unicode: [U+0054, U+006F, U+0064, U+0061, U+0079, U+0020, ...]

UTF-8: [54, 6F, 64, 61, 79, 20, 69, 73, 20, 61, 20, 73, 75, 6E, 6E, 79, 20, 64, 61, 79, 21]

Graphemes: ["T" , "o", "d", "a", "y", "_", "i", "s", "_", "a", "_", "s", "u", "n", "n", "y", "_", "d", "a", "y", "!"]

Words: ["Today", "_", "is", "a", "_", "sunny", "_", "day", "!"]

What is tokenisation? And why do we need it?

"Today is a sunny day!"

Small vocabulary
Long sequences

Unicode: [U+0054, U+006F, U+0064, U+0061, U+0079, U+0020, ...]

UTF-8: [54, 6F, 64, 61, 79, 20, 69, 73, 20, 61, 20, 73, 75, 6E, 6E, 79, 20, 64, 61, 79, 21]

Graphemes: ["T" , "o", "d", "a", "y", "_", "i", "s", "_", "a", "_", "s", "u", "n", "n", "y", "_", "d", "a", "y", "!"]

Short sequences
Large vocabulary

Words: ["Today", "_", "is", "a", "_", "sunny", "_", "day", "!"]

What is tokenisation? And why do we need it?

"Today is a sunny day!"

Small vocabulary
Long sequences

Unicode: [U+0054, U+006F, U+0064, U+0061, U+0079, U+0020, ...]

UTF-8: [54, 6F, 64, 61, 79, 20, 69, 73, 20, 61, 20, 73, 75, 6E, 6E, 79, 20, 64, 61, 79, 21]

Graphemes: ["T" , "o", "d", "a", "y", "_", "i", "s", "_", "a", "_", "s", "u", "n", "n", "y", "_", "d", "a", "y", "!"]

Short sequences
Large vocabulary

Words: ["Today", "_", "is", "a", "_", "sunny", "_", "day", "!"]

Trade-off

Subwords: ["_To", "day", "_is", "_a", "_sun", "ny", "_day", "!"]

What is tokenisation? And why do we need it?

"Today is a sunny day!"

Small vocabulary
Long sequences

Unicode: [U+0054, U+006F, U+0064, U+0061, U+0079, U+0020, ...]

UTF-8: [54, 6F, 64, 61, 79, 20, 69, 73, 20, 61, 20, 73, 75, 6E, 6E, 79, 20, 64, 61, 79, 21]

Graphemes: ["T" , "o", "d", "a", "y", "_", "i", "s", "_", "a", "_", "s", "u", "n", "n", "y", "_", "d", "a", "y", "!"]

Short sequences
Large vocabulary

Words: ["Today", "_", "is", "a", "_", "sunny", "_", "day", "!"]

Trade-off

Subwords: ["_To", "day", "_is", "_a", "_sun", "ny", "_day", "!"]

Tokenise with $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$

What is a tokeniser $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$?

What is a tokeniser $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$?

Vocabulary $\mathcal{V}$

Contains the subwords available to the model

What is a bottom-up tokeniser $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$?

Vocabulary $\mathcal{V}$

Contains the subwords available to the model

What is a bottom-up tokeniser $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$?

Vocabulary $\mathcal{V}$

Contains the subwords available to the model

ID	Subwords	Merges
32	_	
33	!	
84	T	
97	a	
100	d	Initial Alphabet
110	n	
115	s	
117	u	
121	y	

What is a bottom-up tokeniser $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$?

Vocabulary $\mathcal{V}$

Contains the subwords available to the model

ID	Subwords	Merges
32	_	Initial Alphabet
33	!	
84	T	
97	a	
100	d	
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

What is a bottom-up tokeniser $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$?

Vocabulary $\mathcal{V}$

Contains the subwords available to the model

Tokenisation Function τ

Breaks down strings into character spans and maps them to vocabulary items

ID	Subwords	Merges
32	_	
33	!	
84	T	
97	a	
100	d	Initial Alphabet
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

What is a bottom-up tokeniser $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$?

Vocabulary $\mathcal{V}$

Contains the subwords available to the model

Tokenisation Function τ

Breaks down strings into character spans and maps them to vocabulary items

ID	Subwords	Merges
32	_	
33	!	
84	T	
97	a	
100	d	Initial Alphabet
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

τ(“Today is a sunny day!”)

What is a bottom-up tokeniser $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$?

Vocabulary $\mathcal{V}$

Contains the subwords available to the model

Tokenisation Function τ

Breaks down strings into character spans and maps them to vocabulary items

ID	Subwords	Merges
32	_	
33	!	
84	T	
97	a	
100	d	Initial Alphabet
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

τ(“Today is a sunny day!”)

1. Pre-tokenisation*:

   = τ([“_Today”, “_is”, “_a”, “_sunny”, “_day”, “!”])

What is a bottom-up tokeniser $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$?

Vocabulary $\mathcal{V}$

Contains the subwords available to the model

Tokenisation Function τ

Breaks down strings into character spans and maps them to vocabulary items

ID	Subwords	Merges
32	_	
33	!	
84	T	
97	a	
100	d	Initial Alphabet
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

τ(“Today is a sunny day!”)

1. Pre-tokenisation*:

   = τ([“_Today”, “_is”, “_a”, “_sunny”, “_day”, “!”])

*“wHy DoNt YoU jUsT uSe ThE lLaMa ToKeNiZeR??” (Arnett, 2024)

What is a bottom-up tokeniser $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$?

Vocabulary $\mathcal{V}$

Contains the subwords available to the model

Tokenisation Function τ

Breaks down strings into character spans and maps them to vocabulary items

ID	Subwords	Merges
32	_	
33	!	
84	T	
97	a	
100	d	Initial Alphabet
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

τ(“Today is a sunny day!”)

1. Pre-tokenisation*:

   = τ([“_Today”, “_is”, “_a”, “_sunny”, “_day”, “!”])

   = [τ(“_Today”), τ(“_is”), τ(“_a”), τ(“_sunny”), τ(“_day”), τ(“!”)]

*“wHy DoNt YoU jUsT uSe ThE lLaMa ToKeNiZeR??” (Arnett, 2024)

What is a bottom-up tokeniser $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$?

Vocabulary $\mathcal{V}$

Contains the subwords available to the model

Tokenisation Function τ

Breaks down strings into character spans and maps them to vocabulary items

ID	Subwords	Merges
32	_	
33	!	
84	T	
97	a	
100	d	Initial Alphabet
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

τ(“Today is a sunny day!”)

1. Pre-tokenisation*:

   = τ([“_Today”, “_is”, “_a”, “_sunny”, “_day”, “!”])

   = [τ(“_Today”), τ(“_is”), τ(“_a”), τ(“_sunny”), τ(“_day”), τ(“!”)]

2. Apply tokenisation function (e.g., longest-prefix match):

*“wHy DoNt YoU jUsT uSe ThE lLaMa ToKeNiZeR??” (Arnett, 2024)

What is a bottom-up tokeniser $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$?

Vocabulary $\mathcal{V}$

Contains the subwords available to the model

Tokenisation Function τ

Breaks down strings into character spans and maps them to vocabulary items

ID	Subwords	Merges
32	_	
33	!	
84	T	
97	a	
100	d	Initial Alphabet
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

τ(“Today is a sunny day!”)

1. Pre-tokenisation*:

   = τ([“_Today”, “_is”, “_a”, “_sunny”, “_day”, “!”])

   = [τ(“_Today”), τ(“_is”), τ(“_a”), τ(“_sunny”), τ(“_day”), τ(“!”)]

2. Apply tokenisation function (e.g., longest-prefix match):

*“wHy DoNt YoU jUsT uSe ThE lLaMa ToKeNiZeR??” (Arnett, 2024)

What is a bottom-up tokeniser $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$?

Vocabulary $\mathcal{V}$

Contains the subwords available to the model

Tokenisation Function τ

Breaks down strings into character spans and maps them to vocabulary items

ID	Subwords	Merges
32	_	
33	!	
84	T	
97	a	
100	d	Initial Alphabet
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

τ(“Today is a sunny day!”)

1. Pre-tokenisation*:

   = τ([“_Today”, “_is”, “_a”, “_sunny”, “_day”, “!”])

   = [τ(“_Today”), τ(“_is”), τ(“_a”), τ(“_sunny”), τ(“_day”), τ(“!”)]

2. Apply tokenisation function (e.g., longest-prefix match):

   = [_To, day, _is, _a, **_sunny**, _day, !]

*“wHy DoNt YoU jUsT uSe ThE lLaMa ToKeNiZeR??” (Arnett, 2024)

What is a bottom-up tokeniser $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$?

Vocabulary $\mathcal{V}$

Contains the subwords available to the model

Tokenisation Function τ

Breaks down strings into character spans and maps them to vocabulary items

ID	Subwords	Merges
32	_	Initial Alphabet
33	!	
84	T	
97	a	
100	d	
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

τ(“Today is a sunny day!”)

1. Pre-tokenisation*:

   = τ([“_Today”, “_is”, “_a”, “_sunny”, “_day”, “!”])

   = [τ(“_Today”), τ(“_is”), τ(“_a”), τ(“_sunny”), τ(“_day”), τ(“!”)]

2. Apply tokenisation function (e.g., longest-prefix match):

   = [_To, day, _is, _a, **_sunny**, _day, !]

   = [299, 318, 300, 256, **261**, 324, 33]

*“wHy DoNt YoU jUsT uSe ThE lLaMa ToKeNiZeR??” (Arnett, 2024)

What is a bottom-up tokeniser $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$?

Vocabulary $\mathcal{V}$
Contains the subwords available to the model

Tokenisation Function τ
Breaks down strings into character spans and maps them to vocabulary items

Detokenisation Function τ^{-1}
Converts sequences of subwords to sequences of characters

ID	Subwords	Merges
32	_	
33	!	
84	T	
97	a	
100	d	Initial Alphabet
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

τ("Today is a sunny day!")

1. Pre-tokenisation*:
   = τ(["_Today", "_is", "_a", "_sunny", "_day", "!"])
   = [τ("_Today"), τ("_is"), τ("_a"), τ("_sunny"), τ("_day"), τ("!")]

2. Apply tokenisation function (e.g., longest-prefix match):
   = [_To, day, _is, _a, **_sunny**, _day, !]
   = [299, 318, 300, 256, **261**, 324, 33]

*"wHy DoNt YoU jUsT uSe ThE lLaMa ToKeNiZeR??" (Arnett, 2024)

What is a bottom-up tokeniser $\mathbb{T}(\mathcal{V}, \tau, \tau^{-1})$?

Vocabulary $\mathcal{V}$
Contains the subwords available to the model

Tokenisation Function τ
Breaks down strings into character spans and maps them to vocabulary items

Detokenisation Function τ^{-1}
Converts sequences of subwords to sequences of characters

ID	Subwords	Merges
32	_	
33	!	
84	T	
97	a	
100	d	Initial Alphabet
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

τ(“Today is a sunny day!”)

1. Pre-tokenisation*:
   = τ([“_Today”, “_is”, “_a”, “_sunny”, “_day”, “!”])
   = [τ(“_Today”), τ(“_is”), τ(“_a”), τ(“_sunny”), τ(“_day”), τ(“!”)]

2. Apply tokenisation function (e.g., longest-prefix match):
   = [_To, day, _is, _a, **_sunny**, _day, !]
   = [299, 318, 300, 256, **261**, 324, 33]

τ^{-1}([299, 318, 300, 256, 261, 324, 33])
= τ^{-1}([_To, day, _is, _a, _sunny, _day, !])
= “Today is a sunny day!”

*“wHy DoNt YoU jUsT uSe ThE lLaMa ToKeNiZeR??” (Arnett, 2024)

Language modelling and tokenisation

Probabilities over character-strings

Probabilities over subword-strings

Language modelling and tokenisation

Probabilities over character-strings

Probabilities over subword-strings

P(“Today is a sunny day!”)

= P(“T”)

× P(“o” | “T”)

× P(“d” | “To”)

× …

× P(<eos>* | “Today is a sunny day!”)

Language modelling and tokenisation

Probabilities over character-strings

Probabilities over subword-strings

P(“Today is a sunny day!”)

= P(“T”)

× P(“o” | “T”)

× P(“d” | “To”)

× …

× P(<eos>* | “Today is a sunny day!”)

*required to define probability distributions autoregressively (Du et al., 2023)

Language modelling and tokenisation

Probabilities over character-strings

$$P(\text{“Today is a sunny day!”}) = P(\text{“T”}) \times P(\text{“o”} \mid \text{“T”}) \times P(\text{“d”} \mid \text{“To”}) \times \ldots \times P(\langle \text{eos} \rangle^* \mid \text{“Today is a sunny day!”})$$

Probabilities over subword-strings

$$P_{\mathbb{T}}([\,\text{\_To}, \text{day}, \text{\_is}, \text{\_a}, \text{\_sunny}, \text{\_day}, !\,]) = P_{\mathbb{T}}(\text{\_To}) \times P_{\mathbb{T}}(\text{day} \mid \text{\_To}) \times P_{\mathbb{T}}(\text{\_is} \mid \text{\_To}, \text{day}) \times \ldots \times P_{\mathbb{T}}(\langle \text{eos} \rangle \mid \text{\_To}, \text{day}, \text{\_is}, \text{\_a}, \text{\_sunny}, \text{\_day})$$

*required to define probability distributions autoregressively (Du et al., 2023)

Language modelling and tokenisation

Probabilities over character-strings

$$P(\text{“Today is a sunny day!”}) = P(\text{“T”}) \times P(\text{“o”} \mid \text{“T”}) \times P(\text{“d”} \mid \text{“To”}) \times \ldots \times P(\langle \text{eos} \rangle^* \mid \text{“Today is a sunny day!”})$$

Probabilities over subword-strings

$$P_{\mathbb{T}}([\,\text{\_To}, \text{day}, \text{\_is}, \text{\_a}, \text{\_sunny}, \text{\_day}, !\,]) = P_{\mathbb{T}}(\text{\_To}) \times P_{\mathbb{T}}(\text{day} \mid \text{\_To}) \times P_{\mathbb{T}}(\text{\_is} \mid \text{\_To}, \text{day}) \times \ldots \times P_{\mathbb{T}}(\langle \text{eos} \rangle \mid \text{\_To}, \text{day}, \text{\_is}, \text{\_a}, \text{\_sunny}, \text{\_day})$$

Tokenisation is just a way to factor a probability over a string into smaller, conditional terms

*required to define probability distributions autoregressively (Du et al., 2023)

Language modelling and tokenisation

Probabilities over character-strings = **Probabilities over subword-strings**

$$P(\text{“Today is a sunny day!”}) = P(\text{“T”}) \times P(\text{“o”} \mid \text{“T”}) \times P(\text{“d”} \mid \text{“To”}) \times \ldots \times P(\langle \text{eos} \rangle^* \mid \text{“Today is a sunny day!”})$$

$$P_{\mathbb{T}}([\,\text{\_To}, \text{day}, \text{\_is}, \text{\_a}, \text{\_sunny}, \text{\_day}, !\,]) = P_{\mathbb{T}}(\text{\_To}) \times P_{\mathbb{T}}(\text{day} \mid \text{\_To}) \times P_{\mathbb{T}}(\text{\_is} \mid \text{\_To}, \text{day}) \times \ldots \times P_{\mathbb{T}}(\langle \text{eos} \rangle \mid \text{\_To}, \text{day}, \text{\_is}, \text{\_a}, \text{\_sunny}, \text{\_day})$$

Tokenisation is just a way to factor a probability over a string into smaller, conditional terms

*required to define probability distributions autoregressively (Du et al., 2023)

So, is the choice of the tokeniser important?

P(“Today is a sunny day!”)

So, is the choice of the tokeniser important?

$\mathcal{V}$

ID	Subwords	Merges
32	_	
33	!	
84	T	
97	a	
100	d	Initial Alphabet
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

P("Today is a sunny day!")

$=$

$P_{\mathbb{T}}$([_To, day, _is, _a, **_sunny**, _day, !])

So, is the choice of the tokeniser important?

$\mathcal{V}$

ID	Subwords	Merges
32	_	Initial Alphabet
33	!	
84	T	
97	a	
100	d	
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

P("Today is a sunny day!")

=

$P_{\mathbb{T}}$([_To, day, _is, _a, **_sunny**, _day, !])

=

$P_{\mathbb{T}^*}$([_To, day, _is, _a, **_su**, **nny**, _day, !])

$\mathcal{V}^*$

ID	Subwords	Merges
32	_	Initial Alphabet
33	!	
84	T	
97	a	
100	d	
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
~~261~~	~~_sunny~~	~~258+260~~

So, is the choice of the tokeniser important?

$\mathcal{V}$

ID	Subwords	Merges
32	_	Initial Alphabet
33	!	
84	T	
97	a	
100	d	
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

In principle, NO

Tokenisation is simply a way to construct conditional distributions

P("Today is a sunny day!")

$=$ $=$

$P_{\mathbb{T}}$([_To, day, _is, _a, **_sunny**, _day, !]) $=$ $P_{\mathbb{T}^*}$([_To, day, _is, _a, **_su**, **nny**, _day, !])

$\mathcal{V}^*$

ID	Subwords	Merges
32	_	Initial Alphabet
33	!	
84	T	
97	a	
100	d	
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
~~261~~	~~_sunny~~	~~258+260~~

So, is the choice of the tokeniser important?

$\mathcal{V}$

ID	Subwords	Merges
32	_	Initial Alphabet
33	!	
84	T	
97	a	
100	d	
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

In principle, NO

Tokenisation is simply a way to construct conditional distributions

P("Today is a sunny day!")

$=$

$P_{\mathbb{T}}$([_To, day, _is, _a, **_sunny**, _day, !]) $=$ $P_{\mathbb{T}^*}$([_To, day, _is, _a, **_su**, **nny**, _day, !])

$\approx$ $\approx$

P_{LM}([_To, day, _is, _a, _sunny, _day, !]) P_{LM}([_To, day, _is, _a, _su, nny, _day, !])

$\mathcal{V}^*$

ID	Subwords	Merges
32	_	Initial Alphabet
33	!	
84	T	
97	a	
100	d	
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

So, is the choice of the tokeniser important?

$\mathcal{V}$

ID	Subwords	Merges
32	_	Initial Alphabet
33	!	
84	T	
97	a	
100	d	
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

In principle, NO
Tokenisation is simply a way to construct conditional distributions

In practice, YES
Because of inductive biases and imperfect models

P("Today is a sunny day!")

$=$ $=$

$P_{\mathbb{T}}$([_To, day, _is, _a, **_sunny**, _day, !]) $=$ $P_{\mathbb{T}^*}$([_To, day, _is, _a, **_su**, **nny**, _day, !])

$\approx$ $\approx$

P_{LM}([_To, day, _is, _a, _sunny, _day, !]) $\neq$ P_{LM}([_To, day, _is, _a, _su, nny, _day, !])

$\mathcal{V}^*$

ID	Subwords	Merges
32	_	Initial Alphabet
33	!	
84	T	
97	a	
100	d	
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
~~261~~	~~_sunny~~	~~258+260~~

So, is the choice of the tokeniser important?

$\mathcal{V}$

ID	Subwords	Merges
32	_	Initial Alphabet
33	!	
84	T	
97	a	
100	d	
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

In principle, NO
Tokenisation is simply a way to construct conditional distributions

In practice, YES
Because of inductive biases and imperfect models

P("Today is a sunny day!")

$=$ $=$

$P_{\mathbb{T}}$([_To, day, _is, _a, **_sunny**, _day, !]) $=$ $P_{\mathbb{T}^*}$([_To, day, _is, _a, **_su**, **nny**, _day, !])

$\approx$ $\approx$

P_{LM}([_To, day, _is, _a, _sunny, _day, !]) $\neq$ P_{LM}([_To, day, _is, _a, _su, nny, _day, !])

$\mathcal{V}^*$

ID	Subwords	Merges
32	_	Initial Alphabet
33	!	
84	T	
97	a	
100	d	
110	n	
115	s	
117	u	
121	y	
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

The difference in the probability a model assigns to a string because of tokenisation is what we call **tokenisation bias**

Tokenisation bias

In general, we define tokenisation bias as follows:

Definition 1 (General). The tokenisation bias induced by a **property** of the tokeniser is its effect on a model's ability to predict some character-string $\mathbf{c} = "c_1c_2...c_{|\mathbf{c}|}"$.

Tokenisation bias

In general, we define tokenisation bias as follows:

Definition 1 (General). The tokenisation bias induced by a **property** of the tokeniser is its effect on a model's ability to predict some character-string $\mathbf{c} = c_1c_2...c_{|\mathbf{c}|}$.

The property we focus on in our paper is the **inclusion of a subword in its vocabulary**—i.e., $v \in \mathcal{V}$—and how it affects predictions on its characters.

Tokenisation bias

In general, we define tokenisation bias as follows:

Definition 1 (General). The tokenisation bias induced by a **property** of the tokeniser is its effect on a model's ability to predict some character-string $\mathbf{c} = \text{“}c_1c_2\ldots c_{|\mathbf{c}|}\text{”}$.

The property we focus on in our paper is the **inclusion of a subword in its vocabulary**—i.e., $v \in \mathcal{V}$—and how it affects predictions on its characters.

Definition 2 (Our paper). The tokenisation bias of $v \in \mathcal{V}$ is its effect on a model's ability to predict $\mathbf{c}_v$.

Tokenisation bias

In general, we define tokenisation bias as follows:

Definition 1 (General). The tokenisation bias induced by a **property** of the tokeniser is its effect on a model's ability to predict some character-string $\mathbf{c} = \text{“}c_1c_2...c_{|\mathbf{c}|}\text{”}$.

The property we focus on in our paper is the **inclusion of a subword in its vocabulary**—i.e., $v \in \mathcal{V}$—and how it affects predictions on its characters.

Definition 2 (Our paper). The tokenisation bias of $v \in \mathcal{V}$ is its effect on a model's ability to predict $\mathbf{c}_v$.

For example, consider the subword _sunny. We are interested in estimating how the probability of its characters, “_sunny” changes if its associated subword is removed from $\mathcal{V}$, that is:

$\mathcal{V}$

ID	Subwords	Merges
...	...	...
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

τ(“_sunny”) = [_sunny]

vs.

$\mathcal{V}^*$

ID	Subwords	Merges
...	...	...
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
~~261~~	~~_sunny~~	~~258+260~~

τ^*(“_sunny”) = [_su, _nny]

Tokenisation bias

In general, we define tokenisation bias as follows:

Definition 1 (General). The tokenisation bias induced by a **property** of the tokeniser is its effect on a model's ability to predict some character-string $\mathbf{c} = \text{“}c_1c_2...c_{|\mathbf{c}|}\text{”}$.

The property we focus on in our paper is the **inclusion of a subword in its vocabulary**—i.e., $v \in \mathcal{V}$—and how it affects predictions on its characters.

Definition 2 (Our paper). The tokenisation bias of $v \in \mathcal{V}$ is its effect on a model's ability to predict $\mathbf{c}_v$.

For example, consider the subword _sunny. We are interested in estimating how the probability of its characters, “_sunny” changes if its associated subword is removed from $\mathcal{V}$, that is:

$\mathcal{V}$

ID	Subwords	Merges
...	...	...
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

$\boldsymbol{\tau}$(“_sunny”) = [_sunny]

vs.

$\mathcal{V}^*$

ID	Subwords	Merges
...	...	...
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
~~261~~	~~_sunny~~	~~258+260~~

$\boldsymbol{\tau}^*$(“_sunny”) = [_su, _nny]

$$\psi_v = P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \in \mathcal{V}) - P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \notin \mathcal{V})$$

Estimating tokenisation bias is hard

$$\psi_v = P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \in \mathcal{V}) - P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \notin \mathcal{V})$$

Estimating tokenisation bias is hard

$$\psi_v = \underbrace{P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \in \mathcal{V})}_{\text{Observable}} - \underbrace{P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \notin \mathcal{V})}_{\text{Counterfactual}}$$

Estimating tokenisation bias is hard

$$\psi_v = \underbrace{P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \in \mathcal{V})}_{\text{Observable}} - \underbrace{P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \notin \mathcal{V})}_{\text{Counterfactual}}$$

Given that we train a model using $\mathcal{V}$, how can we estimate the probability under a **counterfactual** vocabulary?

Estimating tokenisation bias is hard

$$\psi_v = \underbrace{P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \in \mathcal{V})}_{\text{Observable}} - \underbrace{P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \notin \mathcal{V})}_{\text{Counterfactual}}$$

Given that we train a model using $\mathcal{V}$, how can we estimate the probability under a **counterfactual** vocabulary?

1. Tokenise differently using the same $\mathcal{V}$

$P_{LM}(\text{\_su} \mid \text{context}) \times P_{LM}(\text{nny} \mid \text{context}, \text{\_su})$

$\mathcal{V}$

ID	Subwords	Merges
...	...	...
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

Estimating tokenisation bias is hard

$$\psi_v = \underbrace{P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \in \mathcal{V})}_{\text{Observable}} - \underbrace{P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \notin \mathcal{V})}_{\text{Counterfactual}}$$

Given that we train a model using $\mathcal{V}$, how can we estimate the probability under a **counterfactual** vocabulary?

✗ 1. Tokenise differently using the same $\mathcal{V}$

$P_{LM}(\text{\_su} \mid \text{context}) \times P_{LM}(\text{nny} \mid \text{context}, \text{\_su})$

$\mathcal{V}$

ID	Subwords	Merges
…	…	…
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

Estimating tokenisation bias is hard

$$\psi_v = \underbrace{P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \in \mathcal{V})}_{\text{Observable}} - \underbrace{P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \notin \mathcal{V})}_{\text{Counterfactual}}$$

Given that we train a model using $\mathcal{V}$, how can we estimate the probability under a **counterfactual** vocabulary?

1. Tokenise differently using the same $\mathcal{V}$

$P_{LM}(\text{\_su} \mid \text{context}) \times P_{LM}(\text{nny} \mid \text{context}, \text{\_su})$

$\mathcal{V}$

ID	Subwords	Merges
...	...	...
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

2. Compare to OOV words probabilities, e.g., “_appoggiatura”

$P_{LM}(\tau(\text{“\_appoggiatura”}) \mid \text{context})$

$= P_{LM}(\text{\_appoggia} \mid \text{context})$
$\times P_{LM}(\text{tura} \mid \text{context}, \text{\_appoggia})$

Estimating tokenisation bias is hard

$$\psi_v = \underbrace{P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \in \mathcal{V})}_{\text{Observable}} - \underbrace{P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \notin \mathcal{V})}_{\text{Counterfactual}}$$

Given that we train a model using $\mathcal{V}$, how can we estimate the probability under a **counterfactual** vocabulary?

✗ 1. Tokenise differently using the same $\mathcal{V}$

✗ 2. Compare to OOV words probabilities, e.g., “_appoggiatura”

$P_{LM}(\text{\_su} \mid \text{context}) \times P_{LM}(\text{nny} \mid \text{context}, \text{\_su})$

$\mathcal{V}$

ID	Subwords	Merges
...	...	...
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

$P_{LM}(\tau(\text{“\_appoggiatura”}) \mid \text{context})$

$= P_{LM}(\text{\_appoggia} \mid \text{context})$
$\times P_{LM}(\text{tura} \mid \text{context}, \text{\_appoggia})$

Estimating tokenisation bias is hard

$$\psi_v = \underbrace{P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \in \mathcal{V})}_{\text{Observable}} - \underbrace{P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \notin \mathcal{V})}_{\text{Counterfactual}}$$

Given that we train a model using $\mathcal{V}$, how can we estimate the probability under a **counterfactual** vocabulary?

1. Tokenise differently using the same $\mathcal{V}$

$P_{LM}(\text{\_su} \mid \text{context}) \times P_{LM}(\text{nny} \mid \text{context}, \text{\_su})$

$\mathcal{V}$

ID	Subwords	Merges
...	...	...
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

2. Compare to OOV words probabilities, e.g., “_appoggiatura”

$P_{LM}(\tau(\text{“\_appoggiatura”}) \mid \text{context})$
$= P_{LM}(\text{\_appoggia} \mid \text{context})$
$\times P_{LM}(\text{tura} \mid \text{context}, \text{\_appoggia})$

3. Retrain the model with a different $\mathcal{V}^*$

$P_{LM^*}(\text{\_su} \mid \text{context}) \times P_{LM^*}(\text{nny} \mid \text{context}, \text{\_su})$

$\mathcal{V}^*$

ID	Subwords	Merges
...	...	...
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
~~261~~	~~_sunny~~	~~258+260~~

Estimating tokenisation bias is hard

$$\psi_v = \underbrace{P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \in \mathcal{V})}_{\text{Observable}} - \underbrace{P_{LM}(\text{“\_sunny”} \mid \text{context}, \text{\_sunny} \notin \mathcal{V})}_{\text{Counterfactual}}$$

Given that we train a model using $\mathcal{V}$, how can we estimate the probability under a **counterfactual** vocabulary?

1. Tokenise differently using the same $\mathcal{V}$

$P_{LM}(\text{\_su} \mid \text{context}) \times P_{LM}(\text{nny} \mid \text{context}, \text{\_su})$

$\mathcal{V}$

ID	Subwords	Merges
...	...	...
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
261	_sunny	258+260

2. Compare to OOV words probabilities, e.g., “_appoggiatura”

$P_{LM}(\tau(\text{“\_appoggiatura”}) \mid \text{context})$
$= P_{LM}(\text{\_appoggia} \mid \text{context})$
$\times P_{LM}(\text{tura} \mid \text{context}, \text{\_appoggia})$

3. Retrain the model with a different $\mathcal{V}^*$

$P_{LM^*}(\text{\_su} \mid \text{context}) \times P_{LM^*}(\text{nny} \mid \text{context}, \text{\_su})$

$\mathcal{V}^*$

ID	Subwords	Merges
...	...	...
256	_a	32+97
257	_s	32+115
258	_su	257+117
259	nn	110+110
260	nny	259+121
~~261~~	~~_sunny~~	~~258+260~~

Tokenisation bias as a causal effect

Tokenisation bias as a causal effect

Bottom-up tokenisers—including BPE (Sennrich et al., 2016) and WordPiece (Schuster and Nakajima, 2012)—select subwords iteratively, one-at-a-time, to compose their vocabularies. We can exploit this fact!

Tokenisation bias as a causal effect

Bottom-up tokenisers—including BPE (Sennrich et al., 2016) and WordPiece (Schuster and Nakajima, 2012)—select subwords iteratively, one-at-a-time, to compose their vocabularies. We can exploit this fact!

v

ID	Subwords
...	...
258	_su
259	nn
260	nny
261	_sunny
262	_open
263	_day

Tokenisation bias as a causal effect

Bottom-up tokenisers—including BPE (Sennrich et al., 2016) and WordPiece (Schuster and Nakajima, 2012)—select subwords iteratively, one-at-a-time, to compose their vocabularies. We can exploit this fact!

v

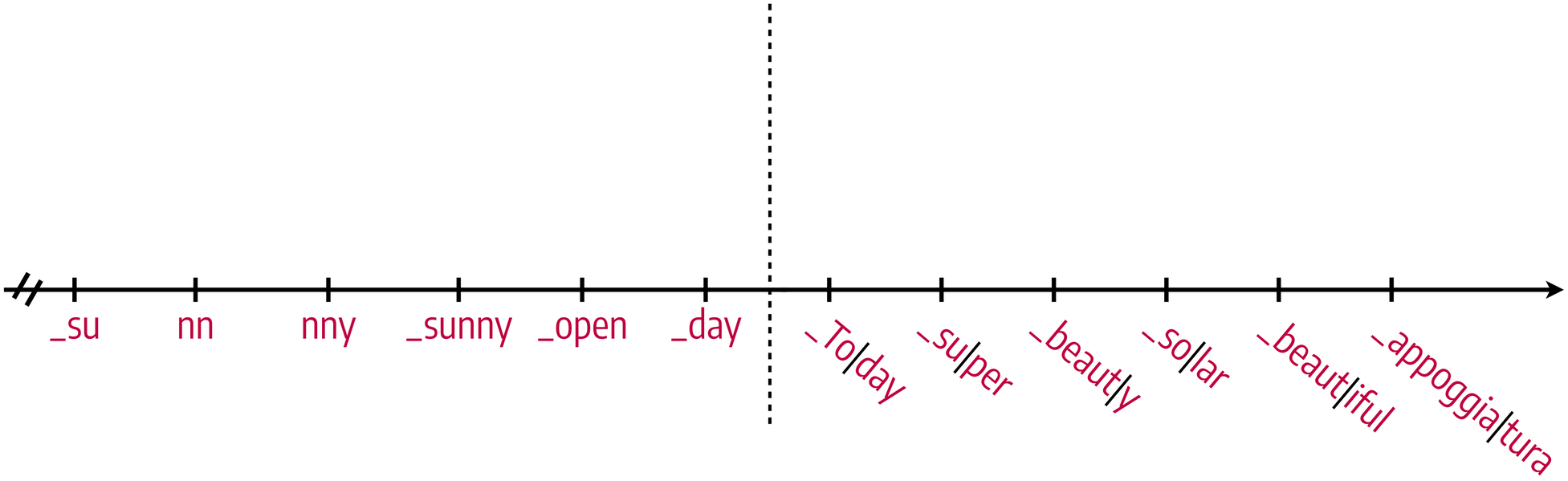
ID	Subwords
...	...
258	_su
259	nn
260	nny
261	_sunny
262	_open
263	_day

Tokenisation bias as a causal effect

Bottom-up tokenisers—including BPE (Sennrich et al., 2016) and WordPiece (Schuster and Nakajima, 2012)—select subwords iteratively, one-at-a-time, to compose their vocabularies. We can exploit this fact!

v

ID	Subwords
...	...
258	_su
259	nn
260	nny
261	_sunny
262	_open
263	_day

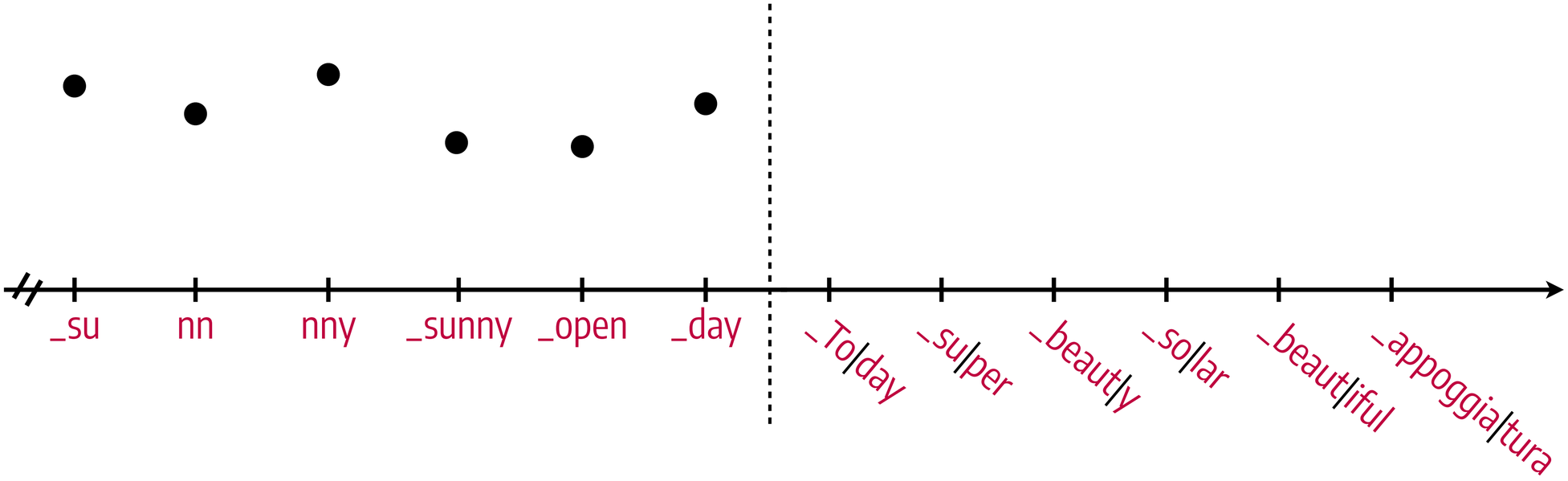


Tokenisation bias as a causal effect

Bottom-up tokenisers—including BPE (Sennrich et al., 2016) and WordPiece (Schuster and Nakajima, 2012)—select subwords iteratively, one-at-a-time, to compose their vocabularies. We can exploit this fact!

v

ID	Subwords
...	...
258	_su
259	nn
260	nny
261	_sunny
262	_open
263	_day

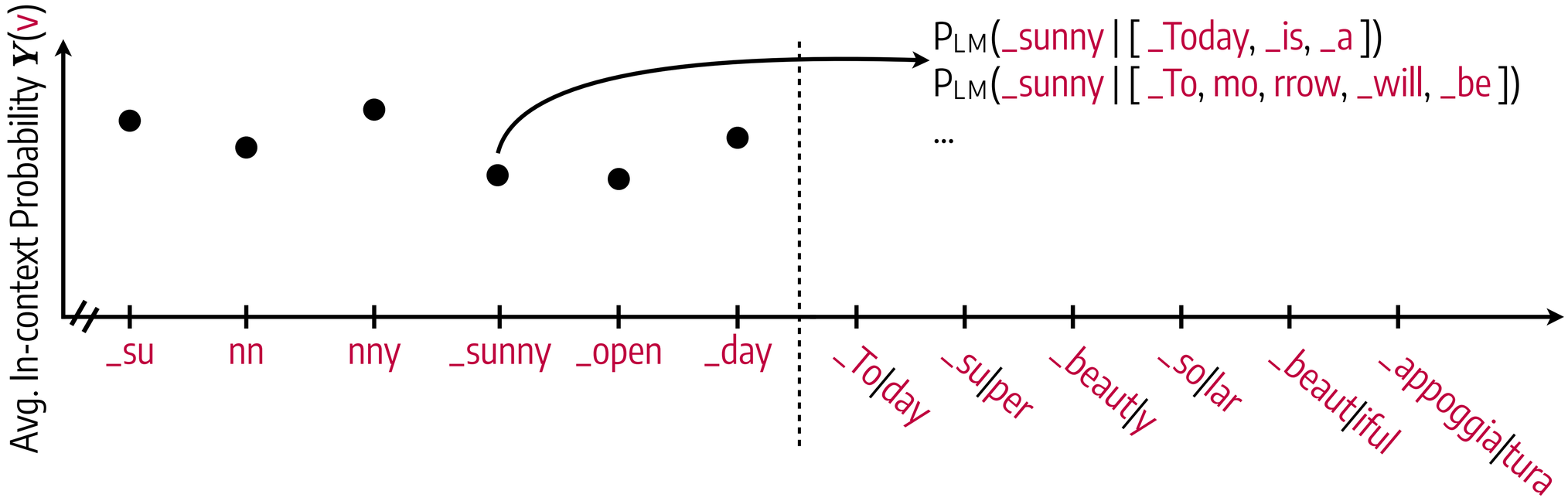


Tokenisation bias as a causal effect

Bottom-up tokenisers—including BPE (Sennrich et al., 2016) and WordPiece (Schuster and Nakajima, 2012)—select subwords iteratively, one-at-a-time, to compose their vocabularies. We can exploit this fact!

$\mathcal{V}$

ID	Subwords
...	...
258	_su
259	nn
260	nny
261	_sunny
262	_open
263	_day

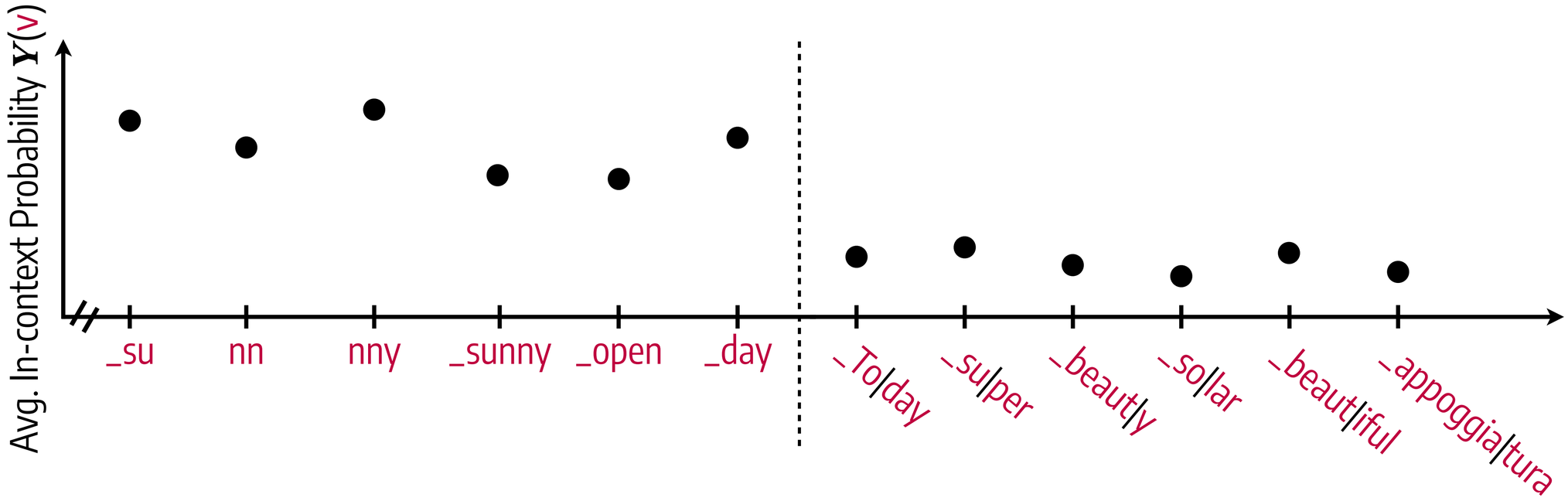


Tokenisation bias as a causal effect

Bottom-up tokenisers—including BPE (Sennrich et al., 2016) and WordPiece (Schuster and Nakajima, 2012)—select subwords iteratively, one-at-a-time, to compose their vocabularies. We can exploit this fact!

v

ID	Subwords
...	...
258	_su
259	nn
260	nny
261	_sunny
262	_open
263	_day

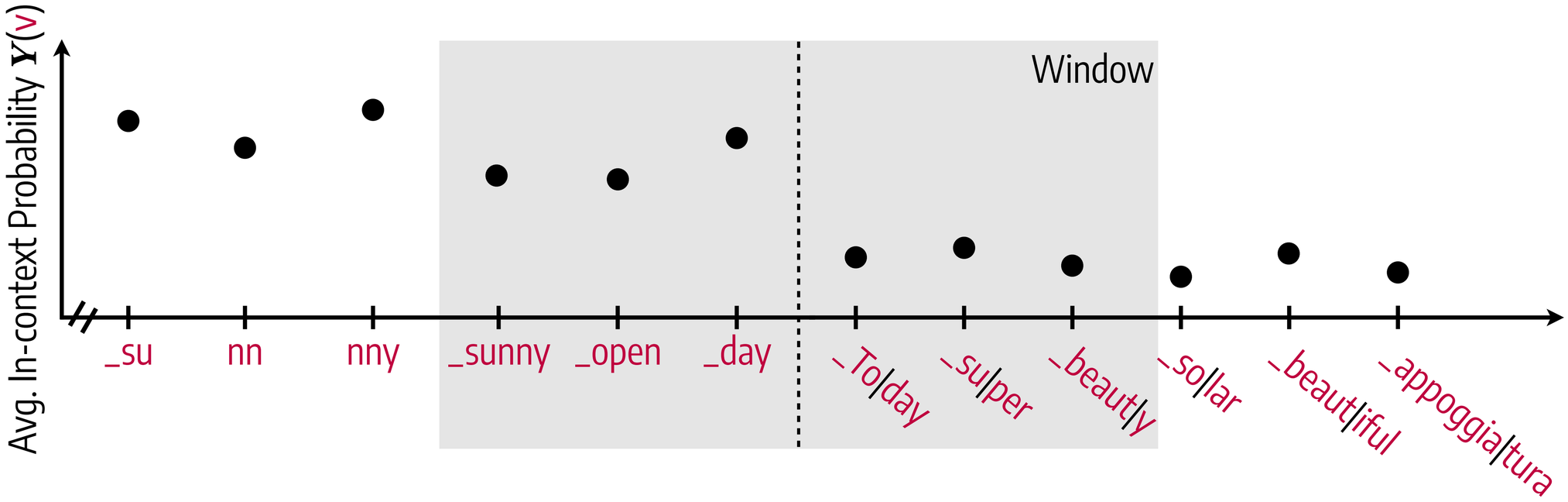


Tokenisation bias as a causal effect

Bottom-up tokenisers—including BPE (Sennrich et al., 2016) and WordPiece (Schuster and Nakajima, 2012)—select subwords iteratively, one-at-a-time, to compose their vocabularies. We can exploit this fact!

$\mathcal{V}$

ID	Subwords
...	...
258	_su
259	nn
260	nny
261	_sunny
262	_open
263	_day



Subwords in the window have **similar frequency** but on one side appear as one vs. two subwords in $\mathcal{V}$.

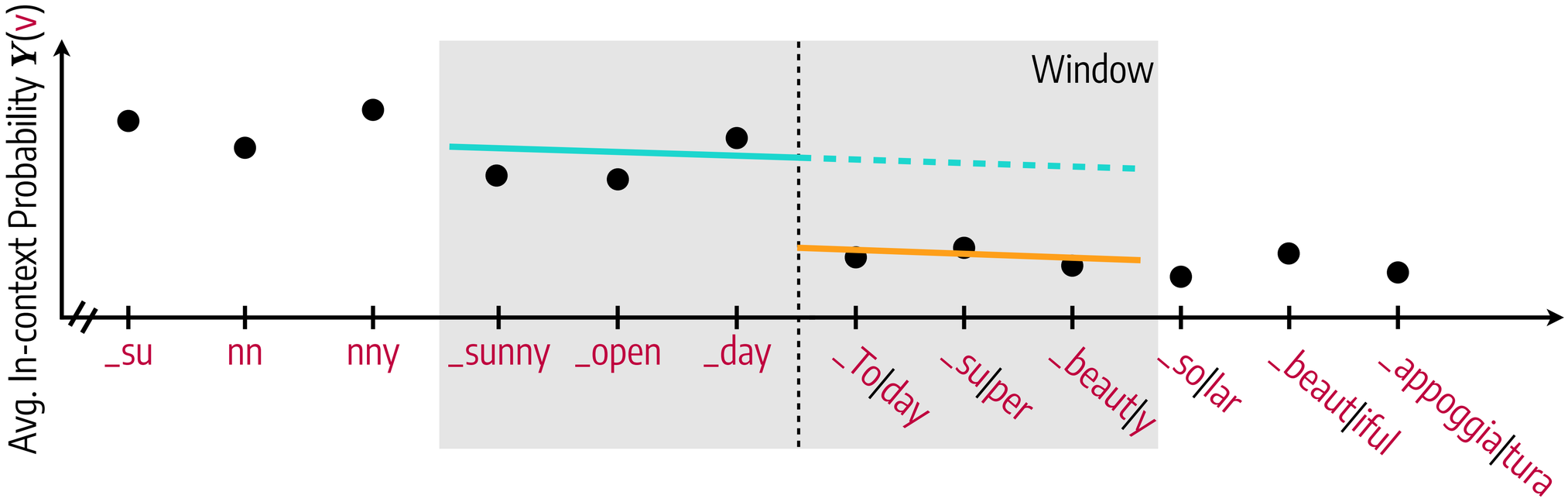
Thus, by comparing their average in-context probability we can estimate tokenisation bias without re-training

Tokenisation bias as a causal effect

Bottom-up tokenisers—including BPE (Sennrich et al., 2016) and WordPiece (Schuster and Nakajima, 2012)—select subwords iteratively, one-at-a-time, to compose their vocabularies. We can exploit this fact!

$\mathcal{V}$

ID	Subwords
...	...
258	_su
259	nn
260	nny
261	_sunny
262	_open
263	_day



Subwords in the window have **similar frequency** but on one side appear as one vs. two subwords in $\mathcal{V}$.

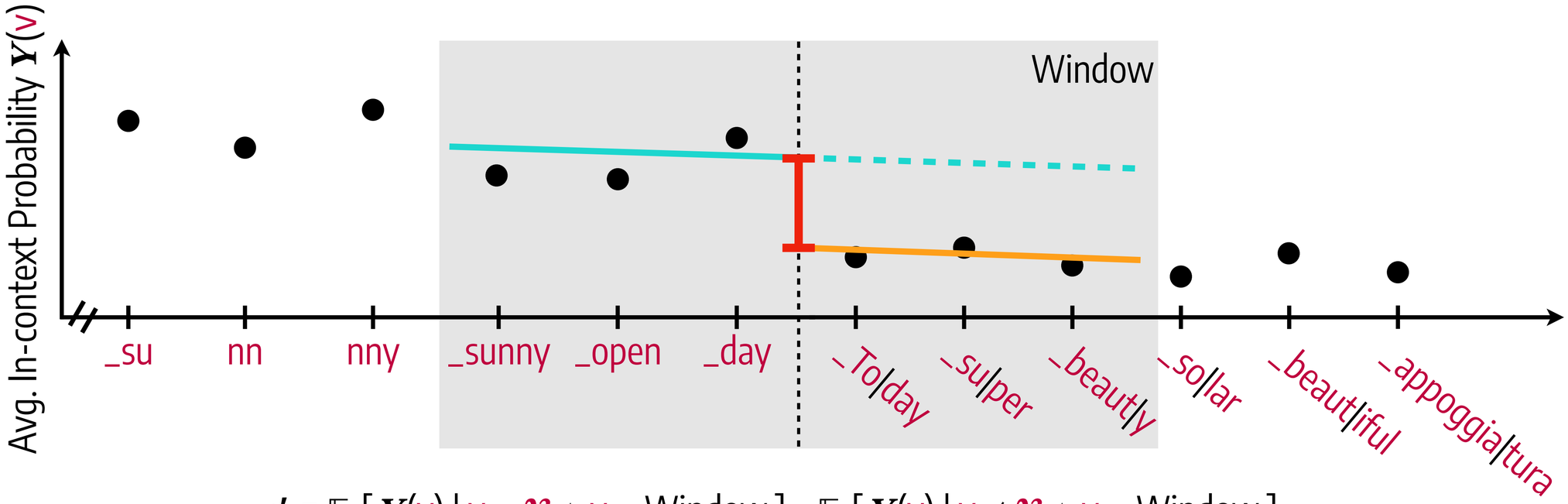
Thus, by comparing their average in-context probability we can estimate tokenisation bias without re-training

Tokenisation bias as a causal effect

Bottom-up tokenisers—including BPE (Sennrich et al., 2016) and WordPiece (Schuster and Nakajima, 2012)—select subwords iteratively, one-at-a-time, to compose their vocabularies. We can exploit this fact!

$\mathcal{V}$

ID	Subwords
...	...
258	_su
259	nn
260	nny
261	_sunny
262	_open
263	_day



$$\psi = \mathbb{E}_v[\, Y(v) \mid v \in \mathcal{V} \wedge v \in \text{Window} \,] - \mathbb{E}_v[\, Y(v) \mid v \notin \mathcal{V} \wedge v \in \text{Window} \,]$$

Subwords in the window have **similar frequency** but on one side appear as one vs. two subwords in $\mathcal{V}$.

Thus, by comparing their average in-context probability we can estimate tokenisation bias without re-training

Experimental Setup

Experimental Setup

Tokeniser training. Our method requires knowing which subwords would be added after the cutoff. Thus, we train* BPE with a large vocabulary of 320k subwords and derive smaller vocabularies.

*This information is typically not available even for open-source models and tokeniser construction details are often unreproducible

Experimental Setup

Tokeniser training. Our method requires knowing which subwords would be added after the cutoff. Thus, we train* BPE with a large vocabulary of 320k subwords and derive smaller vocabularies.

Model training. Llama-style 57M-param (non-embedding) model trained on the MiniPile for 50k steps (~13B tokens) using BPE-32k.

*This information is typically not available even for open-source models and tokeniser construction details are often unreproducible

Experimental Setup

Tokeniser training. Our method requires knowing which subwords would be added after the cutoff. Thus, we train* BPE with a large vocabulary of 320k subwords and derive smaller vocabularies.

Model training. Llama-style 57M-param (non-embedding) model trained on the MiniPile for 50k steps (~13B tokens) using BPE-32k.

Model Evaluations and RDD setup. Window size 5k around the cutoff (i.e, 27k-37k). Collect log-probabilities for these subwords on the MiniPile validation set. We run a simple linear regression and validate more complex models.

*This information is typically not available even for open-source models and tokeniser construction details are often unreproducible

Experimental Setup

Tokeniser training. Our method requires knowing which subwords would be added after the cutoff. Thus, we train* BPE with a large vocabulary of 320k subwords and derive smaller vocabularies.

Model training. Llama-style 57M-param (non-embedding) model trained on the MiniPile for 50k steps (~13B tokens) using BPE-32k.

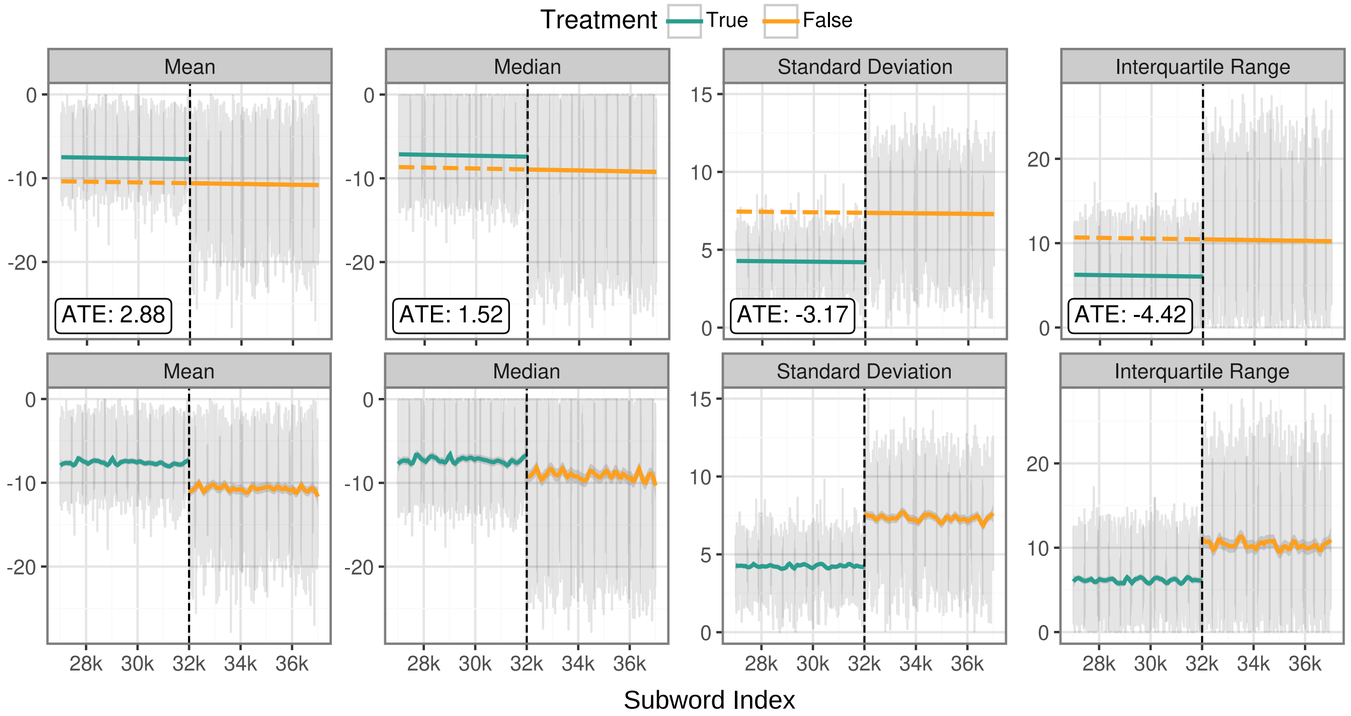
Model Evaluations and RDD setup. Window size 5k around the cutoff (i.e, 27k-37k). Collect log-probabilities for these subwords on the MiniPile validation set. We run a simple linear regression and validate more complex models.

We also study how tokenisation bias changes as a function of:

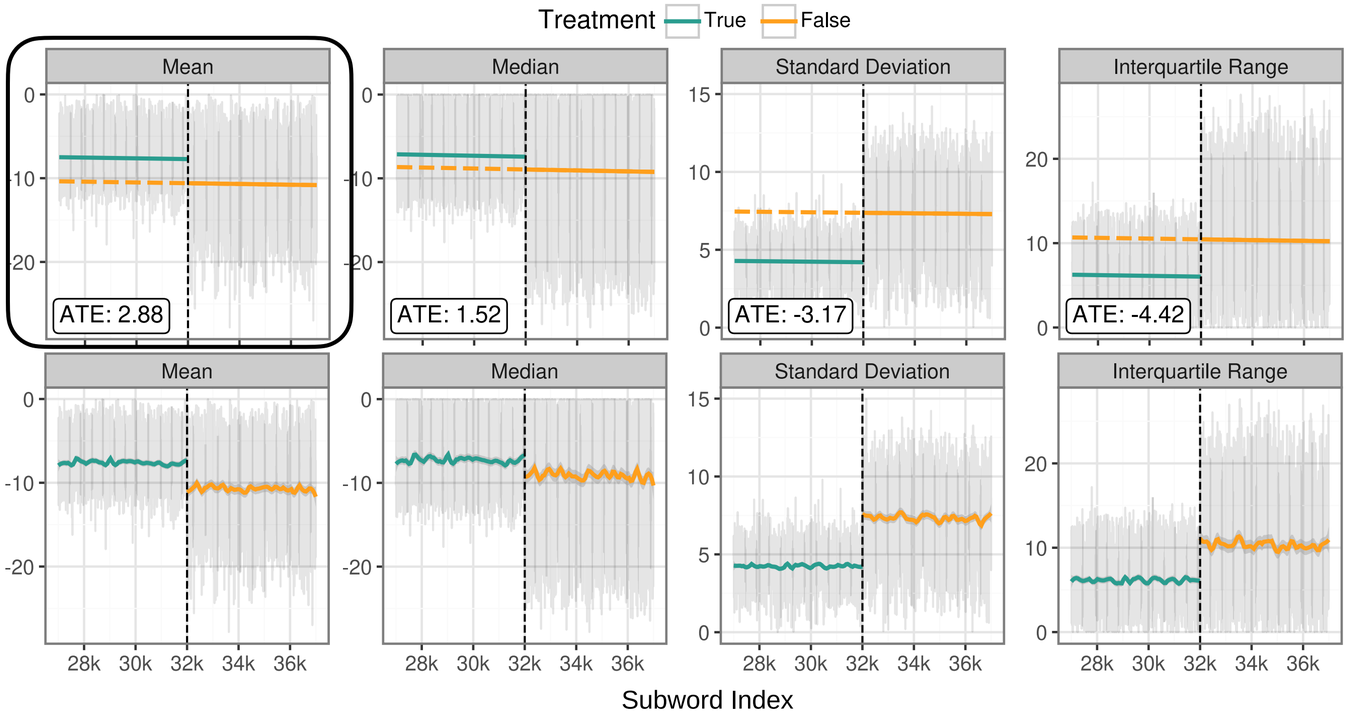
1. **Vocab sizes:** 8k, 32k, 128k (BPE)
2. **Tokenisation:** WordPiece (WP) & hybrid BPE2WP (BPE vocabulary + WP tokenisation function)
3. **Model sizes:** 340M & 850M
4. **Embedding tying & Dataset:** 100M model trained with/without tied embeddings on 20B tokens from Fineweb-Edu
5. **Aggregation function:** Mean, median, standard deviation, and interquartile range of the subword probability across contexts

*This information is typically not available even for open-source models and tokeniser construction details are often unreproducible

Tokenisation bias at the end of training

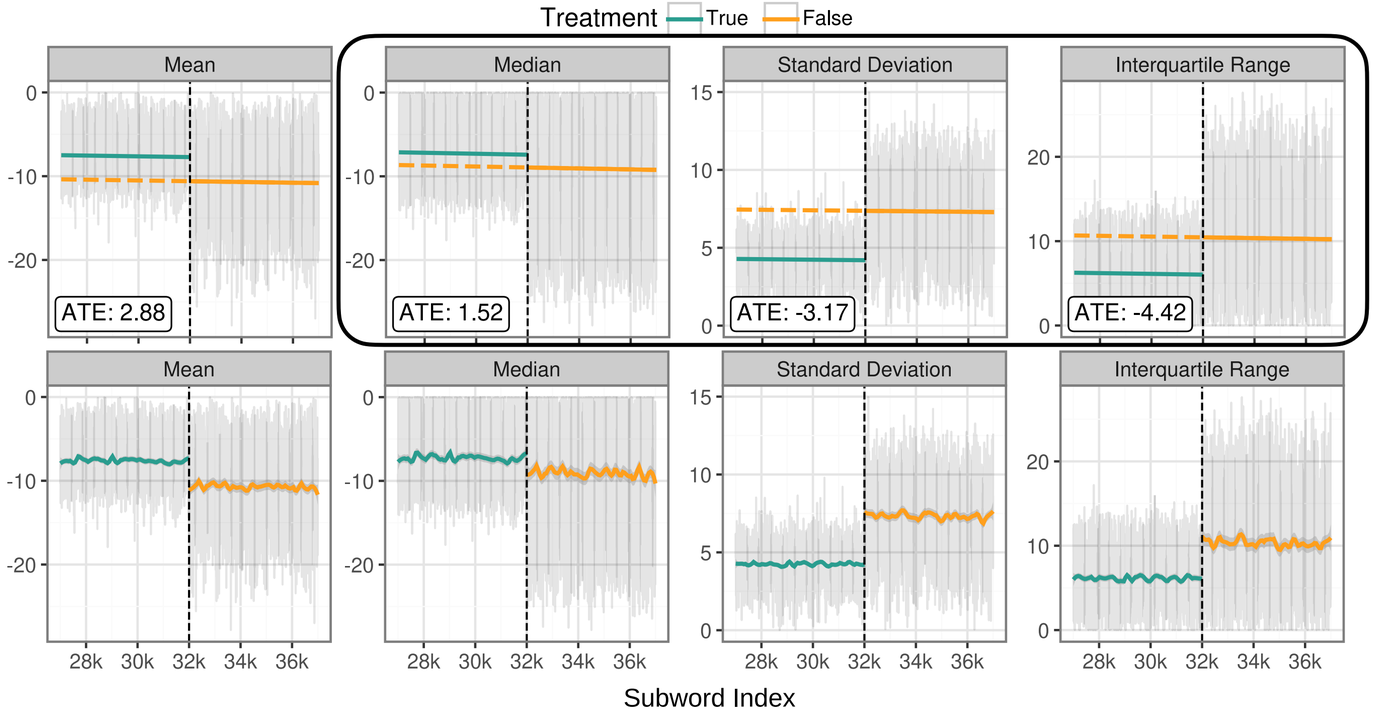


Tokenisation bias at the end of training



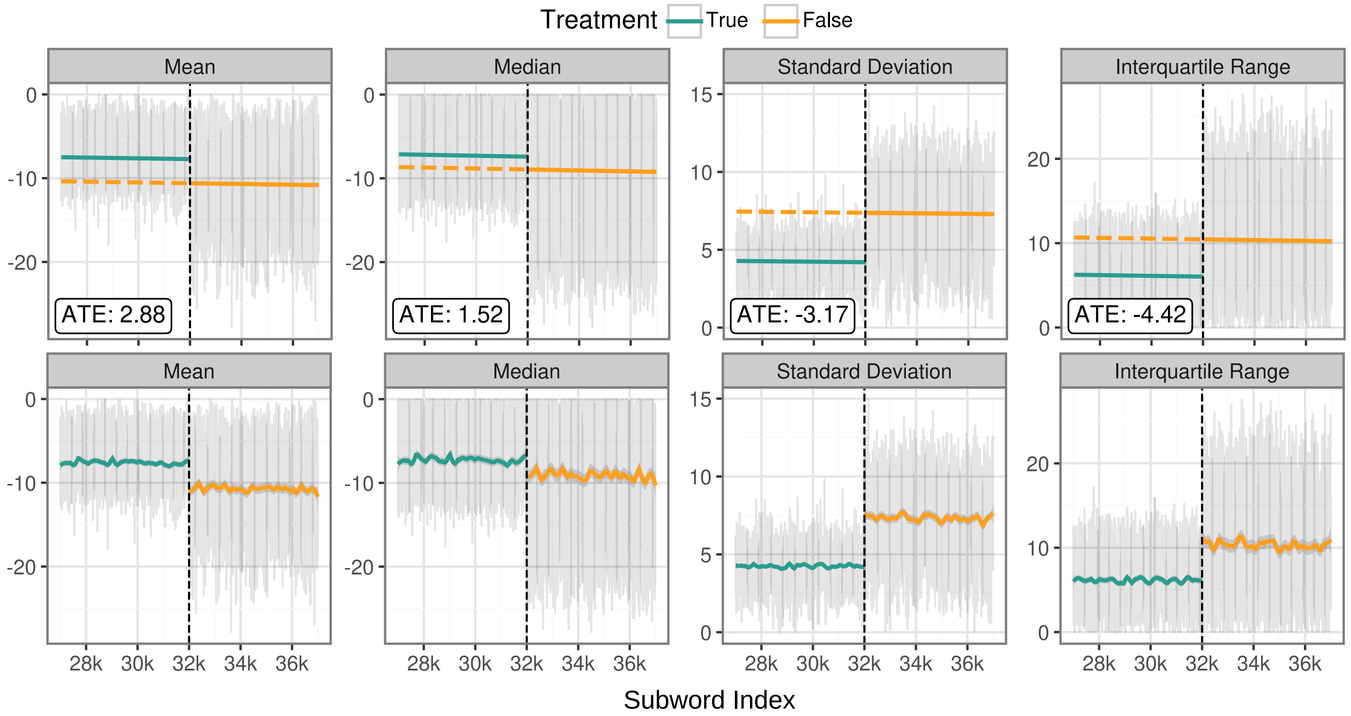
- We estimate this bias to be 2.88 nats. This means a can be assigned roughly 17x less probability due solely to tokenisation

Tokenisation bias at the end of training



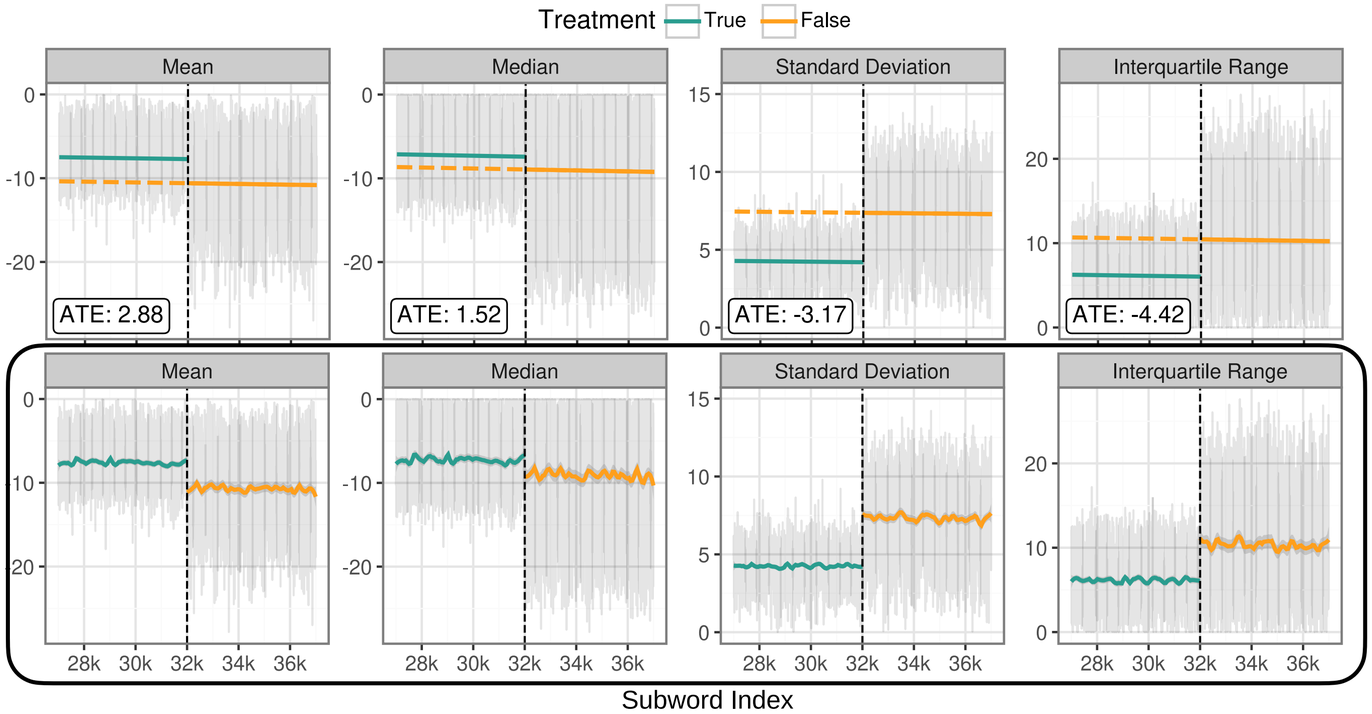
- We estimate this bias to be 2.88 nats. This means a can be assigned roughly 17x less probability due solely to tokenisation
- Subword inclusion significantly affects output stability: in-vocabulary subwords exhibit far less variation in log-probability across contexts than out-of-vocabulary ones

Tokenisation bias at the end of training



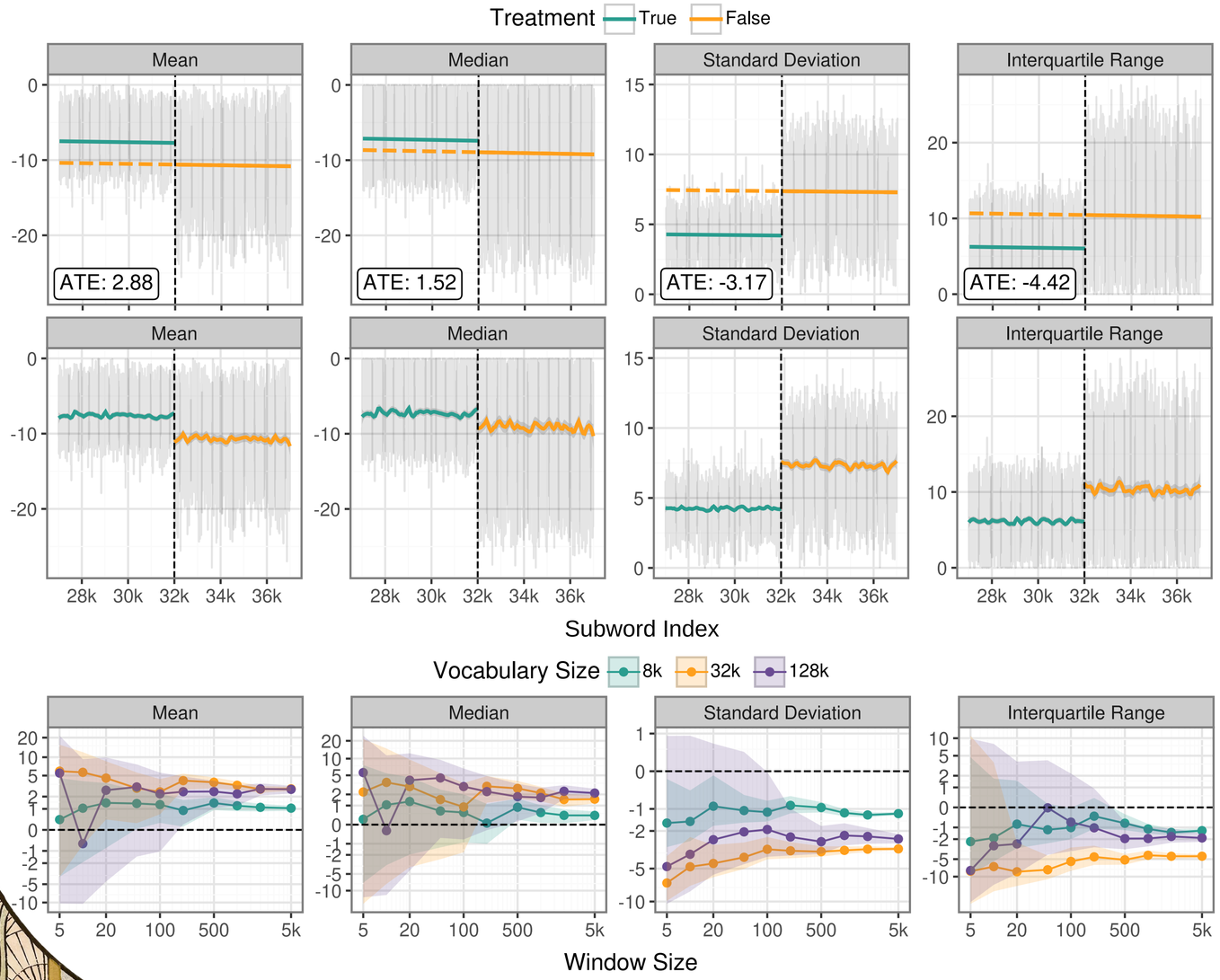
- We estimate this bias to be 2.88 nats. This means a can be assigned roughly 17x less probability due solely to tokenisation
- Subword inclusion significantly affects output stability: in-vocabulary subwords exhibit far less variation in log-probability across contexts than out-of-vocabulary ones
- RDD estimates are robust to

Tokenisation bias at the end of training



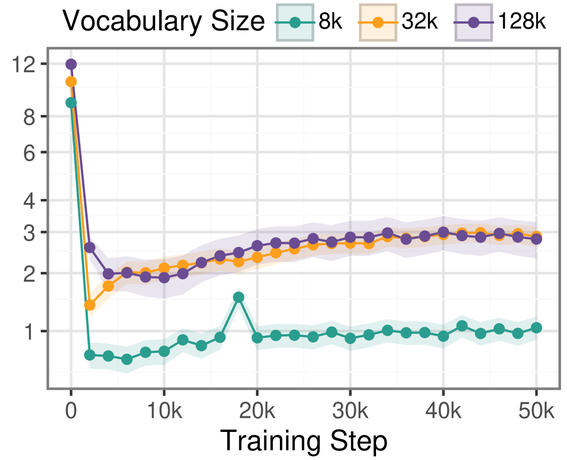
- We estimate this bias to be 2.88 nats. This means a can be assigned roughly 17x less probability due solely to tokenisation
- Subword inclusion significantly affects output stability: in-vocabulary subwords exhibit far less variation in log-probability across contexts than out-of-vocabulary ones
- RDD estimates are robust to
  - Functional form

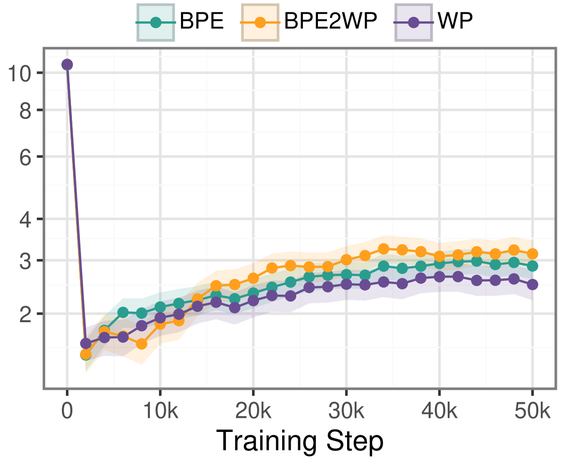
Tokenisation bias at the end of training

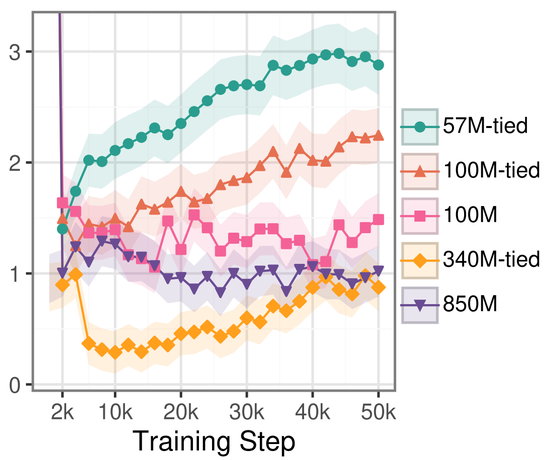


- We estimate this bias to be 2.88 nats. This means a can be assigned roughly 17x less probability due solely to tokenisation
- Subword inclusion significantly affects output stability: in-vocabulary subwords exhibit far less variation in log-probability across contexts than out-of-vocabulary ones
- RDD estimates are robust to
  - Functional form
  - Window size, for sizes >500

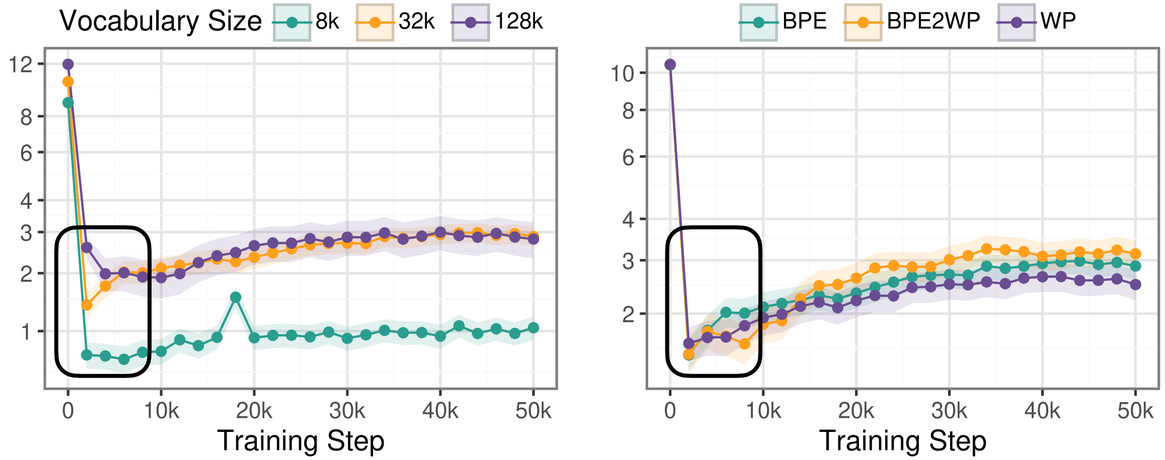
Tokenisation bias across training

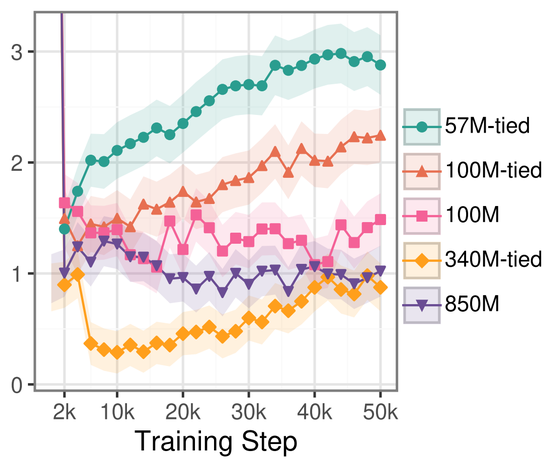






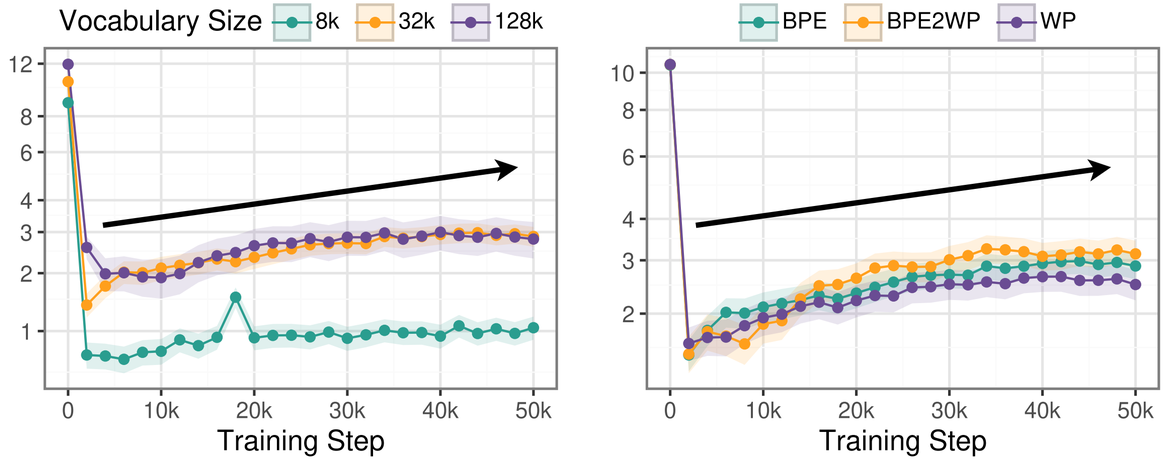
Tokenisation bias across training

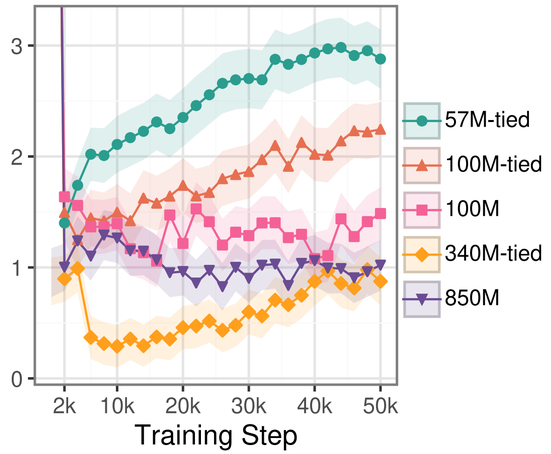




- Drops sharply by step 2k, as expected

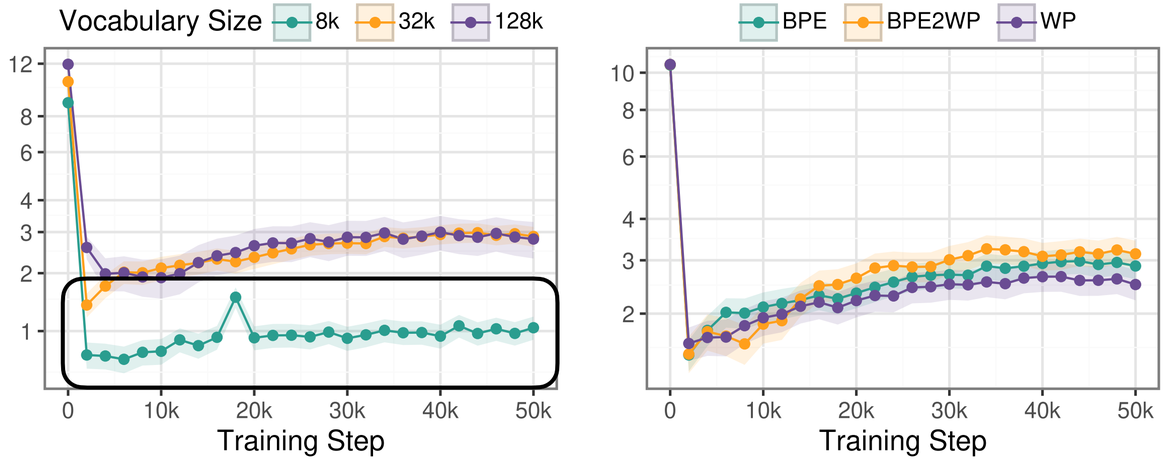
Tokenisation bias across training

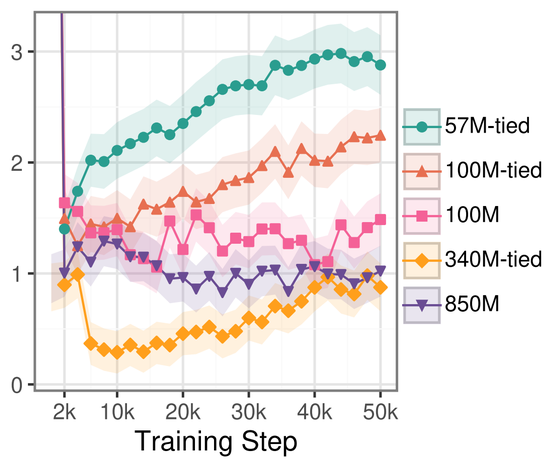




- Drops sharply by step 2k, as expected
- Grows as models improve, contrary to expectations

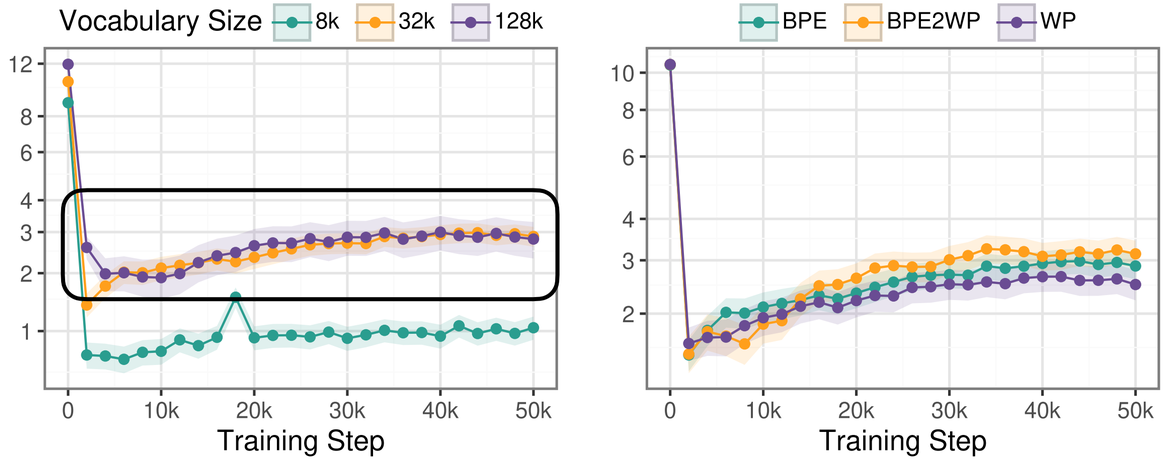
Tokenisation bias across training

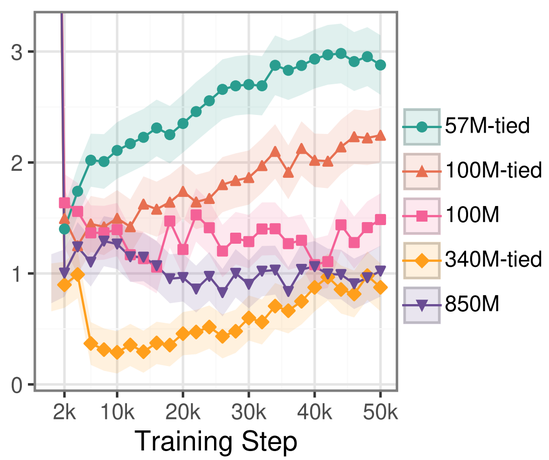




- Drops sharply by step 2k, as expected
- Grows as models improve, contrary to expectations
- Weaker biases when using smaller vocabularies (perhaps due to more frequent subwords even beyond the cutoff)

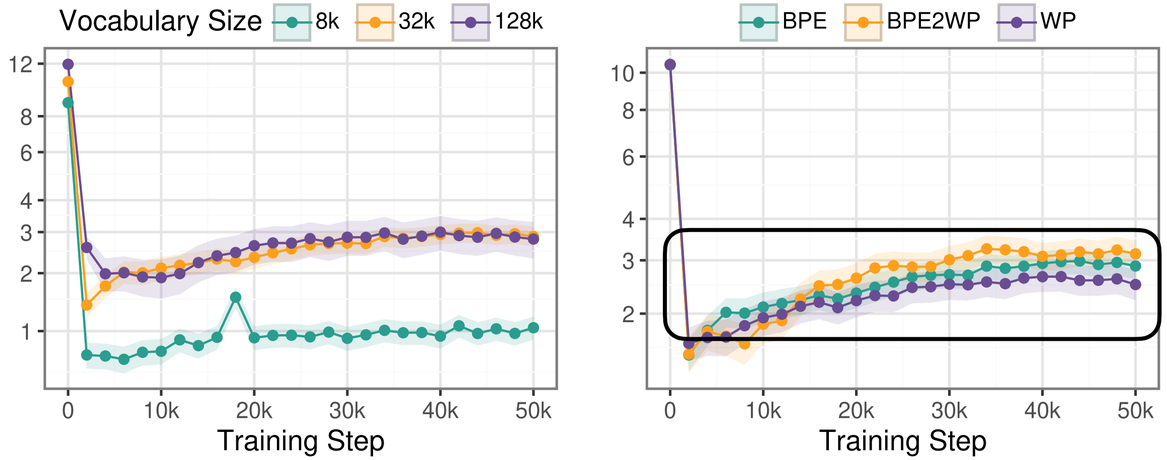
Tokenisation bias across training

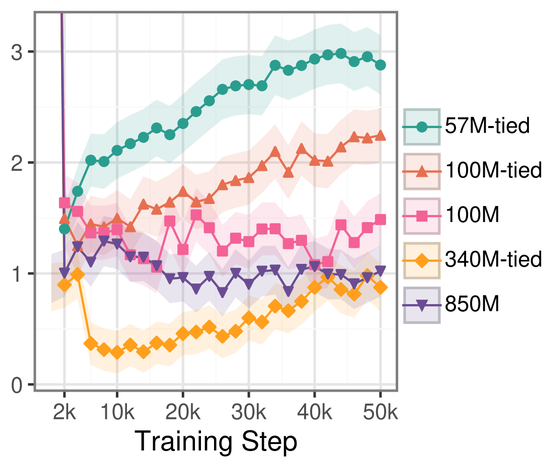




- Drops sharply by step 2k, as expected
- Grows as models improve, contrary to expectations
- Weaker biases when using smaller vocabularies (perhaps due to more frequent subwords even beyond the cutoff)
- Beyond a critical vocabulary size, the effect stabilises, with 128k and 32k showing similar biases despite a large difference

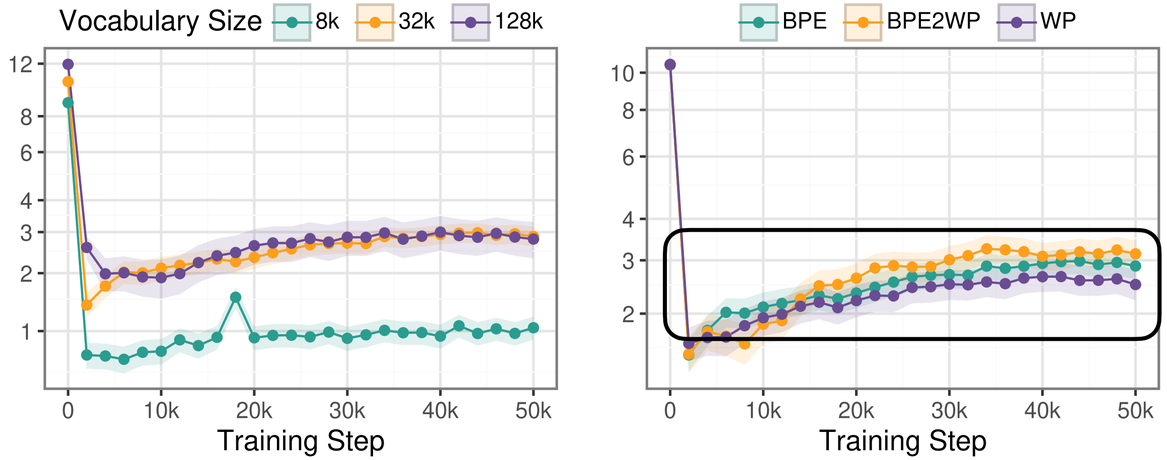
Tokenisation bias across training



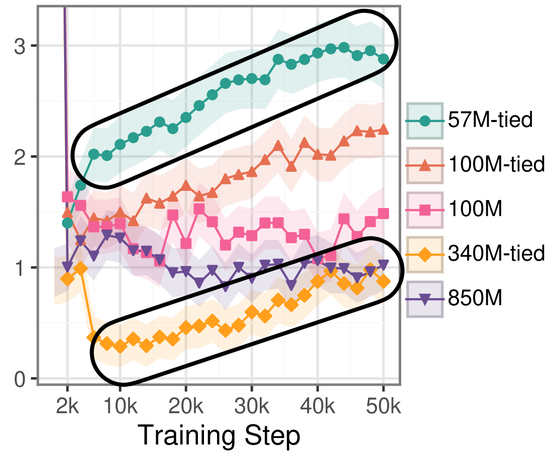


- Drops sharply by step 2k, as expected
- Grows as models improve, contrary to expectations
- Weaker biases when using smaller vocabularies (perhaps due to more frequent subwords even beyond the cutoff)
- Beyond a critical vocabulary size, the effect stabilises, with 128k and 32k showing similar biases despite a large difference
- Consistent across these tokenisers

Tokenisation bias across training

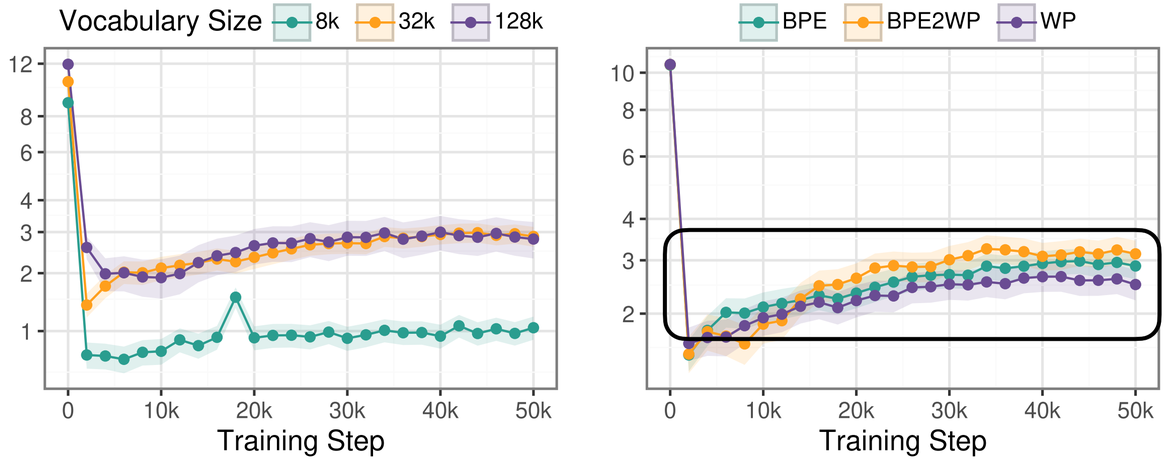


- Drops sharply by step 2k, as expected
- Grows as models improve, contrary to expectations
- Weaker biases when using smaller vocabularies (perhaps due to more frequent subwords even beyond the cutoff)
- Beyond a critical vocabulary size, the effect stabilises, with 128k and 32k showing similar biases despite a large difference
- Consistent across these tokenisers

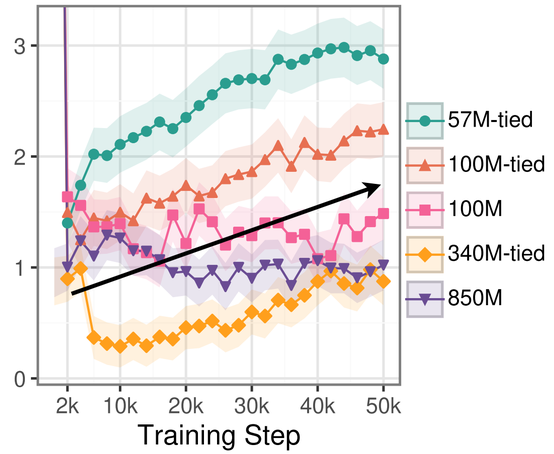


- Smaller bias in larger (e.g., 340M parameters) models than in small models (e.g., 57M)

Tokenisation bias across training

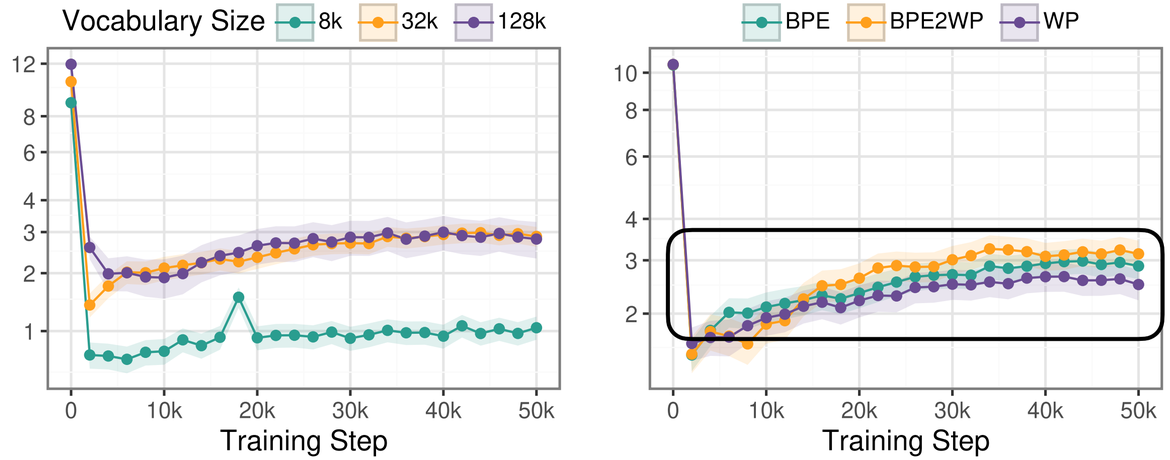


- Drops sharply by step 2k, as expected
- Grows as models improve, contrary to expectations
- Weaker biases when using smaller vocabularies (perhaps due to more frequent subwords even beyond the cutoff)
- Beyond a critical vocabulary size, the effect stabilises, with 128k and 32k showing similar biases despite a large difference
- Consistent across these tokenisers

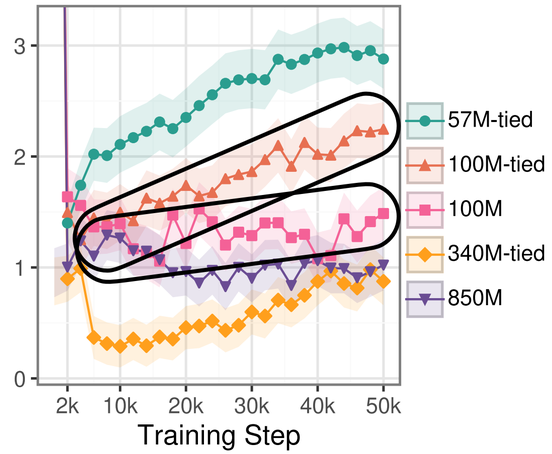


- Smaller bias in larger (e.g., 340M parameters) models than in small models (e.g., 57M)
- Across training, bias does not seem to decrease even for larger (better) models

Tokenisation bias across training

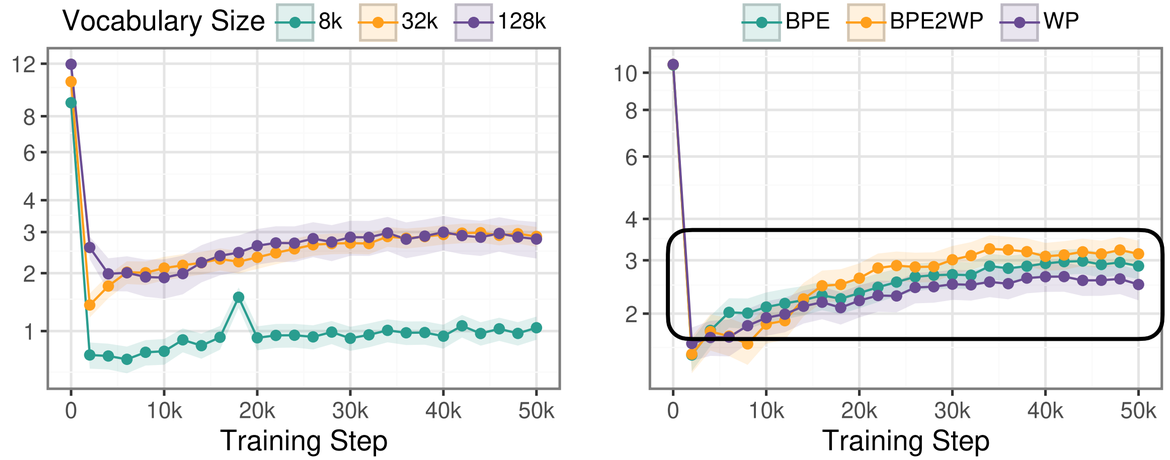


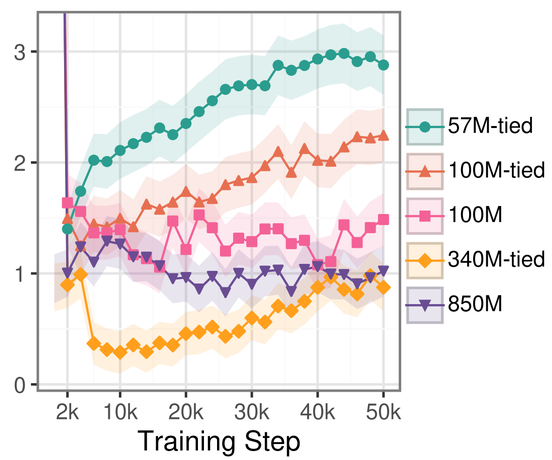
- Drops sharply by step 2k, as expected
- Grows as models improve, contrary to expectations
- Weaker biases when using smaller vocabularies (perhaps due to more frequent subwords even beyond the cutoff)
- Beyond a critical vocabulary size, the effect stabilises, with 128k and 32k showing similar biases despite a large difference
- Consistent across these tokenisers



- Smaller bias in larger (e.g., 340M parameters) models than in small models (e.g., 57M)
- Across training, bias does not seem to decrease even for larger (better) models
- Embedding tying increases tokenisation bias (see 100M) vs. untying

Tokenisation bias across training





- Drops sharply by step 2k, as expected
- Grows as models improve, contrary to expectations
- Weaker biases when using smaller vocabularies (perhaps due to more frequent subwords even beyond the cutoff)
- Beyond a critical vocabulary size, the effect stabilises, with 128k and 32k showing similar biases despite a large difference
- Consistent across these tokenisers

- Smaller bias in larger (e.g., 340M parameters) models than in small models (e.g., 57M)
- Across training, bias does not seem to decrease even for larger (better) models
- Embedding tying increases tokenisation bias (see 100M) vs. untying

Together, these results suggest that tokenisation bias may be a persisting property of LMs

FINE

Pietro Lesci
University of Cambridge

X: @pietro_lesci
LinkedIn: /pietrolesci
Page: pietrolesci.com
Mail: pietrolesci@outlook.com