

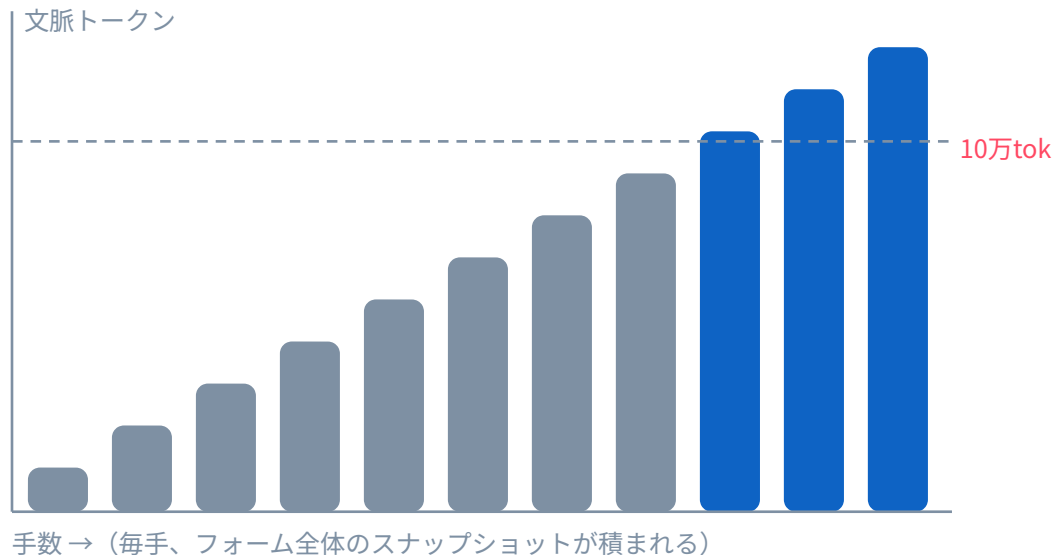


文脈は、少ない方がうまくいく

論文紹介 | Less Context, Better Agents: Efficient Context
Engineering for Long-Horizon Tool-Using LLM Agents
Lodha et al., Microsoft, 2026 (arXiv:2606.10209) | 機械学習勉強会
2026-10-06

ツール応答が、エージェントの文脈を食い潰す

ERP のツール応答は 1 回 500~3,000 トークン。15~30 手で 5 万~15 万トークンに膨らむ。



文脈窓を使い切る

完了前に上限へ。大きな窓でも数十手で尽きる



コスト・遅延が線形に増える

入力トークンが総量の 99.8%。
履歴の長さ = 請求と待ち時間



context rot

上限に達する前から想起精度が落ちる (Anthropic のブログ)

問い: 会話履歴を「全部持つ」ことは、本当に安全なのか

経費明細化: 残額 0.00 で初めて完了

領収書 1 枚を ERP の明細行に分解し、**残額がちょうど 0.00** になって初めて完了。

領収書 (ホテル)

Daily Room Rate	138.10
Hotel Tax	6.82
Hotel Tax	10.23
Entertainment External	101.58
Room Service & Meals	76.32
Total	333.05



ERP の明細行

Room charge	138.10
Room tax	6.82
Room tax	10.23
Business entertainment	101.58
Room service	76.32
remaining	0.00

同名・異額の繰り返し

Hotel Tax が 6.82 と 10.23。3 泊なら Room が 3 行。「もう入れた」と思って飛ばすと、残額がそのぶん残る

23 種の固定語彙への写像

Entertainment External → Business entertainment。Room tax と Non-Room tax のような近い語が並ぶ

1 セントでも残れば経費報告は確定できない。部分完了は業務上「失敗」

GPT-5 が MCP 経由で D365 F&O を操作し、GPT-4.1 が「ユーザー役」として質問に答え、完了まで催促する

評価ハーネス（非対話・人間不在）

User model GPT-4.1

確認質問に答え

残額 0 まで催促



Agent GPT-5

手順・23 サブカテゴリ・

写像規則を

システムプロンプトに



MCP server

21 ツール

フォーム 13

OData 6 / API 2



D365 F&O

経費報告フォーム

明細グリッド

1 応答 500~3,000 tok

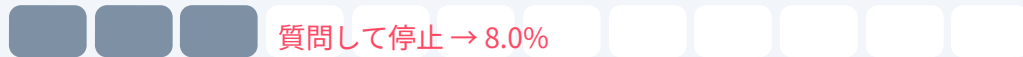
採点は自己申告ではなく、保存後のフォームを ERP から読み戻して行う

4つの構成: 文脈ポリシーだけを変える

文脈ポリシーだけを変えた C2 → C3 → C4 の比較が本論文の主張。C1 は動機づけ用。

帯 = 12 手分のツール呼び出し／応答ペア。青 = 原文のまま、紺 = 要約、白 = 捨てる

C1 GPT-5 のみ・ユーザーモデルなし
確認質問をして止まる（誰も答えない）



C2 全履歴（標準的な作り）
ツール応答をすべて保持。比較の基準線



C3 直近 5 ペアだけ保持
古いペアは破棄。約 2 行分の作業記憶



C4 直近 5 + 追い出した 3 件の要約
要約は LLM 1 回。追い出し位置へ挿入



C1→C2 の跳ねはユーザーモデルの効果。文脈工学は C2→C3→C4 で測る

Algorithm 1: 直近 N ペアを残し、追い出した W 件を要約する

毎手、元の全履歴 H から保持文脈 K を作り直す。追い出しの単位はツール呼び出し／応答のペア。

H 全履歴 (c = 9 ペア)



$d = \max(0, c - N) = 4$ ペアを先頭から追い出す

直近 $N = 5$ ペアは原文のまま

▼ 追い出した直近 $W = 3$ 件（紺）だけを LLM 1 回で要約。
それより古い 1 件（灰）は消える

K 今回の推論に渡す文脈



要約は「追い出した位置」に 1 メッセージとして挿入

C2

$N = \infty$ (追い出しなし)

C3

$N = 5, W = 0$

C4

$N = 5, W = 3$

$W = -1$ 全件要約 = compaction

要約は累積ではなく、追い出した直近 W 件だけのスライディングウィンドウ。N を超えた後は毎手 1 回

完了率 71% → 92%、トークン -63%、時間 -60%

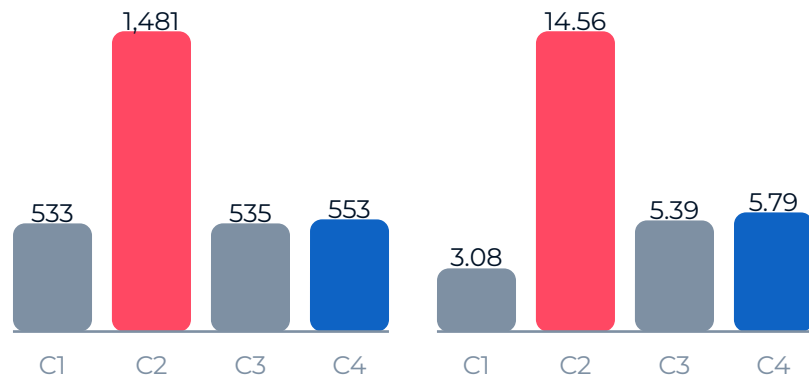
直近 5 + 要約が完了率で最良。全履歴は直近 5 に精度でも負ける（トークンと時間の最小は直近 5）。

完全明細化（残額 0.00 のタスク比率、50 タスク × 5 run）



総トークン (K)

実行時間 (h)



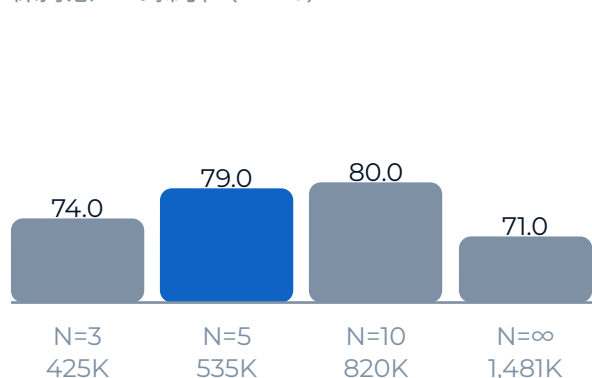
C2 は C4 の 2.68 倍のトークンと 2.5 倍の時間を使い、完了率は 20pt 低い

C2→C4: 完了率 +20.6pt ・ トークン -63% ・ 時間 -60%。要約 +3.4% は要約器未計上の疑い

N=5・W=3 は膝。全履歴要約でも同点

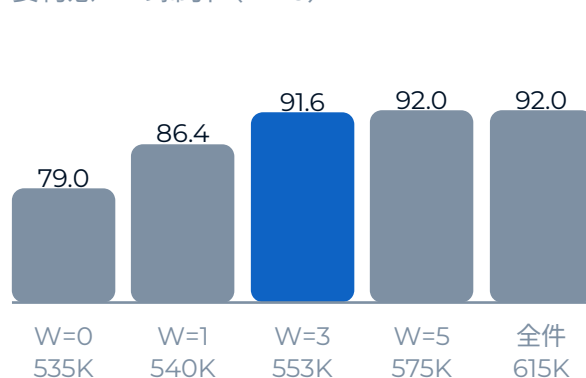
N は 5、W は 3 で頭打ち。全履歴要約 (W = -1) も 92.0% でスライディングウィンドウと同点。

保持窓 N の掃引 (W = 0)



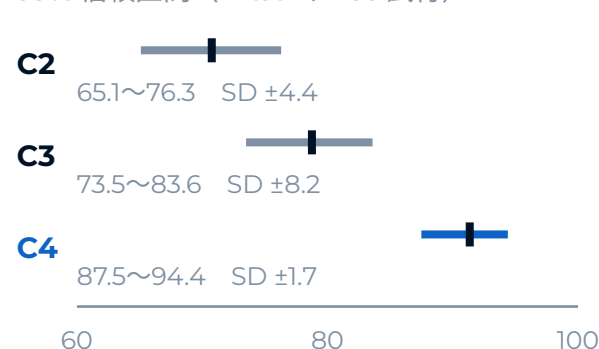
N=10 は +1pt のために 53% 多いトークン。N=∞ は精度でもトークンでも負ける

要約窓 W の掃引 (N = 5)



W=3 が膝。全履歴要約はトークン +11% で精度は同じ

95% 信頼区間 (Wilson、250 試行)



C4 は C3 と分離。C2 と C3 はわずかに重なる。刈り込みだけ (C3) の run 間 SD が最大で、要約 (C4) が分散を吸収する

カテゴリとモデルを変えても順序は同じ

3 カテゴリとも C2 → C4 で +19~21pt。Claude Sonnet 4.5 でも順序は同じだが、刈り込むと時間は増えた。

完全明細化 (%) カテゴリ別・GPT-5



Claude Sonnet 4.5 (ユーザーモデルなし・Hotel)

構成	完了	トークン	時間
全履歴	88.0	3,562K	6.20h
直近 5	92.0	2,161K	10.70h
直近 5 + 要約	94.5	2,235K	11.30h

トークンは4割減るのに時間は1.7倍。全履歴の3,562KもGPT-5の2.4倍で、手数が多し可能性（呼び出し回数の報告なし）

Sonnet はユーザーモデルなしで 88%。C1 の 8% は GPT-5 の「止まって聞く」癖

文脈で大きく直るのは2種類

C2は古い状態の参照、C3は途中終了。C4は両方を抑え、残りはモデル側の誤り。

	C2 全履歴	C3 直近 5	C4 + 要約	
古い状態の参照 (stale-state)	34	6	4	文脈ポリシーで 直る
途中終了 (premature termination)	9	18	3	
サブカテゴリ誤マッピング	8	9	6	直りにくい (C4でも残る) モデルの推論・ ツール結合の誤り = 5.6%
繰り返し行の重複／抜け	12	11	5	
ツール／フォーム操作エラー	6	5	2	
残額の計算ずれ	4	4	1	
非完了タスク数 (5 run 合計)	73	53	21	

予測を先に立てて表で検証している。非完了は 73 → 53 → 21

同じ「残額が合わない」でも、**原因は全履歴と刈り込みで正反対。**

(1) 古い状態の参照 — C2 全履歴

Appendix F より要約

get_form_state → Room 180 / remaining 420

add_line(RoomTax, 40) → ok

get_form_state → Room, RoomTax / remaining 380

… (5 手) …

最初のスナップショットを参照して

add_line(Room, 180) ← 既にある行を再追加

submit_form()

残額 -180 (超過)。古い履歴が「今の状態」に見えた

(4) 途中終了 — C3 直近 5

Appendix F より要約

get_form_state → total 340 / remaining 240

add_line(Parking, 45) → ok

(刈り込み窓が remaining 240 の応答を落とす)

エージェントは残額を見ていない

submit_form() ← 195 残したまま保存

残額 195。進捗を示す情報が窓から落ちた

C4 の要約器の実例: 「ER-00184 を開いた / Itemize を押した / Hotel-Room 180.00 を追加 / Hotel-Tax 14.40 を追加」

残額は計算しないが、途中終了を抑えるには足りた

論文の環境 (社内 D365) は再現できないので、[同じ構造の疑似 ERP](#) を作って回す (gpt-5 と gpt-5.5)。

疑似 ERP 環境

- 6 ツール: open / 一覧 / 状態取得
add_line / remove_line / 保存
- D365 風メタデータを付けて
1 応答 ≒ 3,000 tok (4 倍条件も)
- 領収書 12 件 (10~22 行、最大 5 泊)
同名・異額の繰り返しを再現
- 完了 = 残額 0.00。行正答 = 正解行の
一致率 (余分な行は数えない)
- toolfix 版: add_line が
残額と行一覧も返す

ポリシー	完了	行正答	入力 tok	cache	失敗
gpt-5 全履歴	12/12	100%	649K	89%	—
gpt-5 直近 5	3/12	69%	1,000K	6%	手数上限 9
gpt-5 直近 5+要約	12/12	100%	554K	8%	—
gpt-5 toolfix	12/12	100%	333K	11%	—
gpt-5.5 全履歴	12/12	100%	603K	89%	—
gpt-5.5 直近 5	12/12	99%	317K	16%	—
gpt-5.5 直近 5+要約	12/12	99%	292K	18%	—
gpt-5.5 全履歴 ×4 応答	12/12	100%	2,050K	90%	—

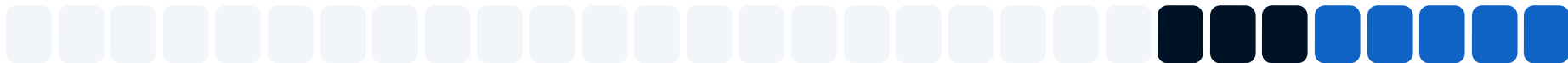
gpt-5 は論文どおり直近 5 だけで崩れ、要約か toolfix で救える

gpt-5.5 は文脈 28 万 tok の全履歴でも崩れず、コストと時間も全履歴が最良

我々の規模では全履歴が最良だった。「少ない方がいい」は条件つき

直近 8 手しか見ていないのに 91.6% 出るのは、**最新の状態取得に全行と残額が入っている**（はず）から。

1 タスク = 15~30 手のツール呼び出し。N=5 + W=3 が覆うのは直近 8 手だけ



消える（最大 22 手）。W はペア単位と解釈（論文は曖昧）

要約

原文

最新の get_form_state 応答が持っているもの

登録済みの全明細行・充当額・残額。「全体の進捗」は最新のツール応答にあり、要約が埋めるのは状態取得が無い区間。ただし取得頻度は論文未報告（仮定）

全履歴要約と同点。ただし要約の中身は効く

91.6%

W = 3（直近 3 件のみ）

同点を示すのは「3 ペアより古い情報は不要」。

W=1→3 で +5pt、C3→C4 で SD 8.2→1.7 なので中身は効く

92.0%

W = -1（全件要約）

「全体の進捗を供給する」は過大。実質は「状態取得が無い区間の穴埋め」

途中終了の原因は「残額が窓から落ちる」こと。**add_line が残額と行一覧を返せば、要約なしでも完了判定できる。**

論文の環境: 他社 ERP の MCP をそのまま使う

- ❌ add_line → {ok, line_id}。残額は get_form_state を呼ばないと見えない
- ❌ 応答はフォーム全体のスナップショット。古い状態が文脈に残る
- ❌ 窓から残額が落ちると途中終了 → 要約で補う



自社ツール: 応答を設計で直せる

- ✅ add_line → {ok, remaining, lines}。毎手、残額が最新応答に載る
- ✅ UI メタデータは削り、完了判定に要る状態（残額・行一覧）は毎回返す
- ✅ 要約なしでも完了判定できる（デモで検証）

論文内部にも食い違い: § 3.2 「全応答がスナップショットを含む」と Appendix F の add_line → ok だけ。全応答に残額があれば「窓から落ちる」は起きないはず。

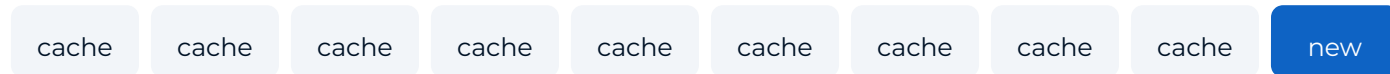
World of Workflows (Skyfall AI, 2026-01) も同じ結論: 応答に DB 変更の監査情報を足すだけで GPT-5.1 の成功率 2% → 14%

ツール応答に手を入れられないときの対症療法。自社ツールなら設計で潰す

コストの主張はキャッシュを無視している

名目トークン 2.68 倍の差は、**プロンプトキャッシュを無視した数字**。

全履歴 C2 末尾に追記



接頭辞が変わらないので、前手までの履歴はキャッシュ読み（通常 1/10 の価格）

刈り込み C3/C4 先頭を削る



每手、最古のペアを消すと接頭辞が変わり、固定部分（system・タスク）以外はキャッシュミス

デモ実測（gpt-5-mini、6 タスクの 1 タスク平均）。換算 = 非キャッシュ + キャッシュ読み × 0.1 + 要約器。右端は時間



コストはキャッシュ込みで比較し直す。精度の優位は残り、時間はキャッシュと手数次第

自社ツールを持つ我々は、**文脈を削る前に「ツールが何を返すか」を決められる。**



経費精算の明細・内訳

領収書 → 内訳 → 金額、合計一致。完了基準（合計一致）とツール応答の設計が同じ論点になる



setting-agent の下書き

マスター一覧の応答は冗長になりやすい。「最新の状態スナップショット+直近操作」だけ渡す



ツール応答の規約

更新系は完了判定に要る状態（残額・件数）を返し、超過を拒むガードも持つ。参照系（マスター一覧）は文脈から追いつかない



評価の作法

DB を読み戻して採点。合計一致だけでなく科目の正誤も測る。失敗を分類してから対策する

文脈工学はツール応答を直せないときの道具。設計で潰せるなら、そちらが先

発表後に話したい5点。どれも「我々のエージェントではどうか」が問い。

1

stale-state の計測

我々のエージェントで「古い状態を見て間違える」失敗は、どれくらい起きているか。観測しているか

2

ツール設計の規約

ツール応答に「残額」のような完了判定用の状態を必ず含める規約を作れるか

3

要約の設計

要約 (compaction) を入れるなら、汎用の自由記述でよいか、業務ごとに構造化すべきか

4

コストの再試算

プロンプトキャッシュ前提で、刈り込みの損益分岐はどこか。デモの規模では全履歴が最安・最速だった

5

完了基準の盲点

合計一致だけを完了基準にしてよいか。科目の正誤をどう測り、参照系の応答をどう守るか

Notion 本文（準備ページ）に論文の全節と私の読み、デモの手順を書いてある