NEURIPS 2026

Causal Edit Serialization

An autoregressive model can point to earlier text and generate instructions to revise it.

Louis Robinson

Independent Researcher

Paper · arxiv.org/abs/2610.22820

Tighten wording and correct agreement

DELETE + REPLACE

RESULTING SEQUENCE

LIVE PREVIEW

We

Applied incrementally for illustration

AUTOREGRESSIVE TOKEN STREAM

pointer / copy

move

remove

payload

SOURCE · FIXED ADDRESSES

We

0

EDIT PROGRAM

GENERATE THE SOURCE

Source tokens appear one at a time. Their addresses stay fixed.

Pointers use original source addresses. Copies retain their source; moves relocate it.

Tighten wording and correct agreement

DELETE + REPLACE

RESULTING SEQUENCE

LIVE PREVIEW

We conducted

Applied incrementally for illustration

AUTOREGRESSIVE TOKEN STREAM

pointer / copy

move

remove

payload

SOURCE · FIXED ADDRESSES

We	conducted
0	1

EDIT PROGRAM

GENERATE THE SOURCE

Source tokens appear one at a time. Their addresses stay fixed.

Pointers use original source addresses. Copies retain their source; moves relocate it.

Tighten wording and correct agreement

DELETE + REPLACE

RESULTING SEQUENCE

LIVE PREVIEW

We conducted a total of three trials. These result is consistent.

Applied incrementally for illustration

AUTOREGRESSIVE TOKEN STREAM

pointer / copy · move · remove · payload

SOURCE · FIXED ADDRESSES

We	conducted	a	total	of	three	trials	.	These	result	is	consistent	.
0	1	2	3	4	5	6	7	8	9	10	11	12

EDIT PROGRAM

GENERATE THE SOURCE

Source tokens appear one at a time. Their addresses stay fixed.

Pointers use original source addresses. Copies retain their source; moves relocate it.

Tighten wording and correct agreement

DELETE + REPLACE

RESULTING SEQUENCE

LIVE PREVIEW

We conducted a total of three trials. These result is consistent.

Applied incrementally for illustration

AUTOREGRESSIVE TOKEN STREAM

pointer / copy · move · remove · payload

SOURCE · FIXED ADDRESSES

We	conducted	a	total	of	three	trials	.	These	result	is	consistent	.
0	1	2	3	4	5	6	7	8	9	10	11	12

EDIT PROGRAM

EDIT START
13

BEGIN EDIT PROGRAM

Append EDIT_START after the source.

Pointers use original source addresses. Copies retain their source; moves relocate it.

Tighten wording and correct agreement

DELETE + REPLACE

RESULTING SEQUENCE

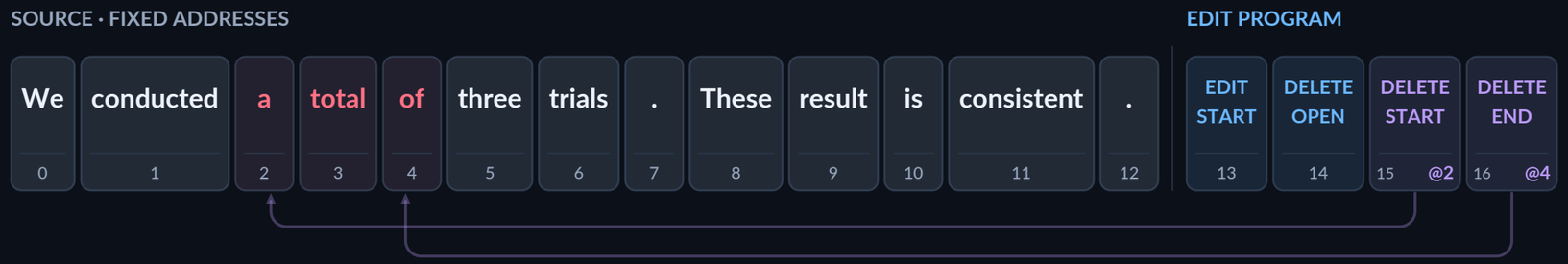
LIVE PREVIEW

We conducted three trials. These result is consistent.

Applied incrementally for illustration

AUTOREGRESSIVE TOKEN STREAM

pointer / copy · move · remove · payload



REMOVE THE SELECTED SPAN

Remove “a total of”; the sentence keeps its meaning.

Pointers use original source addresses. Copies retain their source; moves relocate it.

Tighten wording and correct agreement

DELETE + REPLACE

RESULTING SEQUENCE

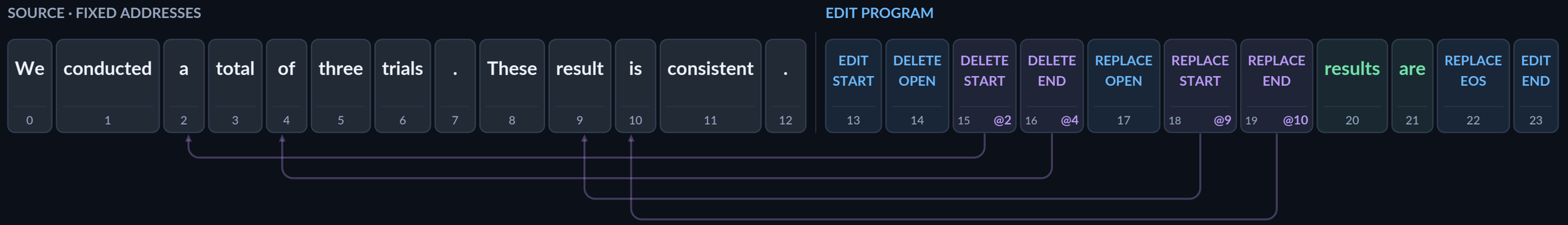
COMMITTED

We conducted three trials. These results are consistent.

The completed program has been applied

AUTOREGRESSIVE TOKEN STREAM

pointer / copy · move · remove · payload



RESULT

We conducted three trials. These results are consistent.

Pointers use original source addresses. Copies retain their source; moves relocate it.

Revise a development checklist

INSERT + COPY + MOVE

RESULTING SEQUENCE

LIVE PREVIEW

Save script. Commit changes. Run tests. Fix if needed.

Applied incrementally for illustration

AUTOREGRESSIVE TOKEN STREAM

pointer / copy · move · remove · payload

SOURCE · FIXED ADDRESSES

Save	script	.	Commit	changes	.	Run	tests	.	Fix	if	needed	.
0	1	2	3	4	5	6	7	8	9	10	11	12

EDIT PROGRAM

GENERATE THE SOURCE

Source tokens appear one at a time. Their addresses stay fixed.

Pointers use original source addresses. Copies retain their source; moves relocate it.

Revise a development checklist

INSERT + COPY + MOVE

RESULTING SEQUENCE

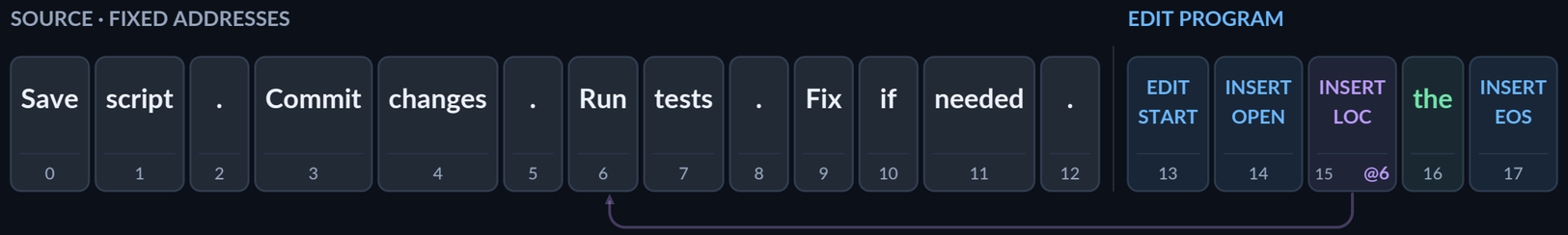
LIVE PREVIEW

Save script. Commit changes. Run the tests. Fix if needed.

Applied incrementally for illustration

AUTOREGRESSIVE TOKEN STREAM

pointer / copy · move · remove · payload



CLOSE THE PAYLOAD

INSERT_EOS closes the payload.

Pointers use original source addresses. Copies retain their source; moves relocate it.

Revise a development checklist

INSERT + COPY + MOVE

RESULTING SEQUENCE

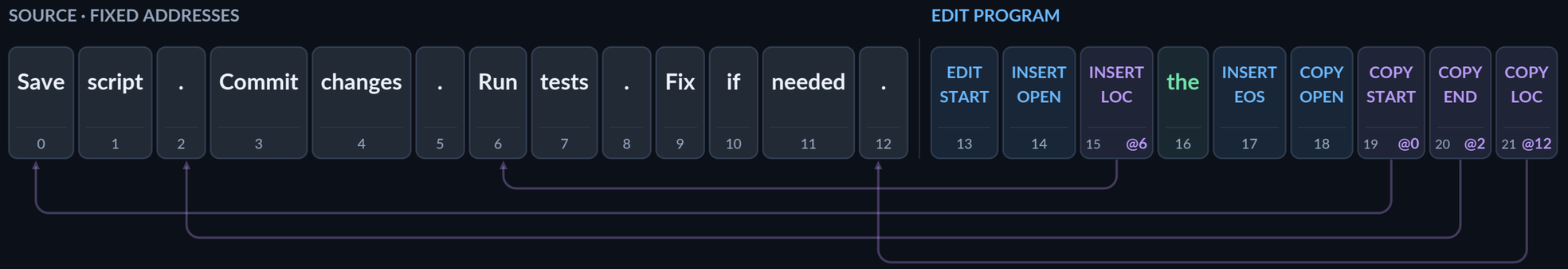
LIVE PREVIEW

Save script. Commit changes. Run the tests. Fix if needed. Save script.

Applied incrementally for illustration

AUTOREGRESSIVE TOKEN STREAM

pointer / copy · move · remove · payload



COPY THE SPAN

Duplicate “Save script.” at the end. The original stays in place.

Pointers use original source addresses. Copies retain their source; moves relocate it.

Revise a development checklist

INSERT + COPY + MOVE

RESULTING SEQUENCE

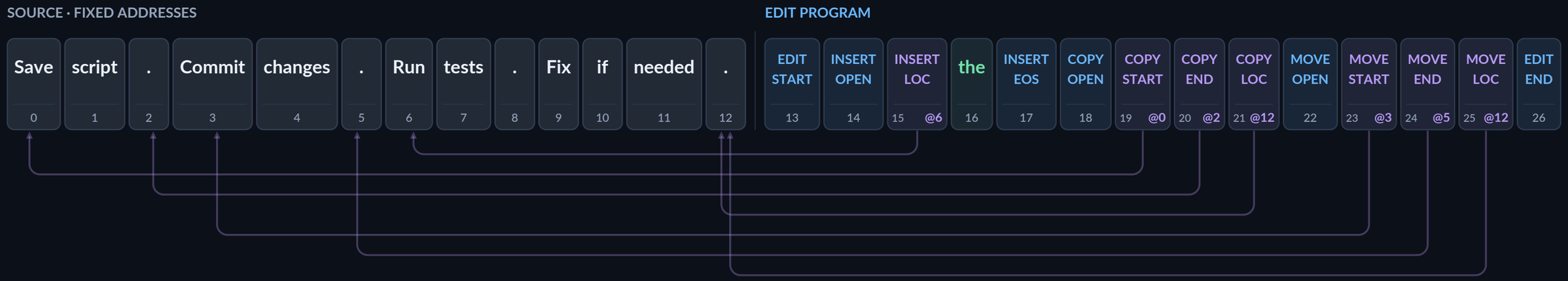
COMMITTED

Save script. Run the tests. Fix if needed. Save script. Commit changes.

The completed program has been applied

AUTOREGRESSIVE TOKEN STREAM

pointer / copy · move · remove · payload

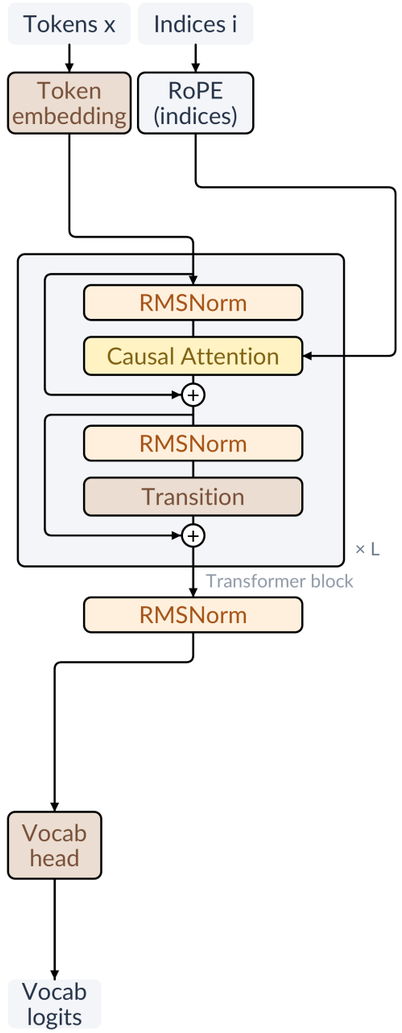


RESULT

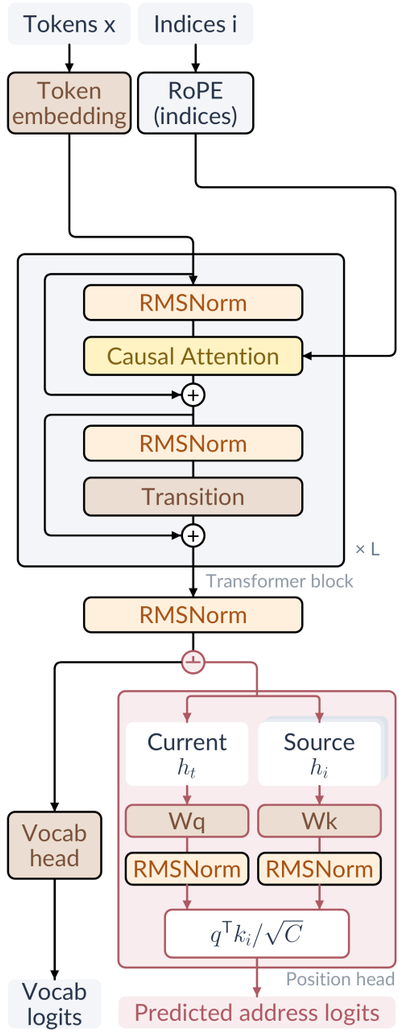
Save script. Run the tests. Fix if needed. Save script. Commit changes.

Pointers use original source addresses. Copies retain their source; moves relocate it.

Start with a standard causal transformer



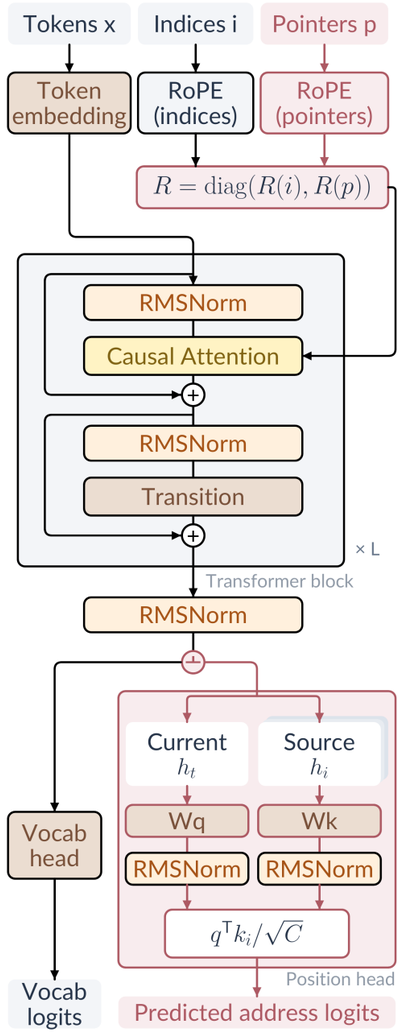
Predict source addresses directly



- **Localization head**

  Predicts a legal source address directly.
  The vocabulary head predicts the pointer role.

Condition on the predicted pointers



- **Dual-index RoPE**

  Splits query/key channels between stream position and source address, so later tokens can condition on the predicted pointers.

- **Localization head**

  Predicts a legal source address directly.
  The vocabulary head predicts the pointer role.

Paper · Figure 1 and Methods · full CES architecture

Standard RoPE

Rotate each copy by (i − j)ω

depth 24

C = 1536 · d_h = 128 · 12 heads

CURRENT QUERY

token	She	walk	to	the	store	.	EDIT START	REPLACE OPEN
index j	0	1	2	3	4	5	6	7
distance d = i − j	7	6	5	4	3	2	1	0

Frequency groups

(i − j)ω

r = 0

ω = 1.000

rad / token



r = 8

ω = 0.316

rad / token



CLOCKS PER FREQUENCY

$$\text{clocks} = \frac{\text{channels}}{\text{freqgroups} \times 2} = \frac{1536}{64 \times 2} = 12$$

12 heads · 1 clock per head

Total frequency groups

d_h / 2 = 128 / 2 = 64

Localization Head

Mask non-source positions · exponentiate the scores · normalize with softmax

depth 24

C = 1536 · d_h = 128 · 12 heads

token	She	walk	to	the	store	.	EDIT START	REPLACE OPEN
index j	0	1	2	3	4	5	6	7

CURRENT QUERY

Transformer

Attention

Attention · RoPE (magnified)

angle = (i − j)ω

d = i − j	7	6	5	4	3	2	1	0
r = 0, ω = 1.000 rad / token	7.000 rad	6.000 rad	5.000 rad	4.000 rad	3.000 rad	2.000 rad	1.000 rad	0.000 rad
r = 8, ω = 0.316 rad / token	2.214 rad	1.897 rad	1.581 rad	1.265 rad	0.949 rad	0.632 rad	0.316 rad	0.000 rad

128 channels/head

64 frequency groups · 12 clocks each

LOCALIZATION HEAD

$s_j = q^T k_j / \sqrt{1536}$

Illustrative activations and scores · 8 components shown / 1536

	h_0	h_1	h_2	h_3	h_4	h_5	h_6	h_7
activations h_j								
project + normalize	Wk + RMS	Wk + RMS	Wk + RMS	Wk + RMS	Wk + RMS	Wk + RMS	Wk + RMS	Wq + RMS
$q = \text{RMSNorm}(W_q h_7)$								
logits	0.400	2.000	-0.300	0.200	-0.600	-1.000	−∞ masked	−∞ masked
$k_j = \text{RMSNorm}(W_k h_j)$								

$P(j) = \exp(s_j) / \Sigma \exp(s_k)$

softmax

Vocabulary head → REPLACE START	0.127	0.628	0.063	0.104	0.047	0.031	0.000	0.000

Localization Head

depth 24

Sample a source address from the softmax distribution: position 1, “walk”

$C = 1536 \cdot d_h = 128 \cdot 12$ heads

CURRENT QUERY

token	She	walk	to	the	store	.	EDIT START	REPLACE OPEN
index j	0	1	2	3	4	5	6	7

Transformer

Attention

Attention · RoPE (magnified)

angle = (i − j)ω

d = i − j	7	6	5	4	3	2	1	0
r = 0, ω = 1.000 rad / token	7.000 rad	6.000 rad	5.000 rad	4.000 rad	3.000 rad	2.000 rad	1.000 rad	0.000 rad
r = 8, ω = 0.316 rad / token	2.214 rad	1.897 rad	1.581 rad	1.265 rad	0.949 rad	0.632 rad	0.316 rad	0.000 rad

128 channels/head

64 frequency groups · 12 clocks each

LOCALIZATION HEAD

$s_j = q^T k_j / \sqrt{1536}$

Illustrative activations and scores · 8 components shown / 1536

activations h_j

h_0 h_1 h_2 h_3 h_4 h_5 h_6 h_7

project + normalize

$q = \text{RMSNorm}(W_q\, h_7)$

$k_j = \text{RMSNorm}(W_k\, h_j)$

Wk + RMS | Wk + RMS | Wk + RMS | Wk + RMS | Wk + RMS | Wk + RMS | Wk + RMS | Wq + RMS

logits

0.400 | 2.000 | -0.300 | 0.200 | -0.600 | -1.000 | −∞ masked | −∞ masked

softmax

$P(j) = \exp(s_j) / \Sigma \exp(s_k)$

0.127 | 0.628 | 0.063 | 0.104 | 0.047 | 0.031 | 0.000 | 0.000

0 | 1 | 2 | 3 | 4 | 5

Vocabulary head → REPLACE START

sample u = 0.324

selected: @1

Localization Head

Append REPLACE START with its sampled address @1

depth 24

C = 1536 · d_h = 128 · 12 heads

CURRENT QUERY

token	She	walk	to	the	store	.	EDIT START	REPLACE OPEN	REPLACE START
index j	0	1	2	3	4	5	6	7	8 @1
pointer index p_j	0	1	2	3	4	5	6	7	1

Transformer

Attention

Attention · RoPE (magnified)

angle = (i − j)ω

d = i − j	8	7	6	5	4	3	2	1	0
r = 0, ω = 1.000 rad / token	8.000 rad	7.000 rad	6.000 rad	5.000 rad	4.000 rad	3.000 rad	2.000 rad	1.000 rad	0.000 rad
r = 8, ω = 0.316 rad / token	2.530 rad	2.214 rad	1.897 rad	1.581 rad	1.265 rad	0.949 rad	0.632 rad	0.316 rad	0.000 rad

128 channels/head

64 frequency groups · 12 clocks each

LOCALIZATION HEAD

$s_j = q^T k_j / \sqrt{1536}$

Illustrative activations and scores · 8 components shown / 1536

activations h_j

	h_0	h_1	h_2	h_3	h_4	h_5	h_6	h_7
project + normalize	Wk + RMS	Wk + RMS	Wk + RMS	Wk + RMS	Wk + RMS	Wk + RMS	Wk + RMS	Wq + RMS
$q = \text{RMSNorm}(W_q\, h_7)$								
logits	0.400	2.000	-0.300	0.200	-0.600	-1.000	−∞ masked	−∞ masked
$k_j = \text{RMSNorm}(W_k\, h_j)$								

$P(j) = \exp(s_j) / \Sigma \exp(s_k)$

softmax

	0.127	0.628	0.063	0.104	0.047	0.031	0.000	0.000
	0	1	2	3	4	5		selected: @1

Vocabulary head → REPLACE START

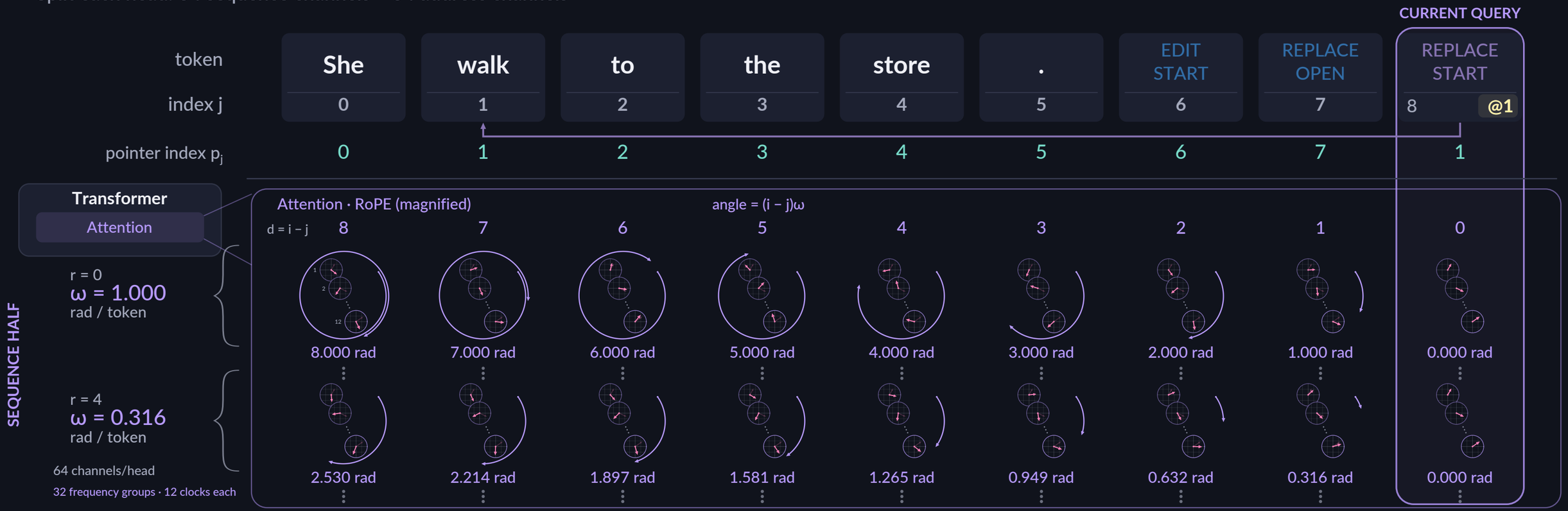
sample u = 0.324

Dual-index RoPE

Split each head: 64 sequence channels + 64 address channels

depth 24

C = 1536 · d_h = 128 · 12 heads



Dual-index RoPE

REPLACE START points to “walk” · its address offset there is zero

depth 24

C = 1536 · d_h = 128 · 12 heads

CURRENT QUERY

	She	walk	to	the	store	.	EDIT START	REPLACE OPEN	REPLACE START
token									
index j	0	1	2	3	4	5	6	7	8 @1
pointer index p_j	0	1	2	3	4	5	6	7	1

Transformer

Attention

Attention · RoPE (magnified)

angle = (i − j)ω

d = i − j

	8	7	6	5	4	3	2	1	0
SEQUENCE HALF r = 0 ω = 1.000 rad / token	8.000 rad	7.000 rad	6.000 rad	5.000 rad	4.000 rad	3.000 rad	2.000 rad	1.000 rad	0.000 rad
SEQUENCE HALF r = 4 ω = 0.316 rad / token	2.530 rad	2.214 rad	1.897 rad	1.581 rad	1.265 rad	0.949 rad	0.632 rad	0.316 rad	0.000 rad

64 channels/head

32 frequency groups · 12 clocks each

angle = (p_i − p_j)ω

distance p_i − p_j

	1	0	-1	-2	-3	-4	-5	-6	0
ADDRESS HALF r = 0 ω = 1.000 rad / token	1.000 rad	0.000 rad	-1.000 rad	-2.000 rad	-3.000 rad	-4.000 rad	-5.000 rad	-6.000 rad	0.000 rad
ADDRESS HALF r = 4 ω = 0.316 rad / token	0.316 rad	0.000 rad	-0.316 rad	-0.632 rad	-0.949 rad	-1.265 rad	-1.581 rad	-1.897 rad	0.000 rad

64 channels/head

32 frequency groups · 12 clocks each

Pretraining mixes language modelling with repair

ClimbMix text + serialized edit programs

- **Random corruptions**

  Locations: f(indices; randomness)

Sample edit locations and lengths

Gaps between candidate edits

$$\Delta^{\mathrm{del}} \sim \mathrm{Geom}(p^{\mathrm{del}}_{\mathrm{start}}), \quad \Delta^{\mathrm{ins}} \sim \mathrm{Geom}(p^{\mathrm{ins}}_{\mathrm{start}})$$

Lengths of inserted / deleted spans

$$L^{\mathrm{del}} \sim \mathrm{Geom}(p^{\mathrm{del}}_{\mathrm{end}}), \quad L^{\mathrm{ins}} \sim \mathrm{Geom}(p^{\mathrm{ins}}_{\mathrm{end}})$$

$$p = 1/\mu, \qquad \mathbb{E}[\mathrm{Geom}(p)] = \mu$$

Earlier event wins; stop at the token budget.
Payloads mix document spans and random tokens.

Pretraining mixes language modelling with repair

ClimbMix text + serialized edit programs

- **Random corruptions**

  Locations: f(indices; randomness)

- **GPT-5 Mini revisions**

  f(data) · content-dependent edits

GPT-5 Mini prompt (excerpt)

Keep the same language and general topic.
Preserve most of the original text.

Prefer several short, independent, local edits over one broad rewrite.

Output only valid operation lines, with no numbering and no extra commentary.

Del(…) **Replace(…)** **Insert(…)** **Move(…)**

Parse operations → resolve source anchors → serialize token-coordinate edits.

Generation prompt · nanochat/nanochat/edit_dataset.py · selected instructions

Pretraining mixes language modelling with repair

ClimbMix text + serialized edit programs

- **Random corruptions**

  Locations: f(indices; randomness)

- **GPT-5 Mini revisions**

  f(data) · content-dependent edits

- **Copy mining**

  f(data) · repeated text in the document

Remove a repeat; learn to copy it back

Find an earlier occurrence of the same span.

ORIGINAL

… same span … same span …

TRAINING SOURCE

… same span … insertion site …

Target program: COPY the earlier span.

Executing it restores the original document.

Pretraining mixes language modelling with repair

ClimbMix text + serialized edit programs

- **Random corruptions**

  Locations: f(indices; randomness)

- **GPT-5 Mini revisions**

  f(data) · content-dependent edits

- **Copy mining**

  f(data) · repeated text in the document

Subsample stored edit candidates

Sample compatible edits within the token budget.
Mix repair streams with ordinary language modelling.
Adjust sampling exposure; measure the resulting ρ.

$$\rho = \frac{\text{total script edit mass}}{\text{total raw source tokens consumed}}$$

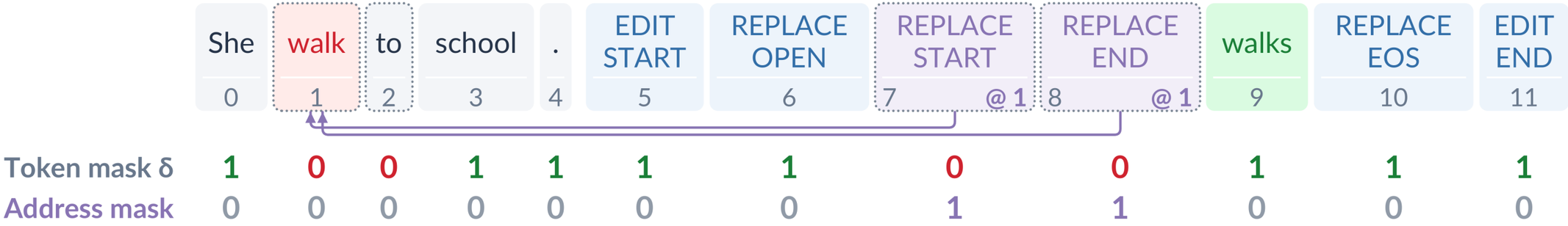
ρ ≈ 0.0236 at control 1.0 (depth-18 sweep)

Edit mass excludes operation and pointer tokens.

Train token predictions and source addresses

1 = supervised · 0 = masked. Pointer addresses have their own supervision.

Replace one word → She walks to school.



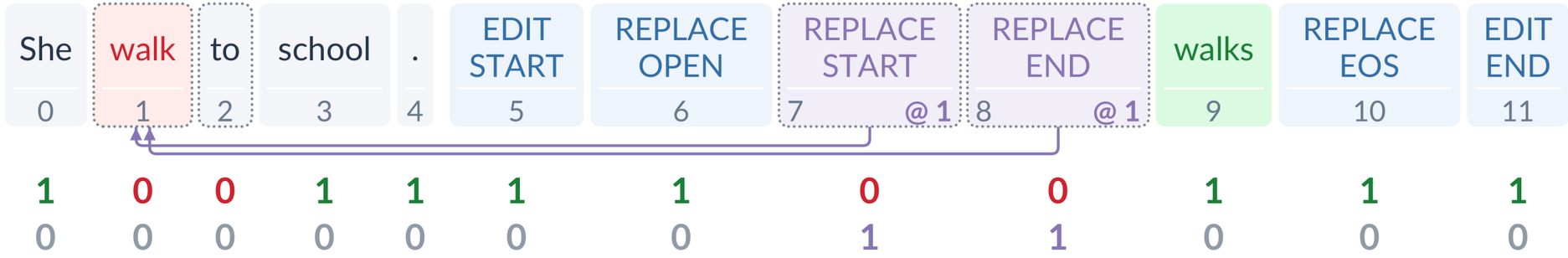
$$\mathcal{L} = \frac{\sum_t \delta_t \, \mathcal{L}_{\text{token}}(t)}{\sum_t \delta_t} + \lambda_{\text{addr}} \frac{\sum_{t \in \mathcal{A}} \mathcal{L}_{\text{addr}}(t)}{|\mathcal{A}|}, \qquad \lambda_{\text{addr}} = 0.01$$

Paper · Figure 1 A/D and Training Objective · masks align with targets; preceding context omitted

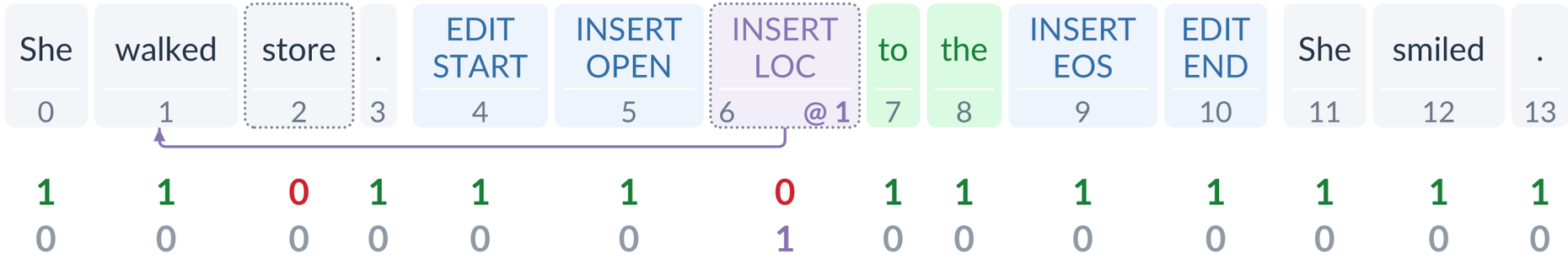
Train token predictions and source addresses

1 = supervised · 0 = masked. Pointer addresses have their own supervision.

Replace one word → She walks to school.



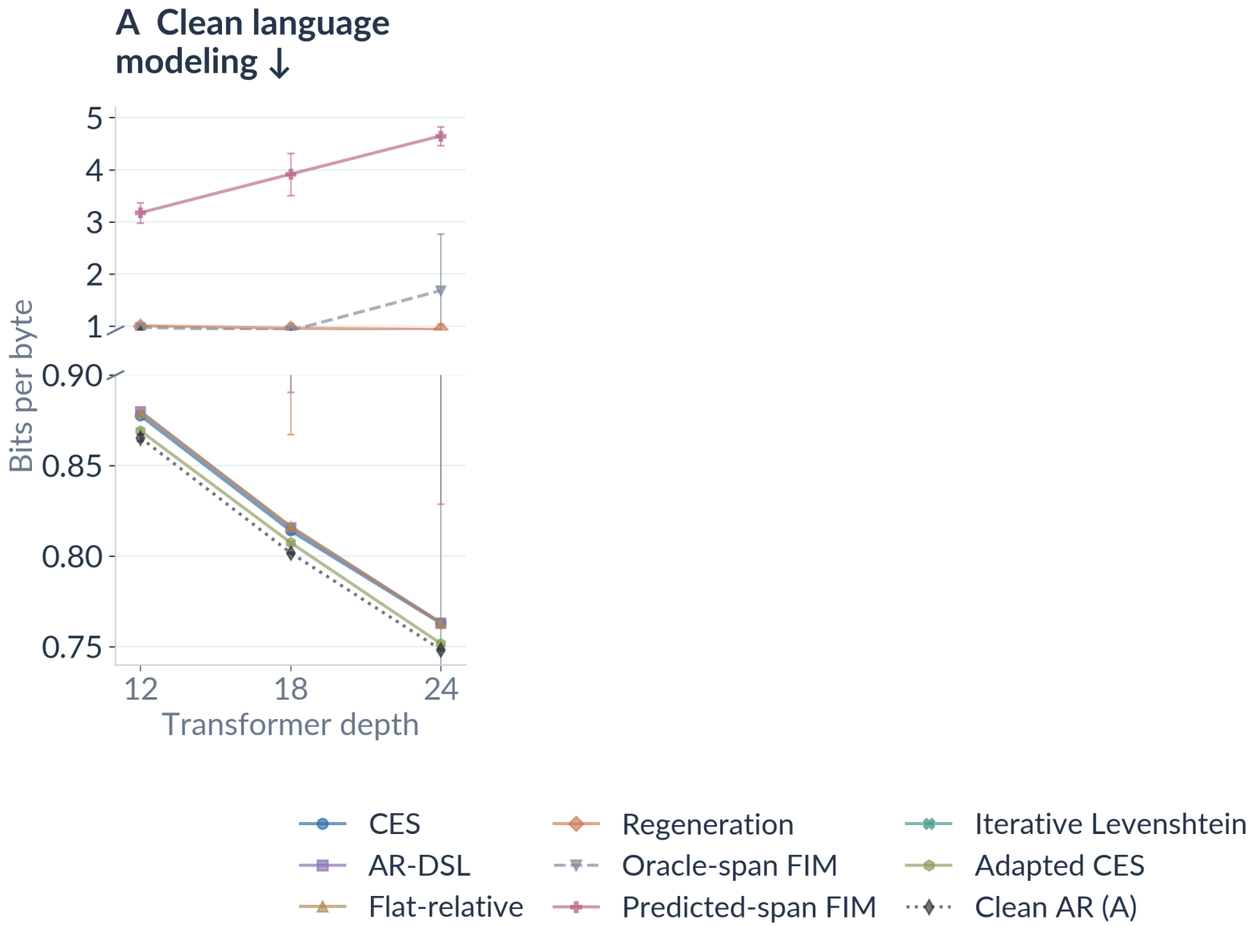
Insert a phrase, then continue → She walked to the store. She smiled.



$$\mathcal{L} = \frac{\sum_t \delta_t \, \mathcal{L}_{\text{token}}(t)}{\sum_t \delta_t} + \lambda_{\text{addr}} \frac{\sum_{t \in \mathcal{A}} \mathcal{L}_{\text{addr}}(t)}{|\mathcal{A}|}, \qquad \lambda_{\text{addr}} = 0.01$$

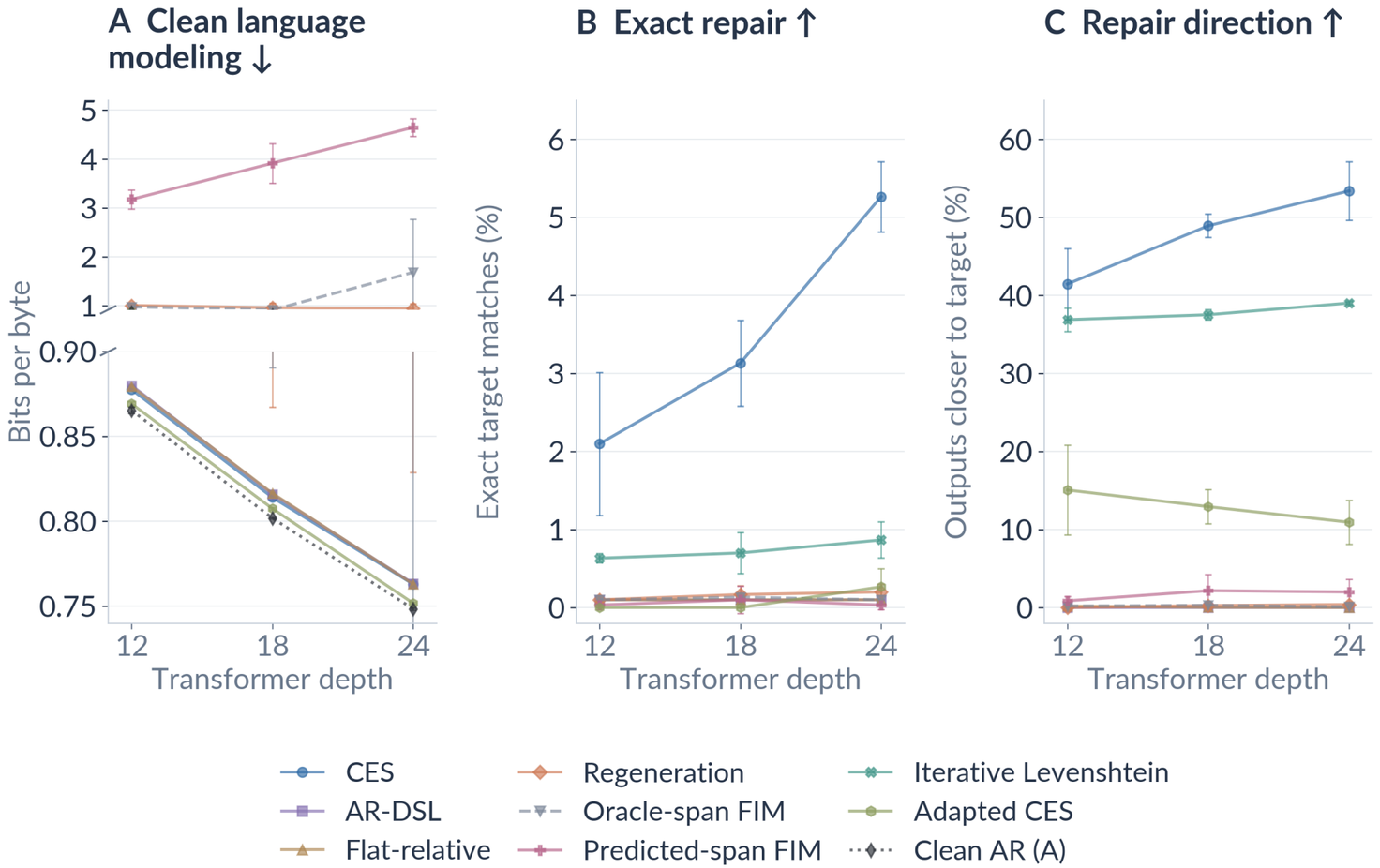
Paper · Figure 1 A/D and Training Objective · masks align with targets; preceding context omitted

Repair ability and language modelling



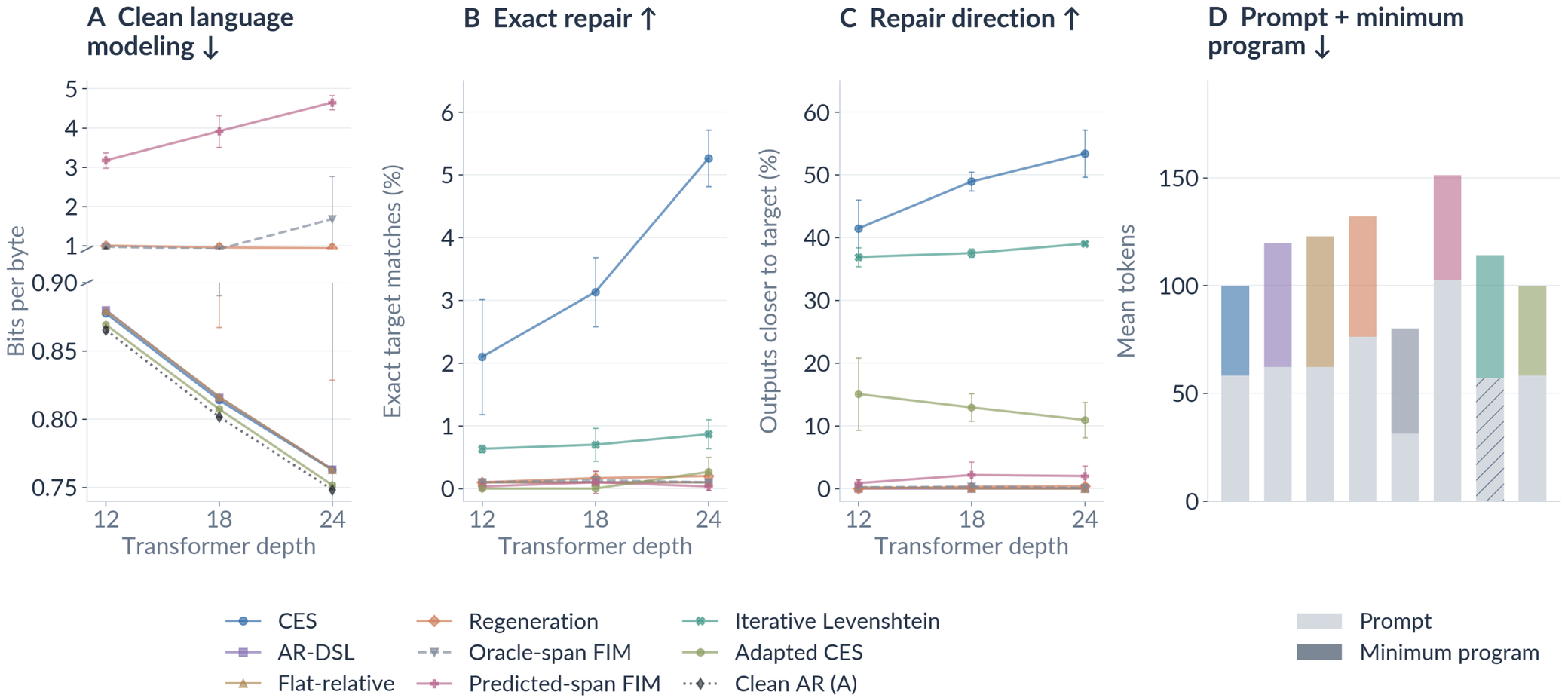
Poster Figure 2 A · clean-text BPB · mean ± sample SD over 3 seeds

Repair ability and language modelling



Poster Figure 2 A–C · pretrained models · mean ± sample SD over 3 seeds

Repair ability and language modelling



Poster Figure 2 · A–C: pretraining, mean ± SD over 3 seeds · D: CoEdIT representation lengths

Post-training rewards executed repairs

Balanced SFT

800 updates

Human edits + clean no-ops,
synthetic repairs + continuations.

Post-training rewards executed repairs

Balanced SFT

800 updates

Human edits + clean no-ops,
synthetic repairs + continuations.

Broad repair RL

100 updates

Model-generated + random
corruptions, and clean no-ops.

Recovery after executing the program

$$\Delta_k = \mathrm{clip}\left(\frac{d_k(x, y^*) - d_k(\hat{y}, y^*)}{\max(d_k(x, y^*), 1)}, -1, 1\right)$$

x: source; y*: clean target; ŷ: executed output.
k: token or character Levenshtein distance.

Reward adds exact recovery and penalizes
invalid, excessive and missed edits.

Leave-one-out policy updates

$$A_i = r_i - \frac{1}{G-1}\sum_{j \neq i} r_j, \qquad G = 8$$

$$\mathcal{L}_{\mathrm{RL}} = -\frac{\sum_i A_i \sum_{t \in \mathcal{D}_i} \log \pi_\theta(a_{i,t} \mid h_{i,t})}{\sum_i |\mathcal{D}_i|}$$

Sampled token + pointer decisions in D_i.
RL is mixed with supervised losses.

Post-training rewards executed repairs

Balanced SFT

800 updates

Human edits + clean no-ops, synthetic repairs + continuations.

Broad repair RL

100 updates

Model-generated + random corruptions, and clean no-ops.

GEC-focused RL

200 updates

Localized corruptions of clean CoEdIT training targets.

Recovery after executing the program

$$\Delta_k = \operatorname{clip}\left(\frac{d_k(x, y^*) - d_k(\hat{y}, y^*)}{\max(d_k(x, y^*), 1)}, -1, 1\right)$$

x: source; y*: clean target; ŷ: executed output.
k: token or character Levenshtein distance.

Reward adds exact recovery and penalizes invalid, excessive and missed edits.

Leave-one-out policy updates

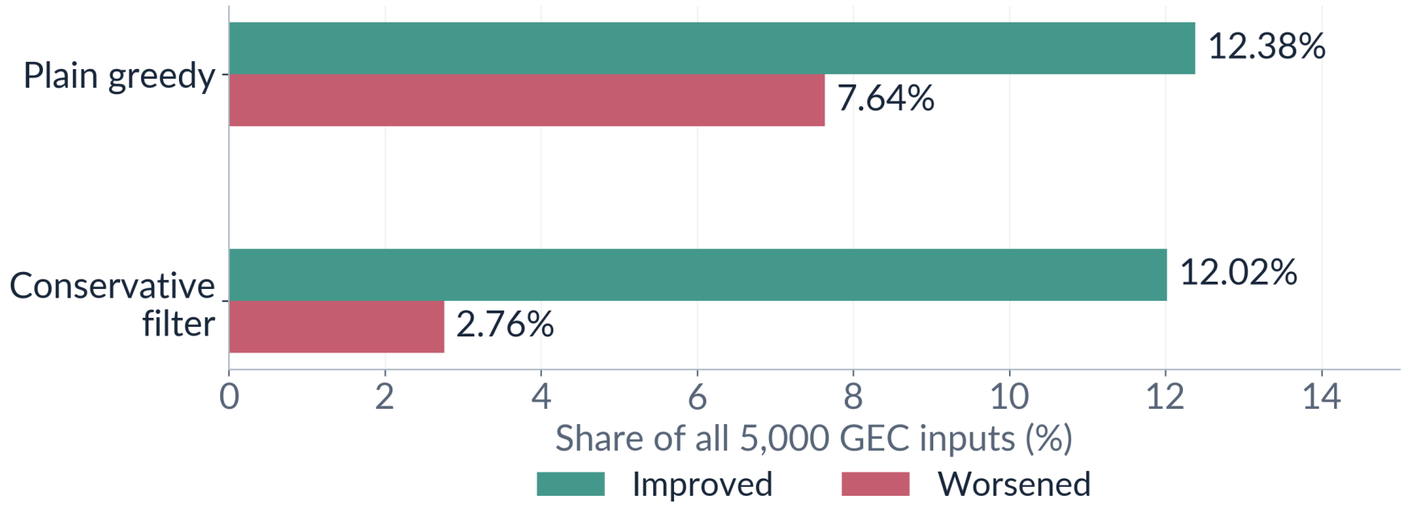
$$A_i = r_i - \frac{1}{G-1}\sum_{j \neq i} r_j, \qquad G = 8$$

$$\mathcal{L}_{\mathrm{RL}} = -\frac{\sum_i A_i \sum_{t \in \mathcal{D}_i} \log \pi_\theta(a_{i,t} \mid h_{i,t})}{\sum_i |\mathcal{D}_i|}$$

Sampled token + pointer decisions in D_i.
RL is mixed with supervised losses.

Post-training produces positive net improvement

Reference-blind GPT-5 Mini judgments · 5,000 CoEdIT inputs



NET IMPROVEMENT

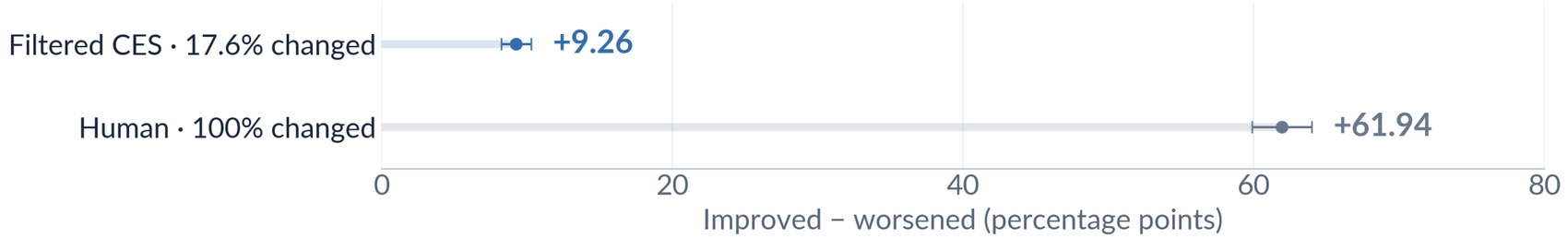
+4.74 pp

Plain greedy
22.74% changed

+9.26 pp

Conservative filter
17.60% changed

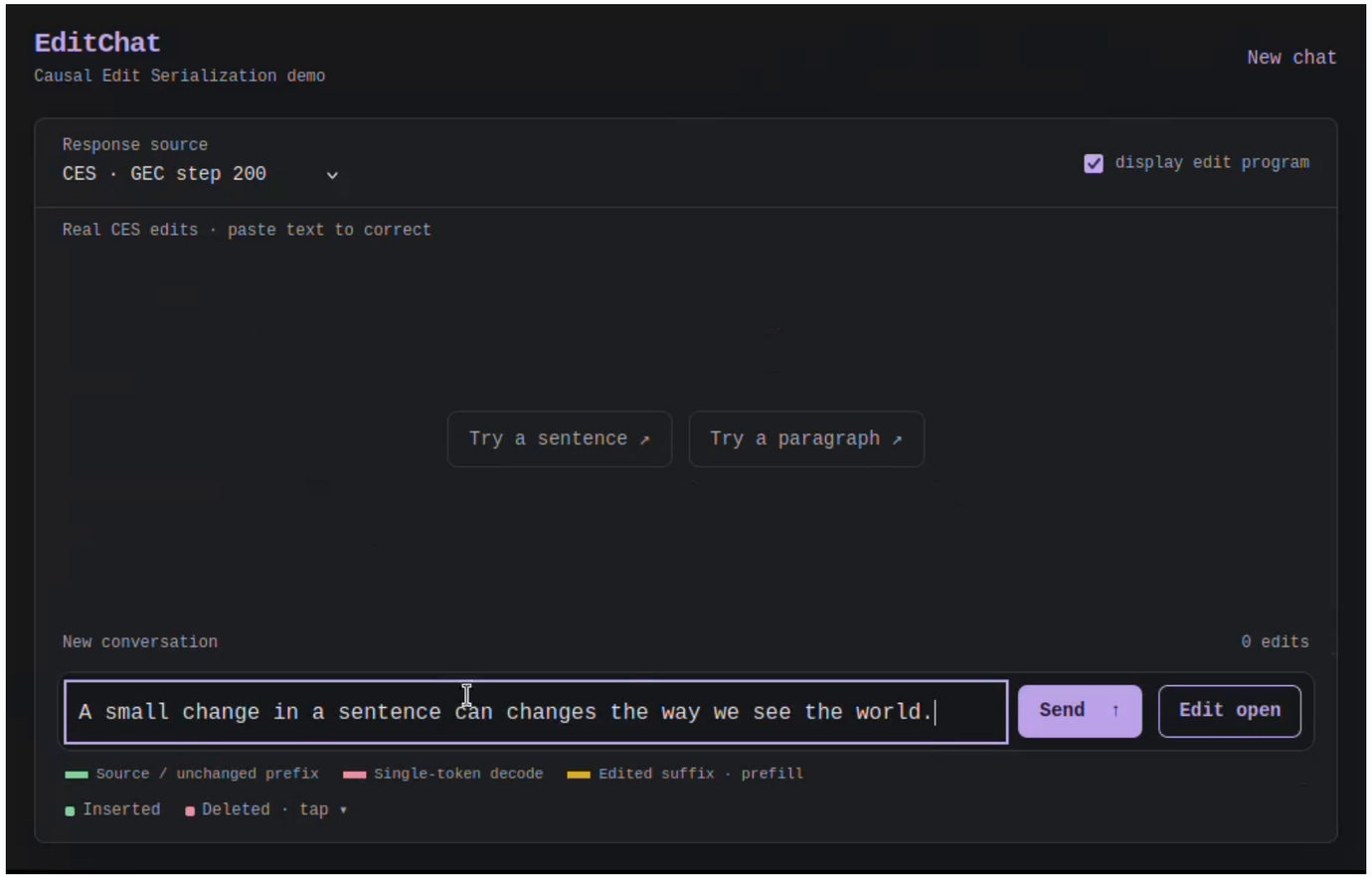
Net judged improvement



Paper · Figure 3 · all-input denominators · net-comparison intervals: 95% bootstrap

Real checkpoint: example 1

1. Input text | 2. Source tokens | 3. Edit program | 4. Apply the edit



Before submission

A small change in a sentence can changes the way we see the world.

The original input, before requesting an edit program.

Recorded checkpoint · CES / GEC step 200 · source video 16.900 s

Real checkpoint: example 1

1. Input text
2. Source tokens
3. Edit program
4. Apply the edit



The source is in context

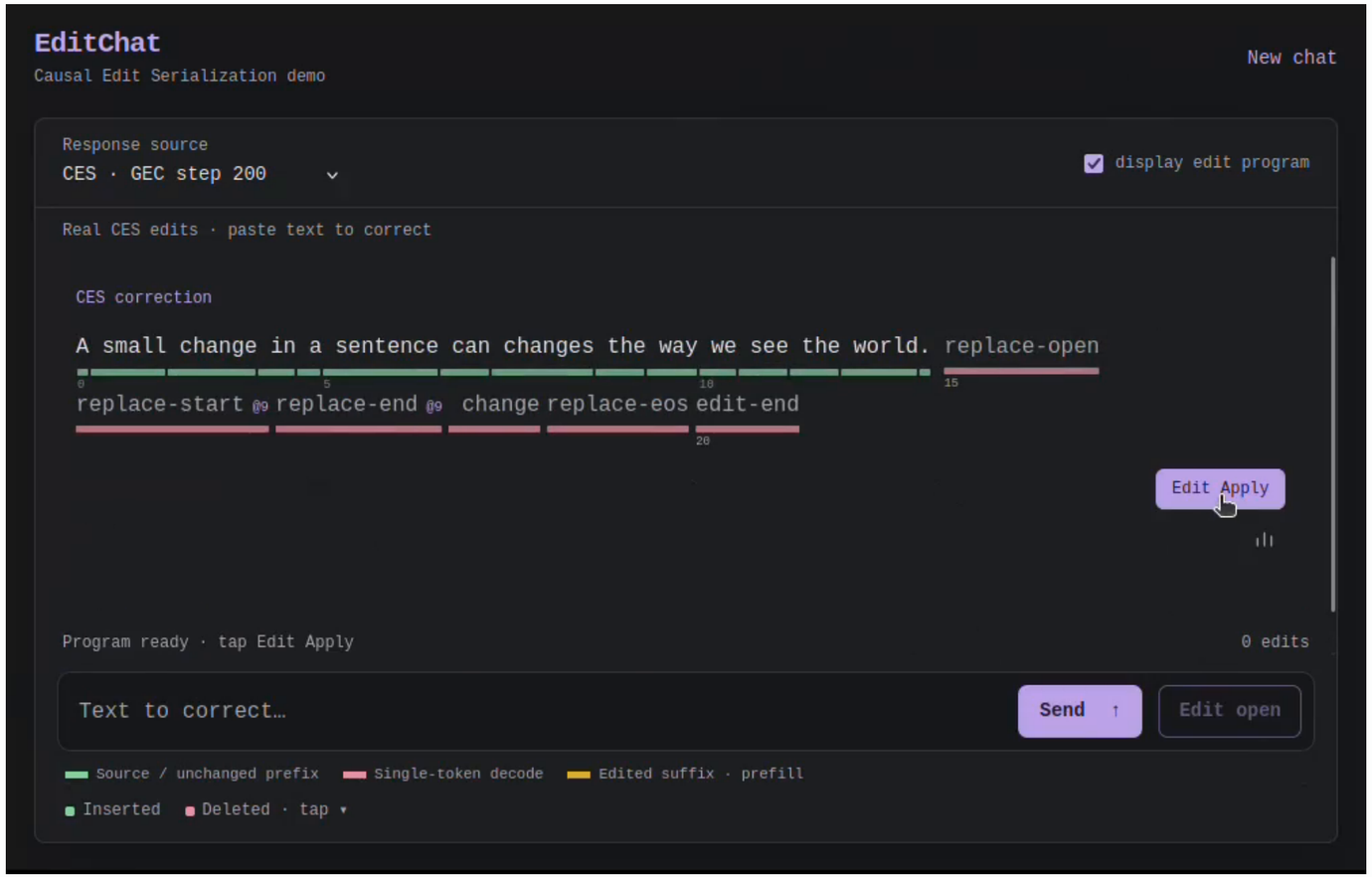
A small change in a sentence can changes the way we see the world.

The source appears before any edit instructions. Token bars mark the context positions.

Recorded checkpoint · CES / GEC step 200 · source video 17.783 s

Real checkpoint: example 1

1. Input text
2. Source tokens
3. Edit program
4. Apply the edit



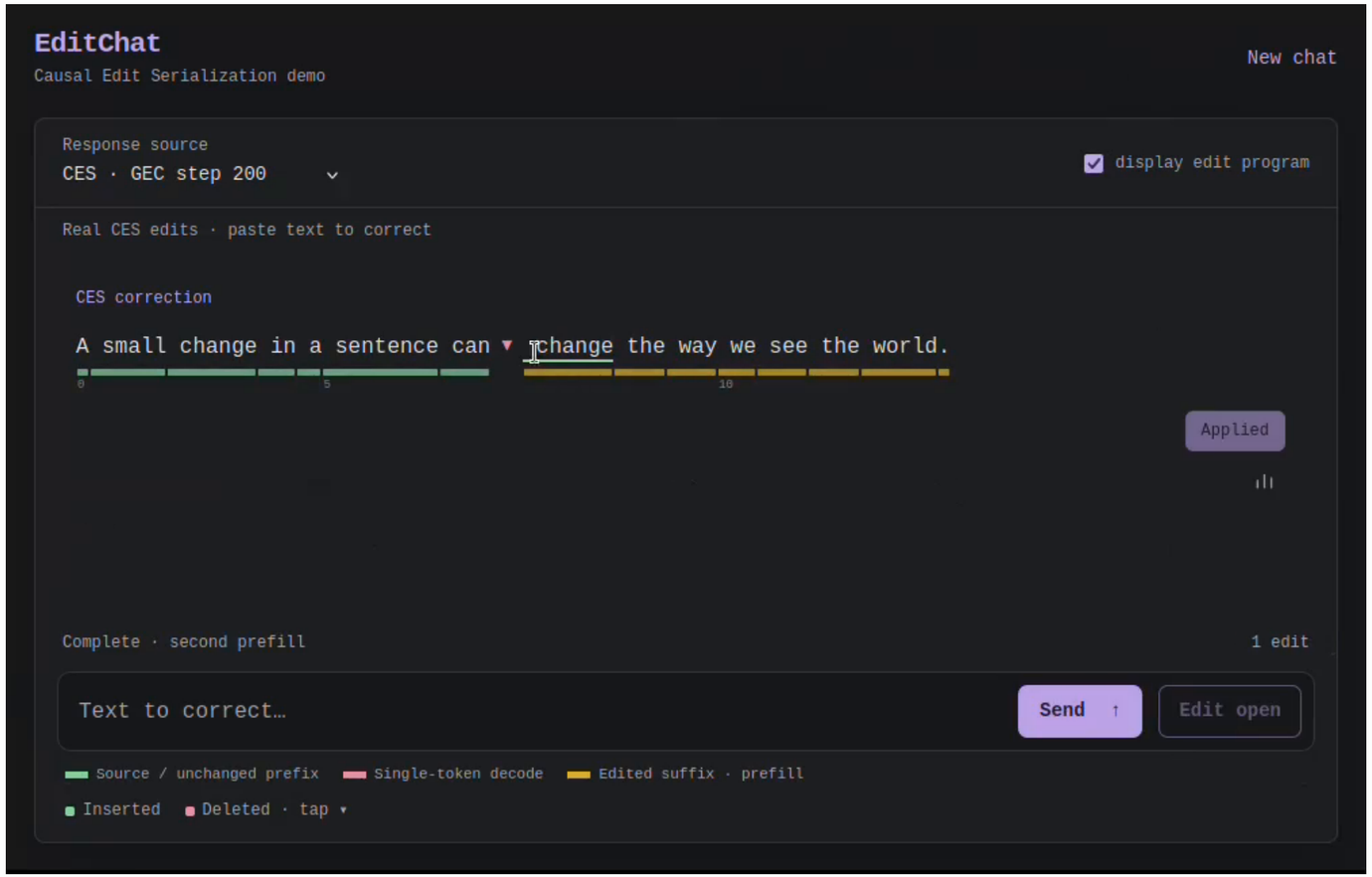
The predicted program

```
replace-open
replace-start @9
replace-end   @9
change
replace-eos
edit-end
```

Replace the token at source address @9 with “change”.

Real checkpoint: example 1

1. Input text | 2. Source tokens | 3. Edit program | **4. Apply the edit**



The revised text

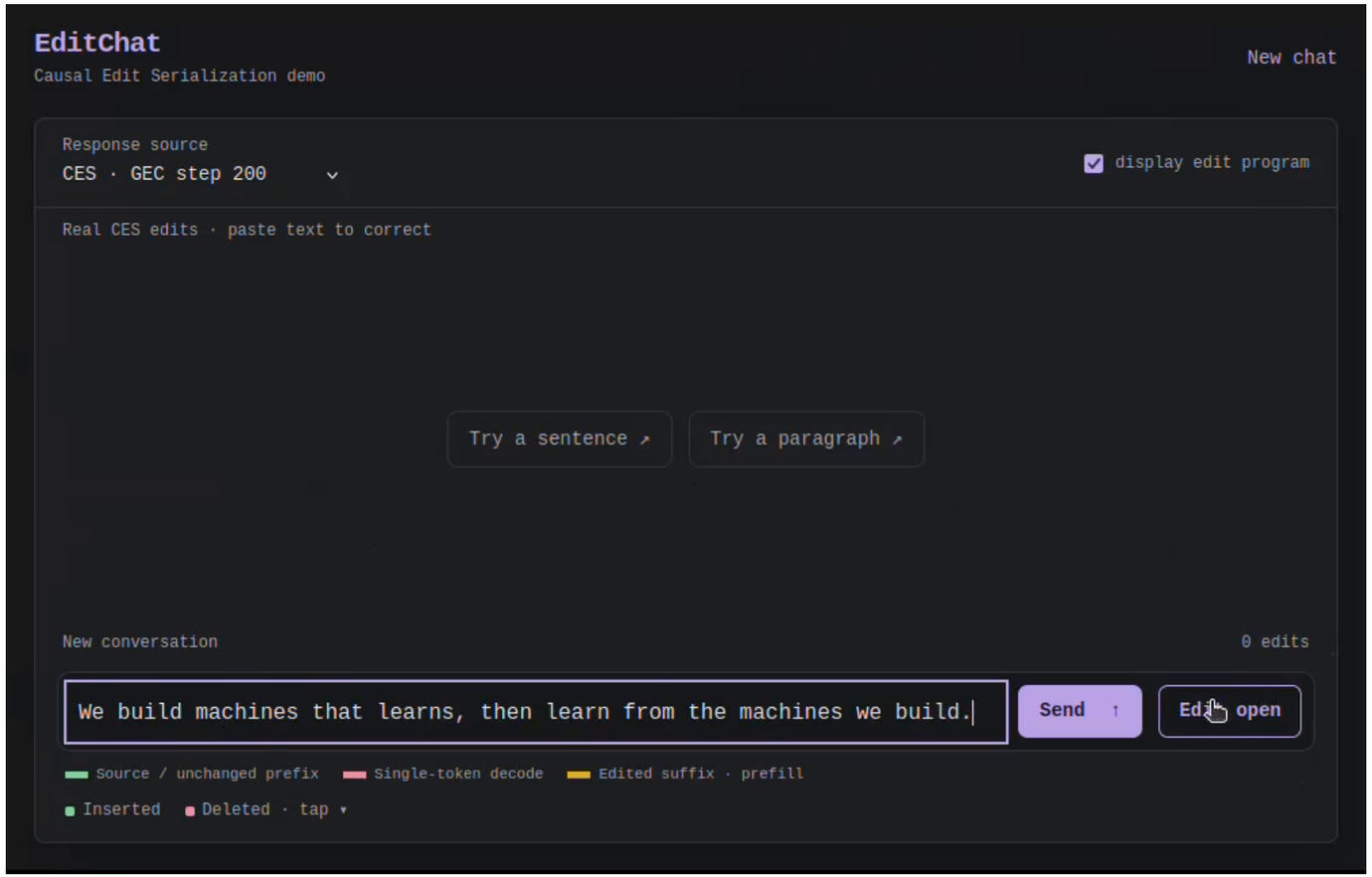
A small change in a sentence can change the way we see the world.

changes → change

The unchanged prefix is retained. The yellow suffix is recomputed after applying the edit.

Real checkpoint: example 2

1. Input text | 2. Source tokens | 3. Edit program | 4. Apply the edit



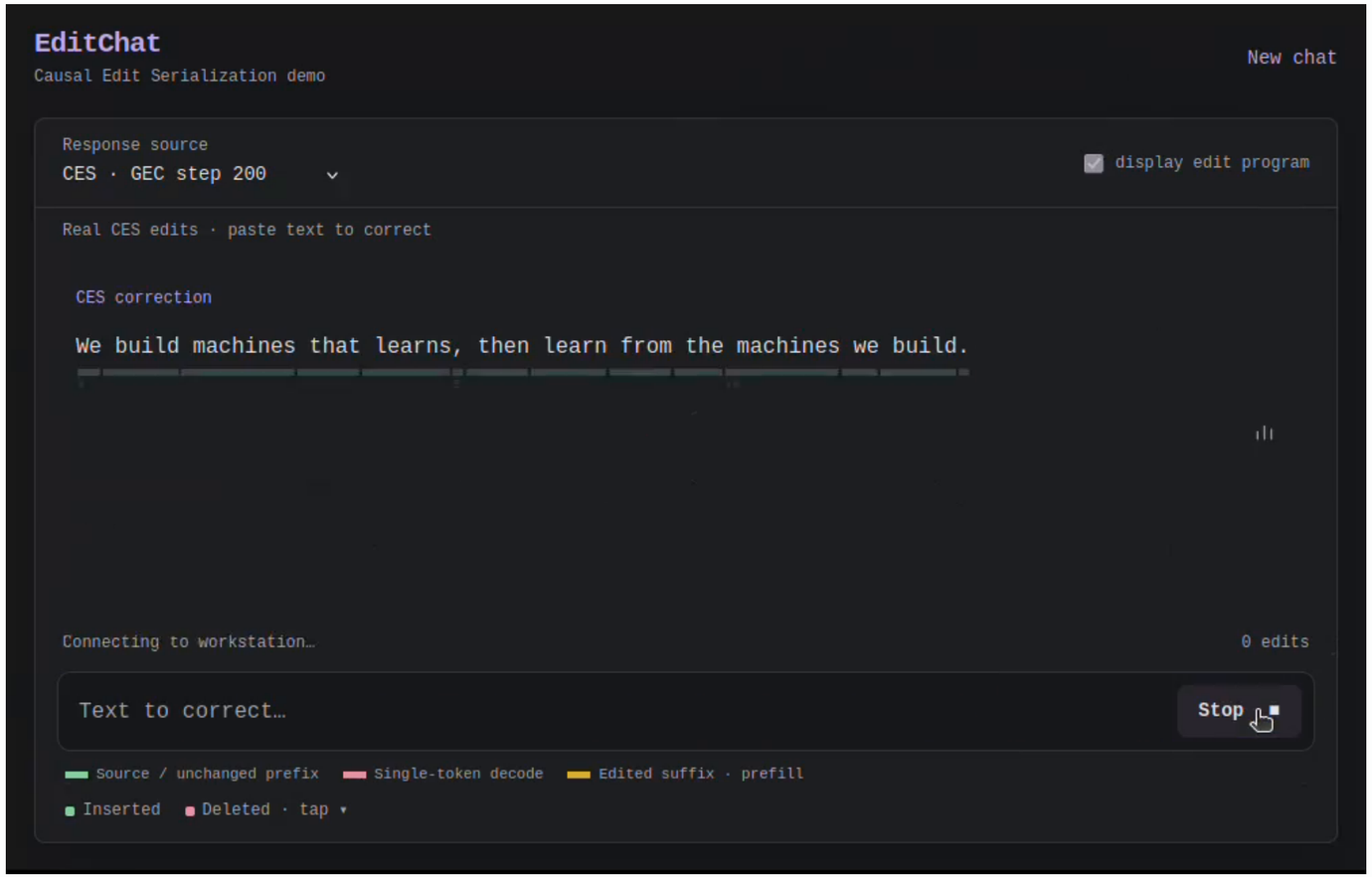
Before submission

We build machines that learns, then learn from the machines we build.

The original input, before requesting an edit program.

Real checkpoint: example 2

1. Input text
2. Source tokens
3. Edit program
4. Apply the edit



The source is in context

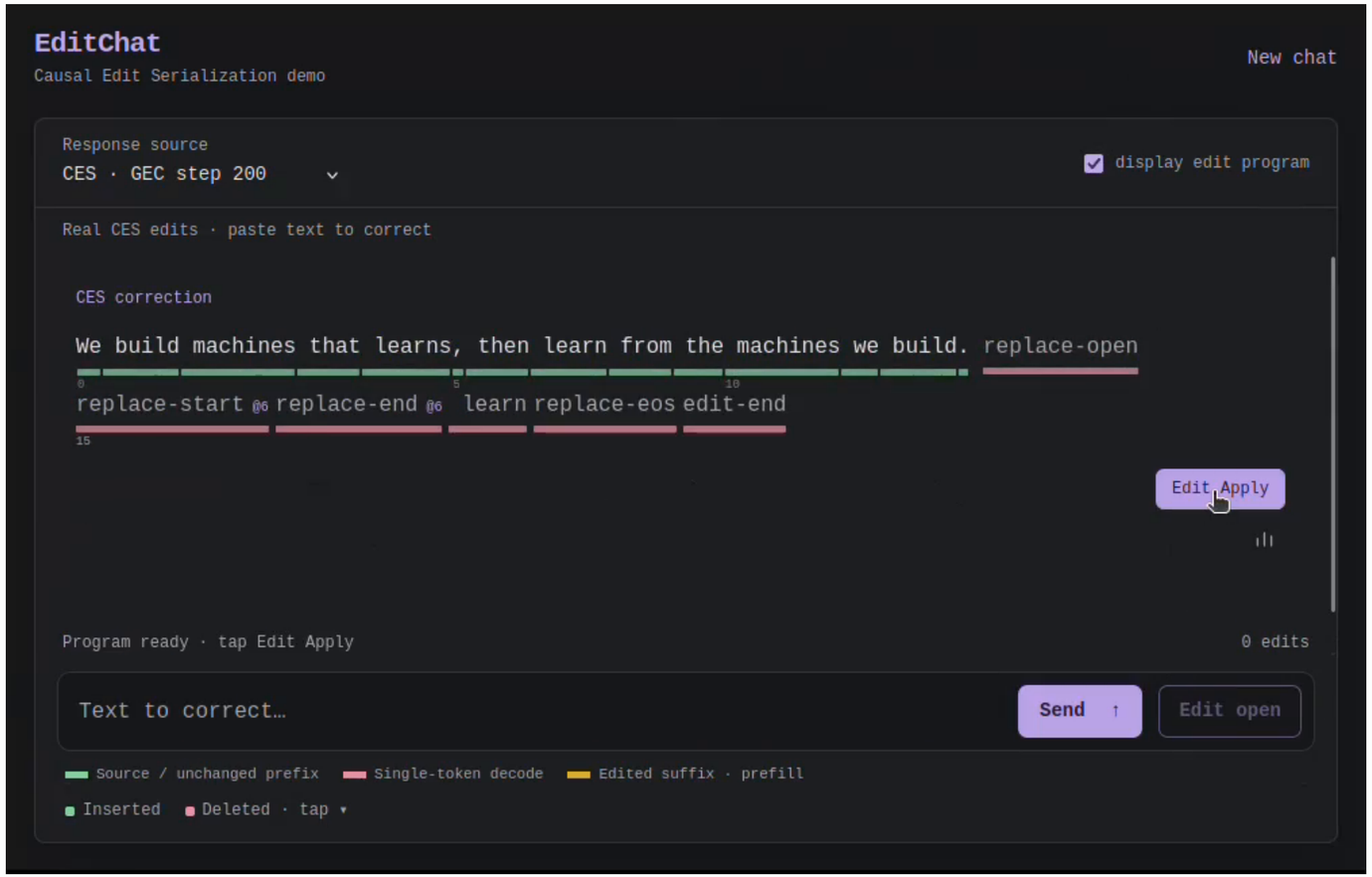
We build machines that learns, then learn from the machines we build.

The source appears before any edit instructions. Token bars mark the context positions.

Recorded checkpoint · CES / GEC step 200 · source video 27.783 s

Real checkpoint: example 2

1. Input text
2. Source tokens
3. Edit program
4. Apply the edit



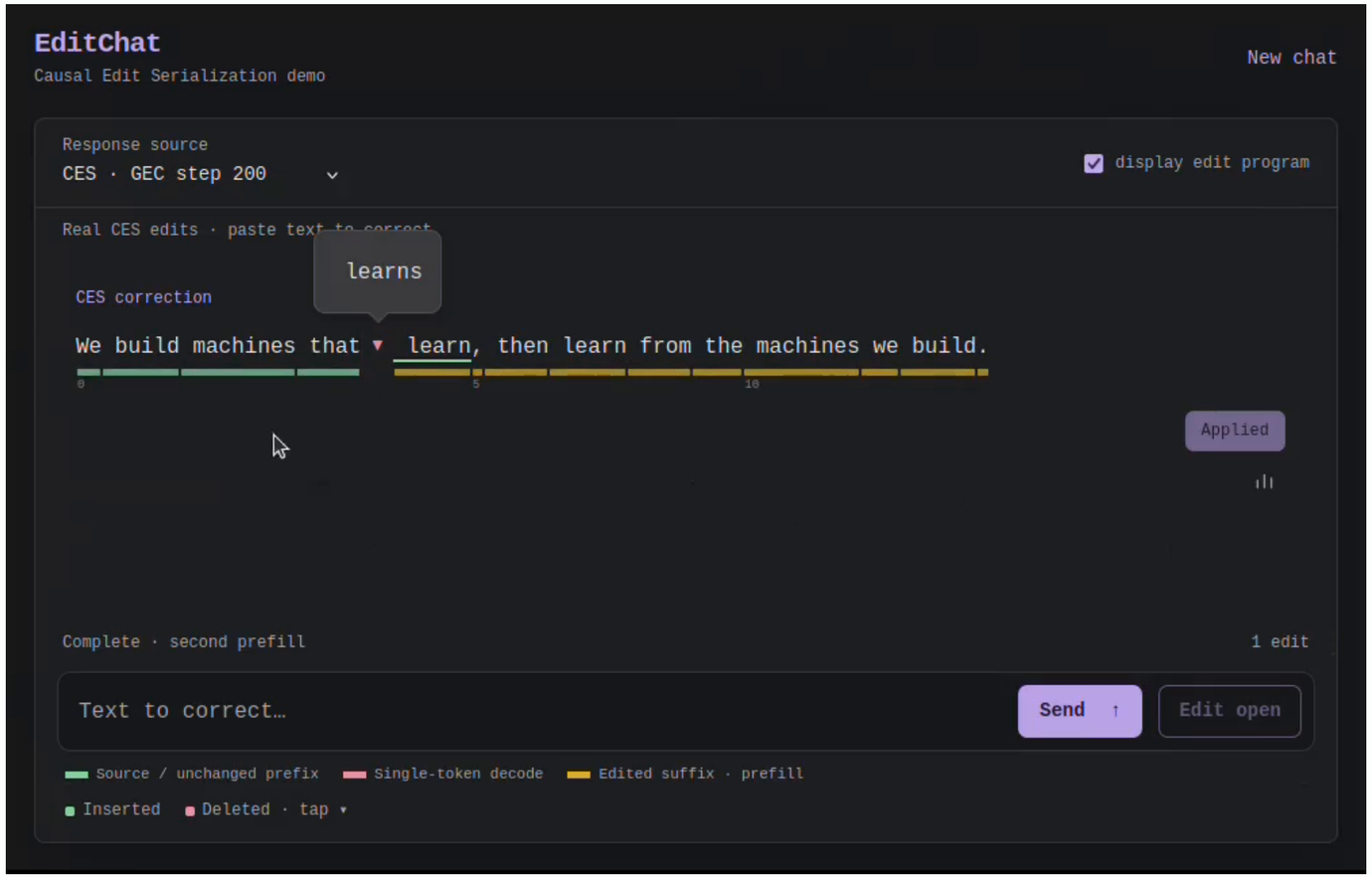
The predicted program

```
replace-open
replace-start @6
replace-end   @6
learn
replace-eos
edit-end
```

Replace the token at source address @6 with “learn”.

Real checkpoint: example 2

1. Input text | 2. Source tokens | 3. Edit program | **4. Apply the edit**



The revised text

We build machines that learn, then learn from the machines we build.

learns → learn

The unchanged prefix is retained. The yellow suffix is recomputed after applying the edit.

TAKEAWAY

Generate left to right. Revise by pointing back.

A causal stream can contain both text and executable, pointer-addressed edits.

Operation + source addresses + any new text

Paper	arxiv.org/abs/2610.22820
Code	github.com/Louiii/CES
Data / models	huggingface.co/datasets/lorob/CES