ICML 2026

Autocomplete, but for spreadsheets.

Predicting a user's next action from their editing history — and a benchmark that scores it the way users feel it.

A Benchmark and Framework for Evaluating Next Action Predictions in Spreadsheets

Tejas Agrawal · Vu Le · Sumit Gulwani · Gust Verbruggen

PROSE @ Microsoft

napeval.github.io

ICML International Conference On Machine Learning

Microsoft

01

The task

Turning **spreadsheet authoring** into a next-action prediction problem.

NEXT ACTION PREDICTION

You finish a line of code. Why not a row of a sheet?

Code editor — autocompletes

```
# price × qty, with tax
def total(price, qty):
  return price * qty * (1+tax)  Tab⇥
```

SAME IDEA

Spreadsheet — predictor fills in

Tax rate	8%	cell **F1** — referenced by **F1**

Price	Qty	Tax
100	2	=B2*C2*F1
60	1	=B3*C3*F1
40	3	=B4*C4*F1

Code editors finish your next line. Spreadsheets — used by hundreds of millions — suggest almost nothing. We cast authoring as sequence modeling: predict the next action from the editing history.

WHAT IS AN ACTION?

One deliberate edit — a single step of user intent.

An action is one deliberate edit a user makes while building something — a single step of intent. The idea isn't unique to spreadsheets: you make them in any tool you author in.

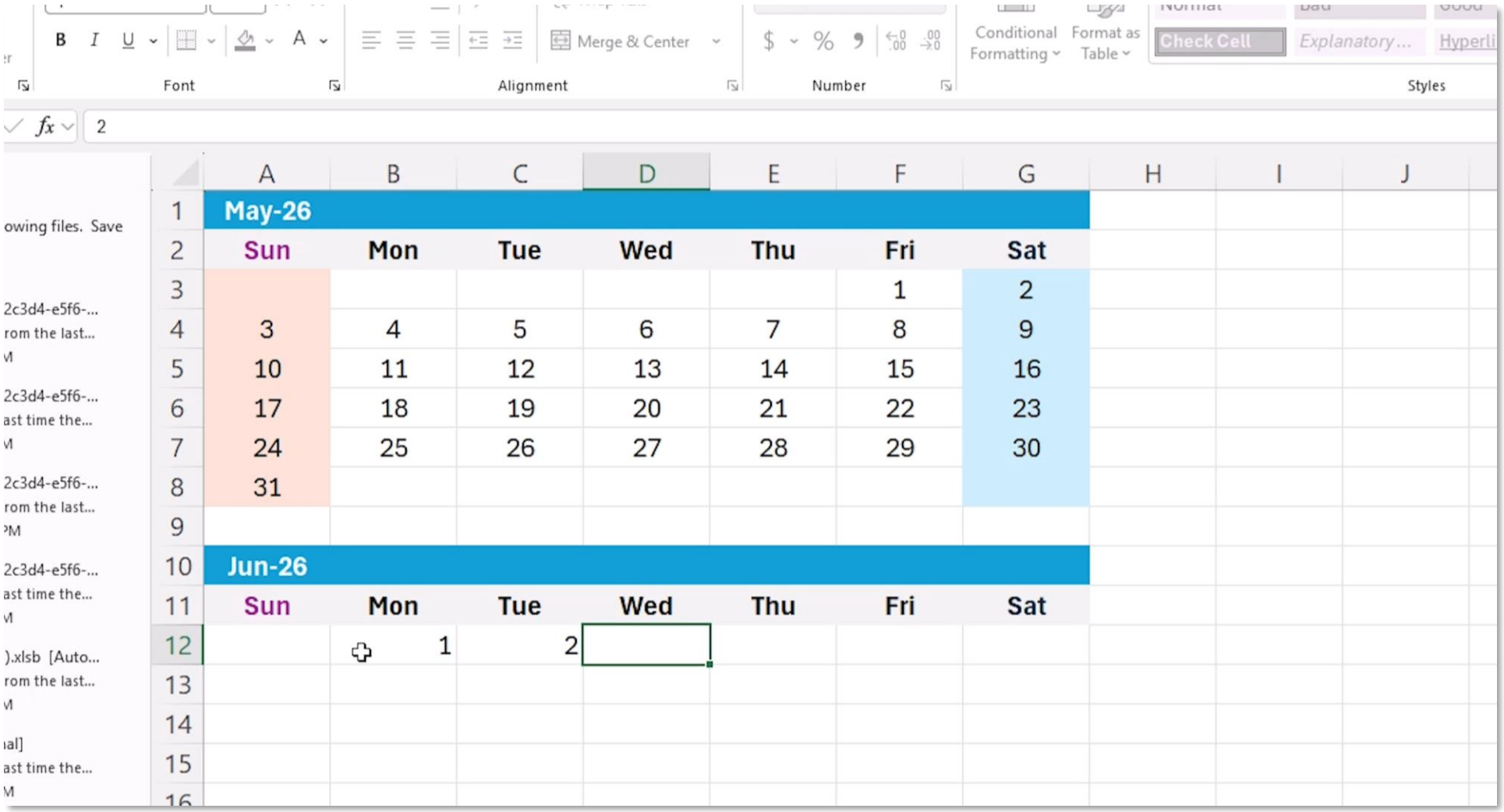
Docs — apply a heading | Code — commit a line | Design — move a layer → **we study them in spreadsheets**

In a spreadsheet, an action is one of:

- Type a value
- Write a formula
- Bold & font
- Number format
- Add a border
- Fill color
- Merge cells
- Paste / autofill

SEE IT LIVE

A real add-in, finishing sheets as you type.



02

Evaluating it right

Static accuracy **lies**. We score the way an autocomplete actually feels in use.

WHY IT'S HARD

Two challenges — and how we tackle each.

1 · No edit histories

Public spreadsheet corpora store only the **final file** — never the sequence of edits that built it.

→ **We reconstruct version histories.** 52 human-validated build trajectories from public sheets.

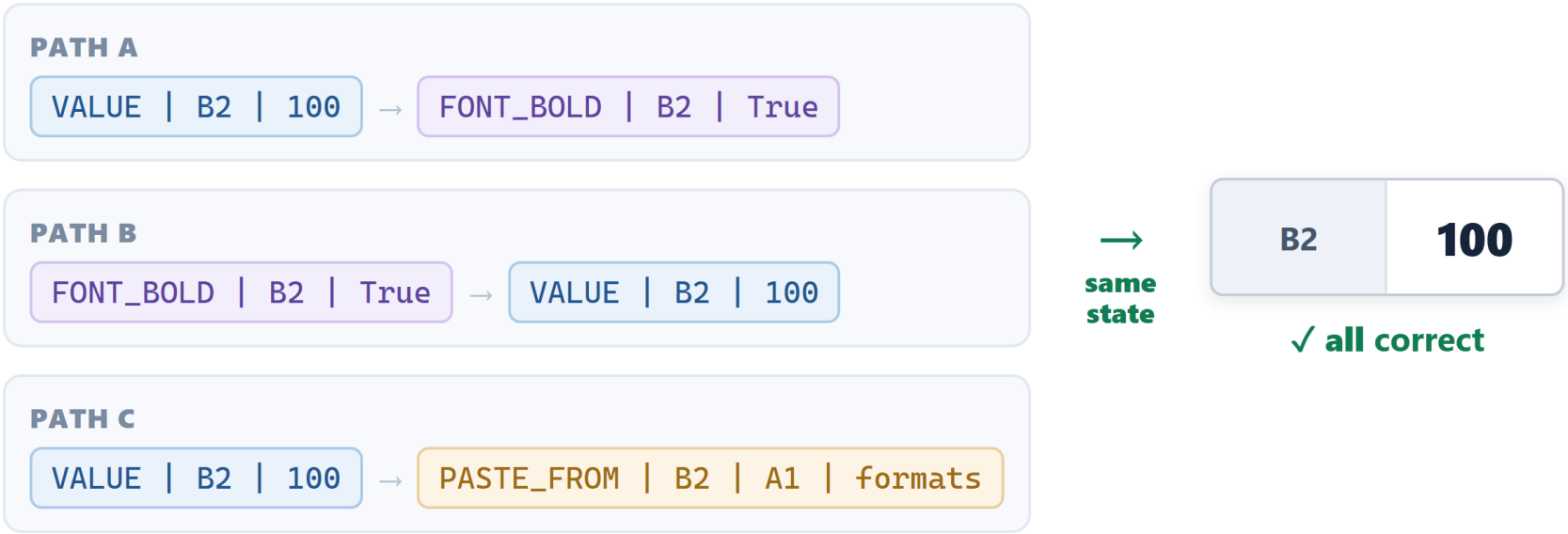
2 · Composite action space

Edits are **spatial, temporal & multi-cell**, reachable in many orders, with **no clear trigger** for when to predict.

→ **So evaluation is the hard part** — a composite, state-based, **online** benchmark that scores the resulting sheet, live.

STATE-BASED EVAL

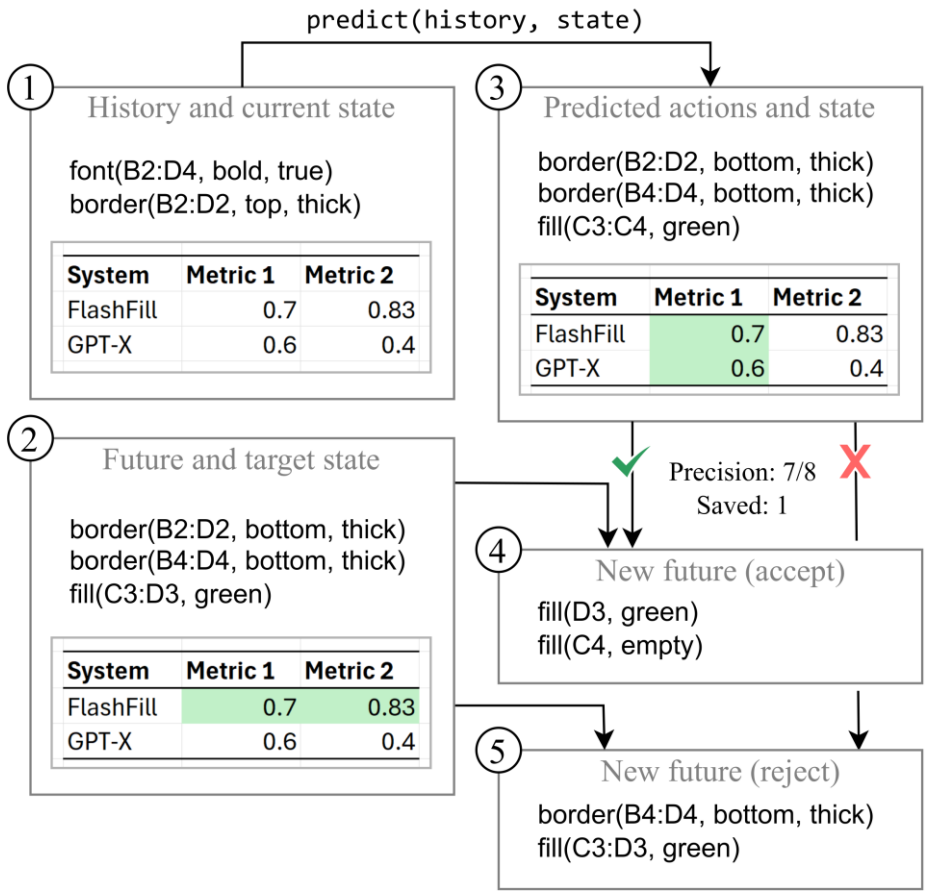
One checkable string per edit — scored on the state.



We grade the resulting **(cell, property) state**, not the keystrokes — different op-orders that reach the same sheet are **all correct.**

ONLINE, NOT OFFLINE

Each acceptance rewrites the future.



Predict at every step → accept or reject; on accept the **future is rewritten** to still reach the target.

Offline lies: A. errors **compound** · B. the model must **repair its own mistakes** · C. it can't re-score the same easy actions in isolation.

So we replay every trajectory **live** and measure **User Actions Saved (UAS).**

WHAT WE REPORT

Four state-level metrics — never just one.

UAS

PRIMARY

User Actions Saved

% of manual actions removed over a sheet — **the number a user feels.**

AR

Acceptance Rate

Of the predictions we offered, how many were **accepted.**

PREC

Precision

Of the predicted edits that landed — **how much was wrong?**

pCov

Predictable Coverage

Of the **predictable** edits, how many we actually caught.

All at the **(cell, property) state level** — and never UAS alone: a model can save actions while being annoying. **Wrong suggestions have a real cost.**

03

The Benchmark

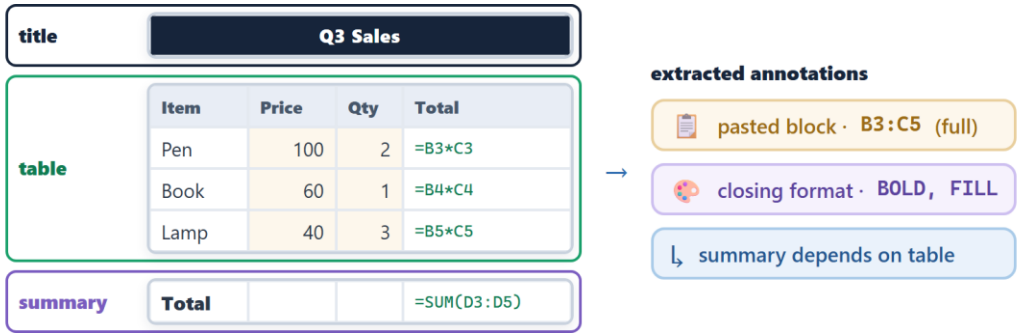
Where the edit histories come from — and what 52 real sheets actually look like.

Reverse-engineering the build order

1 Annotate the sheet

`vision model · regions & pastes`

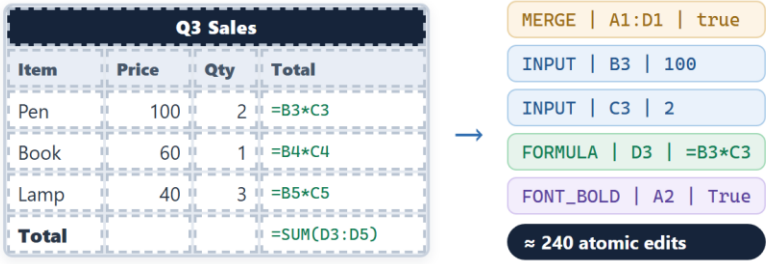
A vision model labels **regions** and marks which ranges were entered as **one action** — pasted blocks, fills, formula drags — plus **dependencies**. These labels decide what may be grouped.



2 Explode into atomic edits

`read → one edit per cell × property`

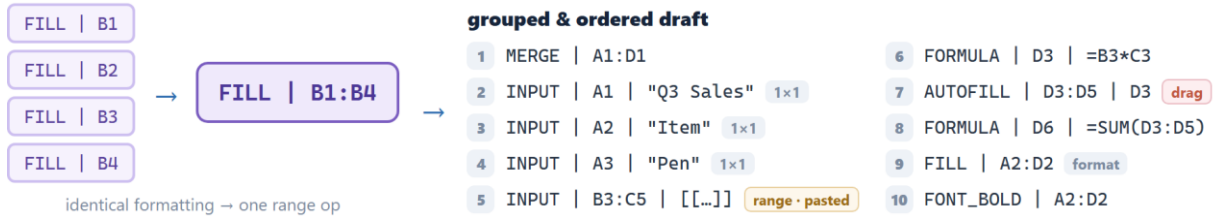
Read the finished sheet into a simple **state map**, then split it into **one op per cell & property**. A tiny sheet becomes **hundreds** of single-cell edits — faithful, but nothing like how a person types.



3 Group & order edits

`annotation-guided + auto-fold`

Inputs merge **only where a label marks a paste**; matching **formats** and **formula drags** fold on their own.



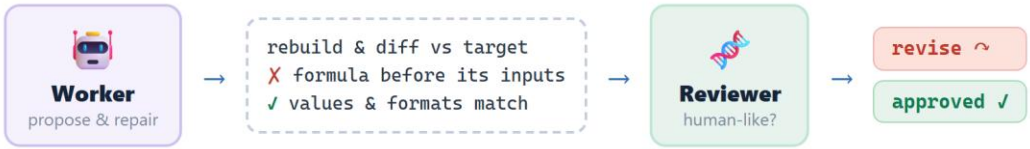
BENCHMARK · BUILDING THE DATA

Validate, and make it human

4 AI worker & reviewer

`propose ⇄ review loop`

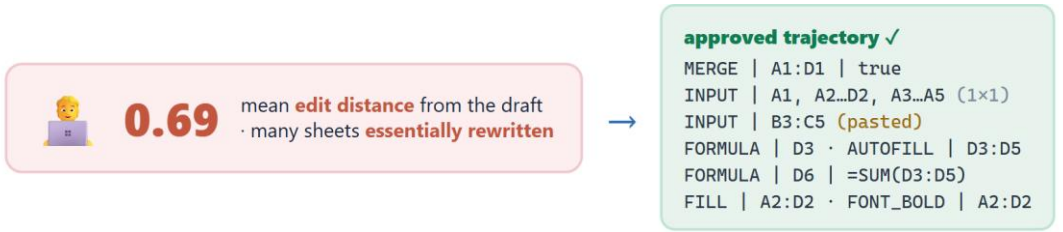
A **worker** rebuilds the sheet, diffs it against the **true target** and repairs mismatches. A **reviewer** checks it matches **and** reads human. Loop until **approved**.



5 Human validation

`annotator pass`

An annotator reviews and **reorders** the draft into a genuine build order, then approves.

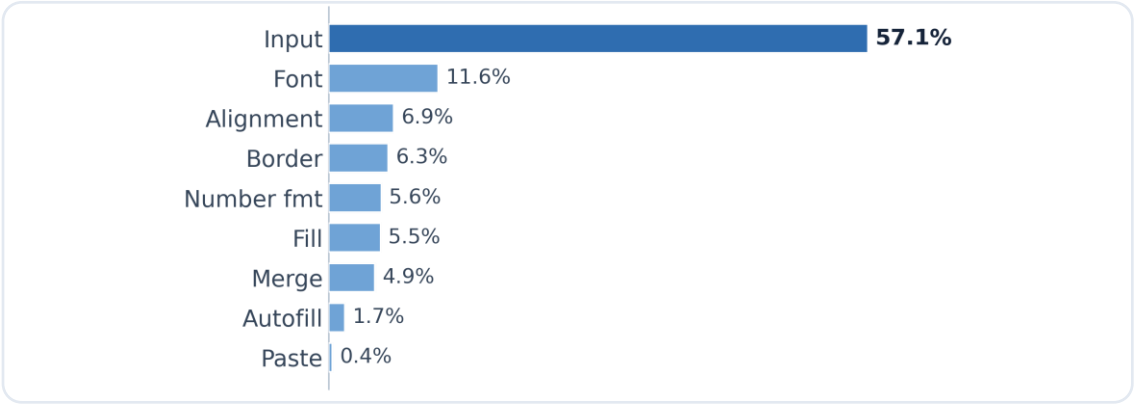


🎉 **One training trajectory — an ordered list of symbolic edits we can replay step-by-step. × 52 sheets = the benchmark.**

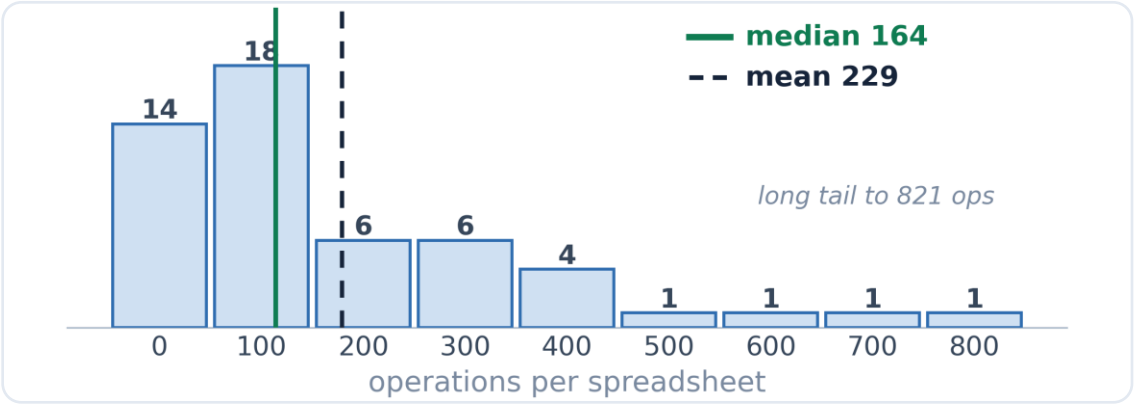
THE CORPUS

Small, real, and human-built.

Action types



Sequence length



52	~12K	229	9
real workbooks	user actions	mean ops / sheet	operation types

Reconstructed from public workbooks — **35–821 ops** each. Input values dominate (~57%); the rest is **formatting & structure**.

HOW MUCH IS PREDICTABLE?

68% — an empirical ceiling.

Oracle — 4 frontier reasoners, ×2 generations

- Claude Opus 4.6
- Claude Opus 4.7
- GPT-5.4
- GPT-5.5

⋃ union of correct (cell, property) pairs

68% of all ground-truth edits are **recoverable from history alone**

median **66%** per sheet · **44 of 52** sheets above 50% — a theoretical ceiling for online prediction.

> Most spreadsheet actions **are** predictable from context — and today's solvers still sit well below this ceiling.

TWO PREDICTION MODES

Predict a block, or one op and chain.

User action Accepted Rejected Empty (abstain)

(a) Multi-action k ≥ 1 · one block per query

A model emits a **block of 0+ ops** per query; the cursor advances by accepted ops, then — **accepted or not** — waits one user action (**stride**) before the next query.

(b) Single-action k = 1 · re-predict

> One op per query; on each accept the solver is **immediately re-queried** — chaining until a reject or empty (**abstention**), then it waits one action. For solvers that can't yet decide when to stop (fine-tuned SLMs, classical) — the loop controls termination, like pressing **Tab**.

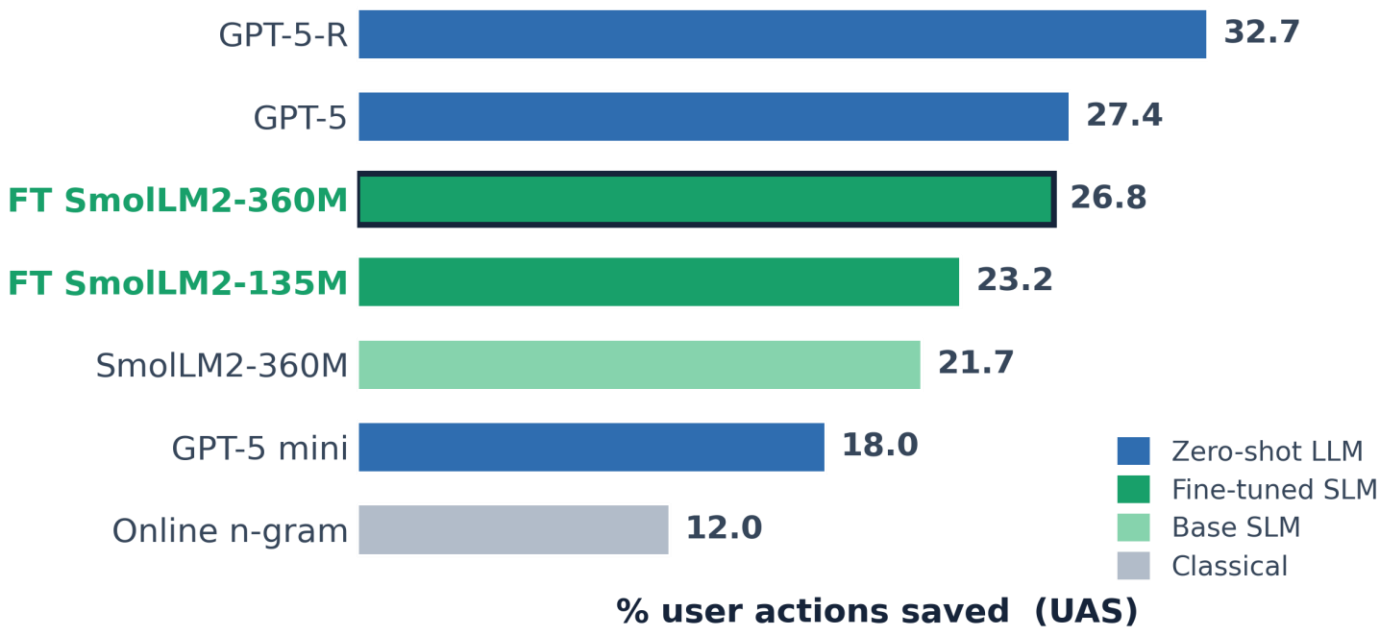
04

What we found

The task is **learnable** — and a tiny fine-tuned model goes toe-to-toe with a frontier LLM.

LEADERBOARD

A 360M model ≈ GPT-5 — ~1000× smaller.



Single-action UAS. Reasoning GPT-5 leads (32.7); a **fine-tuned 360M SLM (26.8)** ties vanilla GPT-5 (27.4). Classical models stay in single digits.

FULL RESULTS

Single & multi-action — and a +5 UAS fine-tuning lift.

MODEL	SINGLE UAS	MULTI UAS
ZERO-SHOT LLMS		
GPT-5 Reasoning	**32.7**	**26.6**
GPT-5 Mini Reasoning	28.2	21.3
GPT-5	27.4	22.3
GPT-5 Mini	18.0	20.1
SLMS — BASE → FINE-TUNED		
SmolLM2-360M	21.7	—
FT-SmolLM2-360M	**(5.1 ▲) 26.8**	—
SmolLM2-135M	18.3	—
FT-SmolLM2-135M	**(4.9 ▲)** 23.2	—
CLASSICAL SEQUENCE MODELS		
Online n-gram	12.0	—
LSTM	5.7	—
XGBoost	2.9	—

WHAT DRIVES UAS

Cheap triggers & more context beat model size.

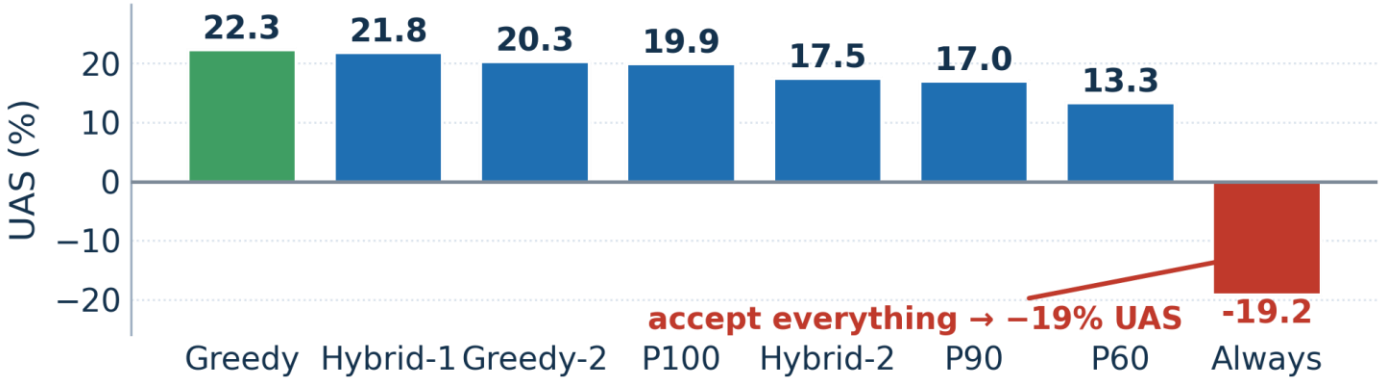
PARAM	VALUE	SINGLE	MULTI
PREDICTION STRIDE — HOW OFTEN WE PREDICT			
stride	1 — every step	**27.4**	**22.3**
	2	22.6	19.5
	4	16.8	14.7
	8	10.6	9.8
CONTEXT WINDOW — HOW MUCH HISTORY			
context	8	19.9	16.2
	32	27.4	22.3
	128	**30.0**	**27.6**
	512	30.8	26.2

Predict every step. Stride 1→8 craters UAS 27→11 — trigger frequency is the biggest lever.

More history helps. Context 8→128 lifts UAS 20→30; beyond that it saturates.

KNOWING WHEN TO STOP

Always-predict is worse than nothing.



Always-predict → −19% UAS — constant wrong suggestions cost more than they save.

ACCEPTANCE POLICY	UAS
Greedy — save ≥ 1	**22.3**
Hybrid-1 — p≥.9, st≥1	21.8
Greedy-2 — save ≥ 2	20.3
P100 — p = 1	19.9
P90 — p ≥ .9	17.0
Always predict	**−19.2**

FUTURE WORK

Where this goes next.

- **Abstention in small models.** Give SLMs an internal stop signal so they work in the practical multi-action setting — no external oracle.
- **Stronger predictors.** Specialized training & cheap triggers to push past today's frontier-LLM ceiling.
- **Beyond spreadsheets.** The same online, state-based evaluation extends to slides, design tools, and other modeless authoring software.
- **A more expressive benchmark.** Fuzzy match-based evaluators for assessing subjective correctness instead of an exact match
- **Predicting intent, not just actions.** Use the history to anticipate the user's next high-level goal — formatting a table, adding a chart — rather than the exact operation.

TAKEAWAYS

Three things to walk away with.

- **A new task & benchmark.**
  68% of actions are predictable — the ceiling is real.
- **Online eval is non-negotiable.** We score User Actions Saved with abstention — the way autocomplete actually feels.
- **Don’t need a multi-trillion param LLM.** A fine-tuned 360M model rivals GPT-5 — cheap, local, deployable.

THANK YOU

Let's make sheets finish themselves.

Data · leaderboard · code — all open.

napeval.github.io

ICML International Conference On Machine Learning

Microsoft

Tejas Agrawal · Vu Le · Sumit Gulwani · Gust Verbruggen

PROSE @ Microsoft

APPENDIX

Example: Seed a few cells — predictor finishes the rest.

	A	B	C	D	E
1	Country	Capital	Currency	ISO	Continent
2	United States	Washington DC	US Dollar	USA	N. America
3	France	Paris	Euro	FRA	Europe
4	Japan	Tokyo	Yen	JPN	Asia
5	Brazil	Brasília	Real	BRA	S. America
6	Egypt	Cairo	Egypt. Pound	EGY	Africa
7	Australia	Canberra	AUD	AUS	Oceania
8	Germany	Berlin	Euro	DEU	Europe
9	Mexico	Mexico City	Peso	MEX	N. America

	A	B	C	D	E	F
1	Time	Event			Time	Today
2	9:00 - 10:30	Standup			9:00	Standup
3	11:00 - 12:00	Code review			9:30	
4	13:00 - 14:30	Design sync			10:00	
5	15:00 - 16:00	1:1 with Carol			10:30	
6	16:30 - 18:00	Deep work			11:00	Code review
7					11:30	
8					12:00	
9					12:30	
10					13:00	
11					13:30	Design sync
12					14:00	
13					14:30	
14					15:00	
15					15:30	1:1 with Carol
16					16:00	
17					16:30	
18					17:00	Deep work
19					17:30	
20					18:00	

List the countries → it fills every column — capitals, currencies, ISO, continents.

Drop one event → it lays out the whole day — right slots, colours, borders.

APPENDIX

Soft correctness — a one-line swap.

Correctness is computed on the resulting **(cell, property) state** — so different **order**, **range-vs-cell**, and sequencing already count as **equivalent**. Only the per-property check is exact.

TODAY · EXACT

```
match = pred[c][p] == gt[c][p]
```

→

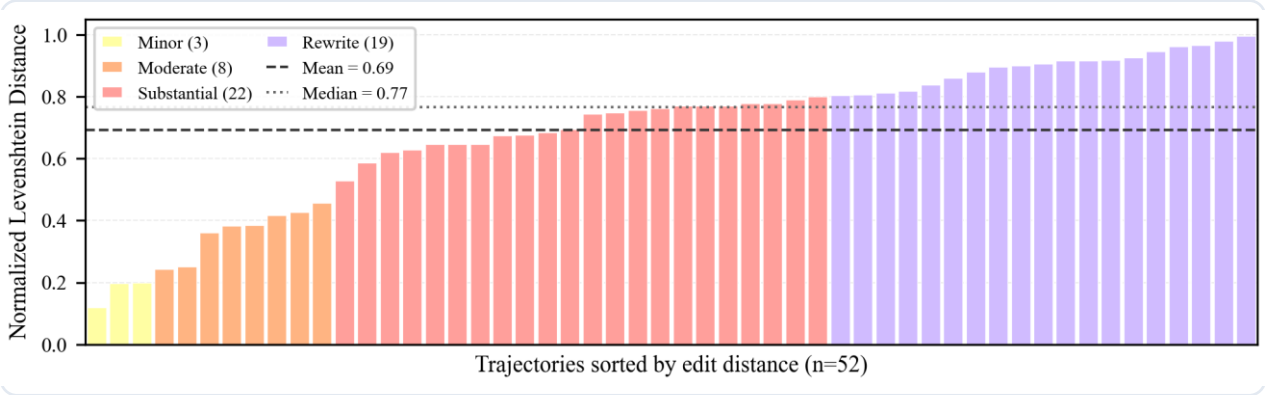
SOFT · ONE-LINE SWAP

```
match = equivalent(pred[c][p], gt[c][p])
```

Swap one operator — **everything downstream is unchanged** (online loop, UAS, acceptance heuristics). "Soft" could be semantic (USA ≈ United States), fuzzy values, tolerant formats, or an **LLM judge**. Because we match exactly today, every number is a **conservative lower bound.**

APPENDIX

Synthetic draft — human trajectory.



Annotators change a **lot** — mean edit distance **0.69**, median **0.77**, and **19 of 52** are near-total rewrites. The trajectories we ship are **human-authored.**

So why generate a synthetic draft first, if it gets rewritten?

1. **Less annotation effort.** Editing a plausible draft is far cheaper than hand-building every trajectory from scratch.

2. **Variant starting points.** Many valid build orders exist for one sheet; a lone annotator defaults to their own style. Stochastic parameter sampling seeds different drafts — injecting that variation while still cutting effort.

APPENDIX

Result Tables – Single-action

MODEL	UAS↑	AR↑	PREC↑	PCOV↑
Zero-shot LLMs				
GPT-5-R	32.7	29.4	41.6	24.8
GPT-5-R mini	28.2	25.5	37.0	20.9
GPT-5	27.4	30.9	44.8	20.7
GPT-5 mini	18.0	16.8	21.9	10.7
SLMs (base)				
SmolLM2-360M	21.7	22.3	29.7	9.6
SmolLM2-135M	18.3	19.0	24.8	8.9
SLMs (finetuned)				
FT-SmolLM2-360M	26.8	26.8	33.7	13.7
FT-SmolLM2-135M	23.2	23.1	30.6	13.0
Classical sequence models				
LSTM	5.7	5.5	12.4	2.4
Trained n-gram	3.8	3.9	11.9	0.7
XGBoost	2.9	2.3	6.5	1.0
Online n-gram	12.0	14.7	20.4	11.1

PARAM	VALUE	UAS↑	AR↑	PREC↑	PCOV↑
Stride (s)	1*	27.4	30.9	44.8	20.7
	2	22.6	36.5	48.4	15.9
	4	16.8	42.3	53.2	9.4
	8	10.6	43.7	55.1	7.1
Context (c)	8	19.9	24.1	39.7	13.6
	32*	27.4	30.9	44.8	20.7
	128	30.0	32.5	47.8	28.5
	512	30.8	33.7	47.9	31.0
Shortening	on*	27.4	30.9	44.8	20.7
	off	27.7	31.2	44.3	21.4
Repredict	on*	27.4	30.9	44.8	20.7
	off	20.3	30.6	44.1	15.2

APPENDIX

Result Tables – Multi-action

MODEL	UAS↑	AR↑	PREC↑	PCOV↑
GPT-5-R	26.6	13.3	27.8	30.7
GPT-5	22.3	20.0	36.0	19.6
GPT-5-R mini	21.3	9.6	23.2	24.3
GPT-5 mini	20.1	10.1	19.0	17.3

HEUR.	RULE	UAS↑	AR↑	PREC↑	PCOV↑
GREEDY	$\ell \geq 1$	22.3	20.0	36.0	19.6
HYBRID-1	$p \geq .9$, $\ell \geq 1$	21.8	16.6	39.4	17.3
GREEDY-2	$\ell \geq 2$	20.3	10.5	37.6	15.9
HYBRID-2	$p = 1$, $\ell \geq 2$	17.5	7.9	40.1	10.7
P100	$p = 1$	19.9	21.8	38.2	20.1
P90	$p \geq .9$	17.0	23.3	36.0	25.5
P60	$p \geq .6$	13.3	26.9	32.9	33.0
ALWAYS	—	−19.2	100.0	9.3	8.1

PARAM	VALUE	UAS↑	AR↑	PREC↑	PCOV↑
Stride (s)	1*	22.3	20.0	36.0	19.6
	2	19.5	24.4	39.4	16.0
	4	14.7	33.4	45.1	12.2
	8	9.8	36.5	49.4	6.8
Context (c)	8	16.2	17.5	33.8	12.3
	32*	22.3	20.0	36.0	19.6
	128	27.6	19.7	37.3	30.0
	512	26.2	19.0	35.9	32.0
	2048	27.4	19.4	36.6	33.0
Shortening	on*	22.3	20.0	36.0	19.6
	off	24.2	21.5	38.4	21.4
Num ops (m)	1	20.3	30.6	44.1	15.2
	4	20.8	23.1	39.2	19.4
	8	22.1	19.4	36.1	18.9
	16	21.8	20.6	36.2	20.2
	∞*	22.3	20.0	36.0	19.6

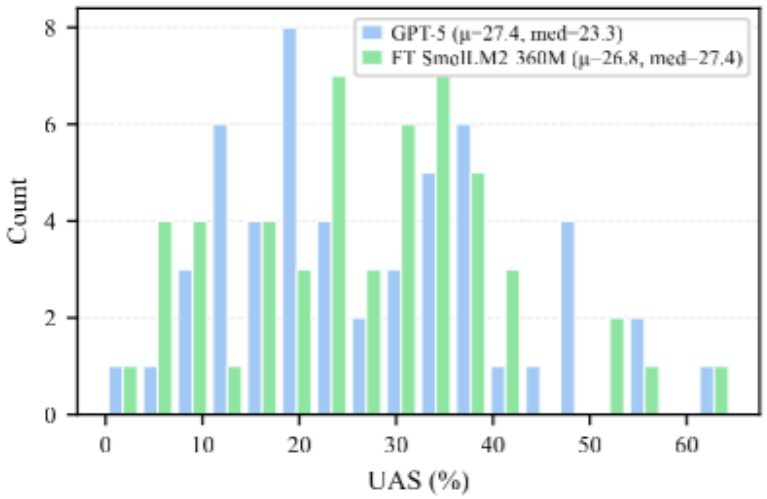
APPENDIX

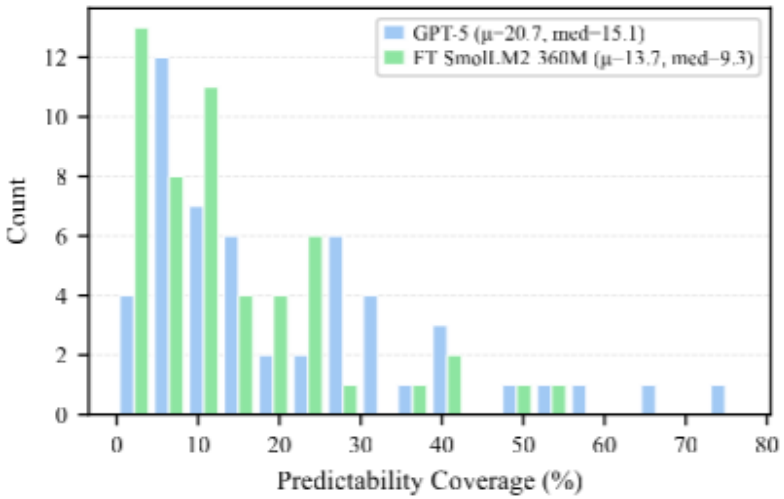
Variance reruns

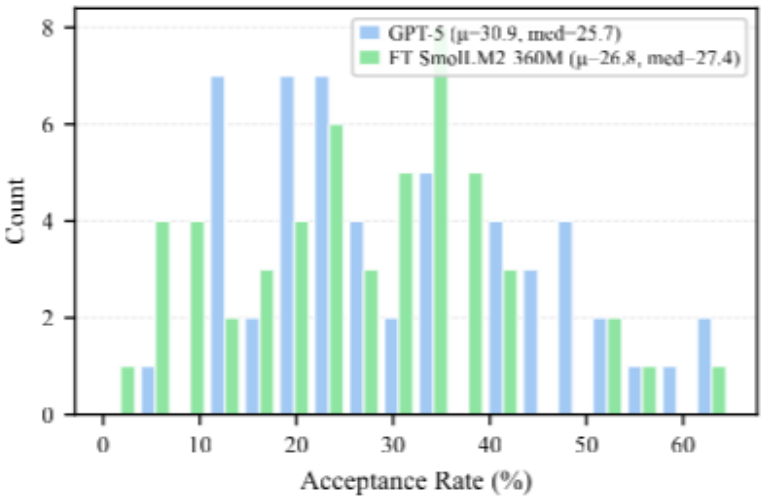
SETTING	RUN	UAS	AR
Multi-action	Run 0	22.31	19.55
	Run 1	23.29	19.87
	Run 2	23.32	20.11
	Run 3	23.72	20.52
	Run 4	24.04	19.92
	Mean $\pm$ s.d.	23.34 ± 0.65	19.99 ± 0.36
Single-action repredict	Run 0	27.35	31.73
	Run 1	27.42	31.43
	Run 2	26.49	31.15
	Run 3	26.59	31.01
	Run 4	27.11	31.40
	Mean $\pm$ s.d.	26.99 ± 0.43	31.34 ± 0.28

APPENDIX

Distributional shift







APPENDIX

Algorithm

```
class ISolver:
  def predict(s: Spreadsheet,
              h: Action[]) → Action[]:
    // to implement

def evaluate(ISolver solver,
             Action[] sequence,
             bool repredict = False):
    S ← Spreadsheet.empty()
    H ← []
    F ← sequence
    while F:
        while True:
            P ← solver.predict(S, H)
            if not accept(P, H, F):
                break
            S ← execute(P, S)
            F ← update(F, P, S)
            H ← H + P
            if not repredict: break

def update(F, P, S):
    S_p ← execute(P, S)
    S_f ← execute(F, S_p)
    F' ← []
    // remove satisfied operations and only keep
    for op in F:
        if not satisfied(op, S_p):
            F' += residual(op, S_p)
    // undo false positives with inverse operations
    for p in false_pos(S_p, S_f):
        F' ← [invert(p)] + F'
    // verify and patch
    if execute(F', S_p) ≠ S_f:
        F' ← patch(F', S_p, S_f)
    return F'
```

APPENDIX

Action Granularity.

FINER — COARSER

Too fine

Keystrokes & clicks — record / replay macros loop over your keys & mouse with no idea what they're doing.

Prior work lives here: **keystroke macros**

OUR UNIT

One UI action

One operation at one moment — a value or formula, a format, a border, a merge, on a cell or range. **Symbolic & checkable.**

FlashFill sits near us — but only fills values that follow fixed patterns; we predict the **full action.**

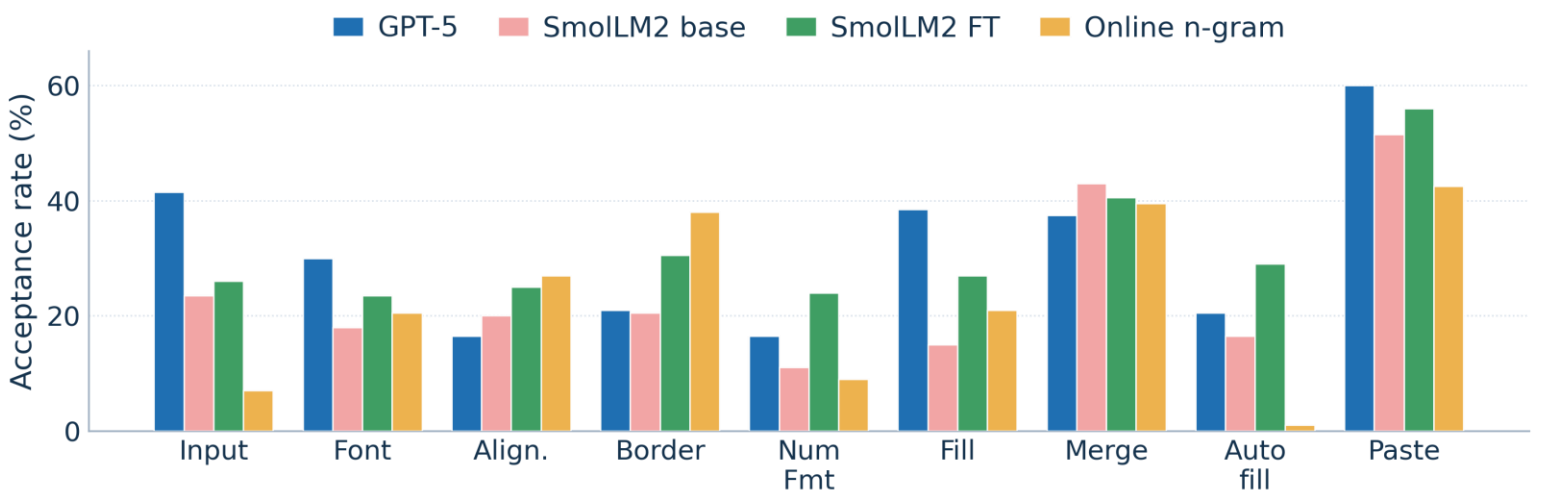
Too coarse

"Build me a budget" — a natural-language goal; useful, but unverifiable as a single next step.

Chat copilots (Copilot in Excel, spreadsheet agents) sit here — they need detailed NL instructions.

APPENDIX

Big wins on content; formatting comes in bursts.



Content (values, formulas) is the most predictable — clear patterns to extend.

Edits are bursty. Mean streak 2.21 — once a user starts, similar ops follow, so one good prediction unlocks several.