EWHA,
THE FUTURE
WE CREATE

CoT or ReAct: Evaluating ReAct-Based LLM Reasoning for Korean Date and Schedule Processing

Yein Oh, Jisu Kim, Jihyun Jung, Sooyeon Park

EWHA,
THE FUTURE
WE CREATE

Table of Contents

I. Introduction
II. Tasks
III. Methodology
IV. Experiments
V. Results
VI. Conclusion

I. Introduction

Research Background – Korean Date and Schedule Processing

- NLP-based **schedule calculation** and **time normalization** are core functions in various AI services.
- Korean presents a high degree of difficulty due to complex constraints, including
  - **relative tense**
  - **complex combination of public holidays, weekdays, and intervals**
- Existing LLM-based approaches using **few-shot learning**, **Chain-of-Thought (CoT)** reasoning, and **fixed JSON output schemas** have achieved a moderate level of performance.
- However, they have shown **limitations** in consistently and accurately satisfying these constraints.

Research Background – Korean Date and Schedule Processing

<table>
<tr><th colspan="2">year</th><th colspan="2">month</th><th colspan="2">day</th><th colspan="2">week</th><th colspan="3">time</th></tr>
<tr><th>Rules</th><th>Value</th><th>Rules</th><th>Value</th><th>Rules</th><th>Value</th><th>Rules</th><th>Value</th><th>Attribute</th><th>Rules</th><th>Value</th></tr>
<tr><td>N년(앞|전)</td><td>year="-N"</td><td>N개월(앞|전)</td><td>month="-N"</td><td>N일(앞|전)</td><td>day="-N"</td><td>N주(앞|전)</td><td>week="-N"</td><td rowspan="3">hour</td><td>N시간(앞|전)</td><td>hour="-N"</td></tr>
<tr><td>N년(뒤|후)</td><td>year="+N"</td><td>N개월(뒤|후)</td><td>month="+N"</td><td>N일(뒤|후)</td><td>day="+N"</td><td>N주(뒤|후)</td><td>week="+N"</td><td>N시간(뒤|후)</td><td>hour="+N"</td></tr>
<tr><td>내년|다음
(해|년)</td><td>year="+1"</td><td>(내|다음)달</td><td>month="+1"</td><td>내일|다음날</td><td>day="+1"</td><td>(저번|이번|
다음)주</td><td>week="-1",
"0", "+1"</td><td>N시간째</td><td>hour="~N"</td></tr>
<tr><td>작년|(저번|
지난|이전)해</td><td>year="-1"</td><td>(저번|지난|
이전)달</td><td>month="-1"</td><td>어제|이전날</td><td>day="-1"</td><td>N주(차|째)</td><td>week="~N"</td><td rowspan="3">minute</td><td>N분(앞|전)</td><td>minute="-N"</td></tr>
<tr><td>내후년</td><td>year="+2"</td><td>이번달</td><td>month="0"</td><td>(이틀|사흘|...
)뒤</td><td>day="+2",
"+3", ...</td><td></td><td></td><td>N분(뒤|후)</td><td>minute="+N
"</td></tr>
<tr><td>재작년</td><td>year="-2"</td><td>N개월(차|째)</td><td>month="~N"</td><td>(이틀|사흘|...
)전</td><td>day="-2",
"-3", ...</td><td></td><td></td><td>N분째</td><td>minute="~N"</td></tr>
<tr><td>이번
(해|년)|올해</td><td>year="0"</td><td></td><td></td><td>모레|(사흘|
글피)|나흘</td><td>day="+2",
"+3", "+4"</td><td></td><td></td><td rowspan="3">second</td><td>N초(앞|전)</td><td>second="-N"</td></tr>
<tr><td>N년(차|째)</td><td>year="~N"</td><td></td><td></td><td>그저께</td><td>day="-2"</td><td></td><td></td><td>N초(뒤|후)</td><td>second="+N
"</td></tr>
<tr><td></td><td></td><td></td><td></td><td>N일(차|째)</td><td>day="~N"</td><td></td><td></td><td>N초째</td><td>second="~N"</td></tr>
</table>

※ [1] Source: Jeong, Y. S. et al. (2022). Rule-based Normalization of Relative Temporal Information.

Research Background – Korean Date and Schedule Processing

- However, they have shown **limitations** in consistently and accurately satisfying these constraints.

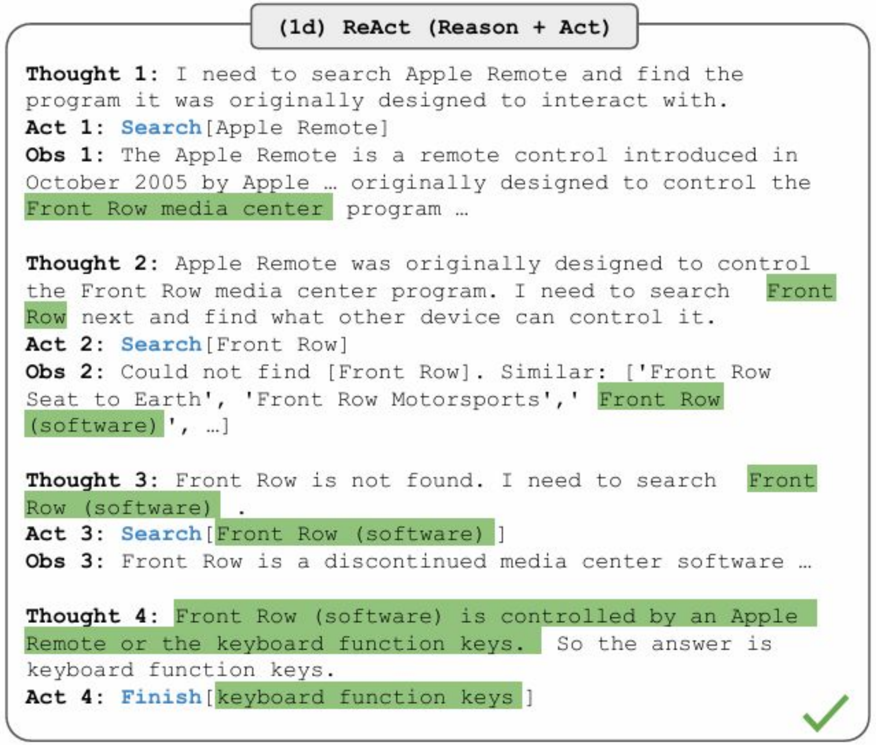
Attribute	Accuracy(%)
year	73.14%
month	61.21%
day	74.57%
week	91.39%
hour	75.00%
minute	50.00%
second	100.00%

※ [1] Source: Jeong, Y. S. et al. (2022). Rule-based Normalization of Relative Temporal Information.

Proposal – ReAct

- ReAct = **Reason** + **Act**
  - Loop structure of “**Reasoning**” ↔ “**Action**” ↔ “**Observation**”
  - Incorporates external information through tool calls

    → Enables more structured and logical reasoning compared to simple CoT prompting



EWHA WOMANS UNIVERSITY

※ [2] Source: Yao et al. (2023). *ReAct: Synergizing Reasoning and Acting in Language Models*.

II. Tasks

Three Tasks for Evaluating Korean Temporal Reasoning

- **Constraint-based Korean Schedule Generation**
  : to accurately **interpret complex temporal and scheduling constraints** within a **Korean** natural language query, in order to generate a list of dates that satisfies all specified conditions
  - Input: "수능 다음날부터 공휴일 제외하고 3일 간격으로 2개 날짜 제안해줘"
  - Output: a **list of dates** that **satisfy the given constraints**

- **Three Tasks (Increasing in Complexity)**
  - **T1**: Convert relative time expressions, **phrases** → absolute dates
  - **T2**: Convert relative time expressions, grounded in context, using complete **sentences** → absolute dates
  - **T3**: Generate **schedules** satisfying **multiple interacting constraints**

EWHA WOMANS UNIVERSITY

Task Details

To compare CoT vs. ReAct robustness, we design **three tasks**, each isolating a different reasoning challenge:

- **T1 — Date Normalization (Phrases)**
  - Input: **standalone relative expression**
    - e.g., “이번 달 마지막 수요일”
  - Requires sequential temporal arithmetic (month-end → next weekday).
  - Output: **one absolute date** (YYYY-MM-DD).
  - Tests **core temporal arithmetic** without linguistic ambiguity.
- **T2 — Date Normalization Using Sentences**
  - Input: **full natural-language sentences** containing temporal phrases.
    - e.g., “2주 뒤인데, 만약 그날이 공휴일이면 다음 평일로 해줘.”
  - Requires finding the relevant span **before** calculating the date.
  - Same output format as T1.
  - Tests **linguistic grounding + disambiguation**.
- **T3 — Constraint-based Schedule Generation**
  - Input: **multiple explicit constraints** (intervals, weekdays, exclusions).
    - e.g. “내년 1월 첫 번째 금요일부터 시작해서 5일 간격으로 4회 일정 잡아주세요”
  - Must iteratively generate dates and filter invalid ones.
  - Output: **list of dates**.
  - Tests **long-horizon reasoning, constraint satisfaction, rule integration**.

Hypotheses

- **H1**: The more **complex** the constraints, the **higher** the **accuracy** and constraints satisfaction in **ReAct** Agent.
- **H2**: **ReAct** may **increase response time** and **token usage**, but shows **Pareto-efficient** accuracy-to-cost performance.

III. Methodology

CoT: Chain-of-Thought Reasoning

Few-shot Chain-of-Thought (CoT) Prompting	**Purely Internal Reasoning**	**Task-specific Formatting**	**Purpose as a Baseline**
• Each task uses a small set of demonstrations showing the desired reasoning format. • Model is instructed to produce step-by-step reasoning followed by a final answer.	• All computations occur inside the LLM. • No external tools, symbolic modules, calculators, or validation steps. • Represents the natural behavior of text-only reasoning.	• T1 & T2: Output is a **single normalized date** (YYYY-MM-DD). • T3: Output is a **sequence of dates** satisfying constraints. • Prompts differ only in the input/output description; core CoT style is consistent.	• Provides a clean comparison against ReAct. • Highlights how agentic stepwise execution can improve temporal reasoning beyond static CoT.

ReAct Model: Thought → Action → Observation Loop

Core Mechanism

- Alternates between:

  Thought (internal reasoning) → **Action** (tool call) → **Observation** (tool output).
- Modular design: planner (thought), tool executor (action), decision module (observation evaluation).
- Same loop for all tasks, but behavior differs by task complexity.

Tools Used Across Tasks

- **Calculator** — date arithmetic (offsets, intervals, weekday shifts).
- **Calendar_db** — holidays, lunar/solar terms, anniversaries.
- **Search** — event lookup when rules are non-deterministic.
- Modular separation supports multi-step reasoning in T3 while keeping T1/T2 lightweight.

T1: Phrase-Level Date Normalization	T2: Sentence-Level Date Normalization	T3: Constraint-Based Schedule Generation
• Input: isolated temporal phrase. • Single-step reasoning → one tool call (e.g., calculator, calendar_db). • No iteration needed; observation directly yields normalized date. • Focus: pure temporal arithmetic.	• Must first locate the temporal span inside a full sentence. • Thought module extracts the temporal phrase and reformulates it into canonical tool input. • One tool call → one observation → final normalized date. • Difference from T1: requires linguistic span identification before arithmetic.	• Fully iterative, multi-step reasoning loop. • Planner decides next action: ○ Find start date / Apply intervals / Check rule violations. • Observation is evaluated by decision module → returns structured state + continue/stop signal. • Loop continues until constraints are satisfied (or terminates if the 10-turn limit is exceeded).

EWHA WOMANS UNIVERSITY

IV. Experiments

Experiment Design: Method Implementation

Comparison Setup

<table>
<tr>
<td>
Baseline (CoT)

Few-shot + Chain-of-Thought prompting
No tool use; all reasoning done internally
Fixed JSON output schema for all tasks
One-pass generation (no intermediate verification)

</td>
<td>
ReAct Agent (Method-Aligned)

Implements the Thought → Action → Observation loop
T1 / T2:

Single-step reasoning
Thought: interpret temporal phrase/sentence
Action: one call to calculator or calendar_db
Observation: tool returns the normalized date → final output

T3:

Fully iterative multi-step planning
Thought: decide next operation based on constraints & partial schedule
Action: tool calls for date arithmetic, holiday lookup, or external event search
Observation: decision module evaluates constraint satisfaction and updates state
Loop continues until a complete valid schedule is produced

Modular design separates planning, tool execution, and validation

</td>
</tr>
</table>

Experiment Design: Logistics

Datasets

- **T1:** 500 Korean relative-time *phrases* (single-date normalization)
- **T2:** 500 Korean relative-time *sentences* (embedded temporal span extraction)
- **T3:** 500 Korean multi-constraint scheduling queries (intervals, weekdays, holidays, counts)

Models Evaluated

- **GPT-4.1-mini** (via OpenAI API)
- **Solar Pro 2** (via Upstage API)

Experiment Design: Evaluation Metrics

Task Accuracy

- **T1 & T2 (Single-Date Tasks)**
  - Output: one ISO-8601 date
  - **Exact-match accuracy**: prediction must match gold date *exactly*
  - No credit for partial matches or ±1-day errors
- **T3 (Schedule Generation)**
  - Output: ordered list of dates
  - **Exact full-sequence match** required
  - Any mistake (wrong date, missing/extra dates, interval errors, constraint violation) → **failure**
  - ReAct exceeding **10-turn limit** also counted as failure

Efficiency Metrics

- **Latency**: wall-clock time from first prompt to final output
  - Captures extra overhead from multi-step reasoning in T3
- **Token Usage**: total prompt + completion tokens
  - Compares computational cost of CoT vs. ReAct reasoning loops

Overall Evaluation

- Accuracy = proportion of predictions passing the exact-match criteria
- Latency + token usage provide a measure of **efficiency vs. correctness trade-offs**

V. Results

T1 Performance

- **ReAct improves GPT's accuracy**:
  - GPT-ReAct outperformed GPT-CoT by **+4.0 points** (76.2% vs. 72.2%), a **statistically significant gain** ($p = .0018$)
  - Shows that ReAct meaningfully reduces single-event reasoning errors in GPT models
- **Solar models show a different trend**:
  - Solar-ReAct achieved **+10 points** over Solar-CoT (76.2% vs. 66.0%) **but without statistical significance** ($p > .05$)
  - Suggests Solar benefits from ReAct, but performance variance is higher
- **Efficiency trade-offs differ sharply across models**:
  - **Solar-CoT** = fastest and cheapest (**~1.7s latency**, low tokens), but lowest accuracy
  - **Solar-ReAct** = best balance (**~2.6–2.8s**, **2.3–2.4k tokens**) with accuracy comparable to GPT-ReAct
  - **GPT-ReAct** = highest accuracy but worst cost efficiency (**~25k tokens**, ~3.4s)
  - Highlights a **clear accuracy–cost trade-off**: the best-performing model is also the most expensive per query

- **Key insight**:
  - ReAct is effective for single-step tasks, but **its benefit depends heavily on the model backbone**
  - GPT gains accuracy at high cost; Solar gains efficiency with modest accuracy improvements

Model	Accuracy (%)	Avg Latency (s)	Avg Token Usage
GPT-CoT	72.20	~5.11	~2,388
GPT-ReAct	76.20	~3.40s	~25,000
Solar-CoT	66.00	~1.68s	~1,376
Solar-ReAct	76.20	~2.6–2.8s	~2,350–2,450

EWHA WOMANS UNIVERSITY

T2 Performance

- **GPT-ReAct delivers the strongest accuracy** (79%), outperforming GPT-CoT by **+4.6 points**
  - Improvement is **statistically significant** (McNemar *p = .030*)
- **Solar-CoT shows the lowest accuracy** (66.7%) but remains the **most computationally efficient**
  - ~1.6s latency, minimal generation (1,376 tokens)
- **Solar-ReAct improves accuracy to 74.85%**, matching GPT-CoT-level performance
  - But **not statistically significant** vs its CoT baseline ($p > .05$)
  - Maintains **low cost footprint** (2.5–3.0s latency, ~2.3–2.5k tokens)
- **GPT-ReAct trades extremely high cost for accuracy**
  - ~3.25s latency, **~25k tokens** per sample
  - Lowest **cost-normalized efficiency** despite best accuracy

- **Overall pattern**:
  - ReAct helps GPT meaningfully on T2
  - Solar benefits modestly but not significantly
  - GPT-ReAct gains accuracy through **very long reasoning traces** (tool loops, self-repair, error handling)

Model	Accuracy (%)	Avg Latency (s)	Avg Token Usage
GPT-CoT	74.40	~5.32	~2,393
GPT-ReAct	79.00	~3.25	~25,000
Solar-CoT	66.70	~1.61	~1,376
Solar-ReAct	74.85	~2.5–3.0	~2,350–2,460

T3 Performance

- **All models performed poorly on multi-event temporal reasoning**
  - GPT-CoT: **22.2%**
  - GPT-ReAct: **21.4%**
    → **No improvement** (McNemar p ≈ **1.00**)
- **Solar models also struggled**
  - Solar-CoT: **16.8%** — lowest overall, but computationally light
    (~**2.85s** latency, ~**1.3k** tokens)
  - Solar-ReAct: **21.4%**, but **not statistically significant** vs CoT
    Comes with major cost: **25–30s** latency, **21k–30k** tokens
- **GPT-ReAct showed the worst cost–performance trade-off**
  - **38–50s** latency
  - **18k+ tokens** generated
  - Accuracy still **21.4%** → **high cost, no gain**

- **Overall pattern**:
  - ReAct **does not help** in multi-event reasoning
  - **Large computational expansions** (loops, tool failures, self-repair)
  - Models fail to scale beyond single-event tasks

Model	Accuracy (%)	Avg Latency (s)	Avg Token Usage
GPT-CoT	22.20	~1.55	~768
GPT-ReAct	21.40	~38–50	~18,000
Solar-CoT	16.80	~2.85	~1,350
Solar-ReAct	21.40	~25–30	~21,000–30,000

Error Analysis

Key Failure Modes

- **CoT Failure Patterns**
  - Frequent **±1-day drift** in relative date calculations
  - **Inconsistent Korean week-boundary interpretation** (e.g., Monday vs. "이번 주/다음 주")
  - **Reasoning trace vs. final answer mismatch**, indicating unstable internal grounding
  - **Premature reasoning termination**, leading to partially resolved outputs
- **ReAct Failure Patterns**
  - **Tool-driven instability** (bad tool outputs propagate through later turns)
  - **Turn-limit collapses** (hitting 10-step cap before converging)
  - **Redundant re-evaluation loops** and repeated tool calls
  - **Incomplete constraint chaining**, especially in multi-constraint scheduling
- **Task-Level Observation**
  - Problems **amplify sharply in T3**
  - Tool use often **increased**, rather than reduced, error density
  - Indicates that **tool invocation ≠ structured planning** in multi-event reasoning

Ablation Study: Effect of Tool Flexibility

Objective

- To investigate whether the performance bottleneck originates from reasoning limitations or tool rigidity.
- Comparing **Hand-coded** Tools (Original) vs. **LLM-driven** Tools (Modified).

Experimental Results

- **Original Solar-ReAct (Rigid Tools)**
  - Accuracy: **21.40%** (107 / 500)
  - Failure Cases (Max-turn limit exceeded): **218** cases
- **Modified Solar-ReAct (LLM-driven Tools)**
  - Accuracy: **31.40%** (157 / 500)
  - Failure Cases (Max-turn limit exceeded): **85** cases

Key Findings

- **Performance Gain**: Accuracy improved by +10.0%p.
- **Error Reduction**: Significant decrease in "turn-limit collapse" errors (218 → 85).

Hypotheses Validation Summary

H1: "The more complex the constraints, the higher the accuracy and constraint satisfaction in ReAct Agents."	**H2**: "ReAct may increase response time and token usage, but shows Pareto-efficient accuracy-to-cost performance."
Evaluation: Not supported **Why:** • **Single-event tasks (T1/T2):** ReAct *did* improve accuracy for GPT (statistically significant) and modestly for Solar (not significant). → This is *positive*, but these tasks had **low complexity**. • **Multi-event tasks (T3):** These are the **most complex constraints**, involving multi-step temporal reasoning. Across all models: ○ GPT-CoT: **22.2%** ○ GPT-ReAct: **21.4%** (no improvement, $p \approx 1.0$) ○ Solar-CoT: **16.8%** ○ Solar-ReAct: **21.4%** (not significant; massive token cost) • **ReAct did *not* increase accuracy under high constraint complexity.** In fact, performance **collapsed**, and ReAct often made things *worse* by entering long, unproductive loops. **Conclusion:** ReAct does **not** show higher accuracy when constraints grow more complex. Instead, accuracy **falls sharply**, disproving H1.	**Evaluation: Partially supported (supported for Solar, contradicted for GPT)** **What the data says:** **GPT: Not Pareto-efficient** • Token usage: **~25k** per response • Latency: **3.2–3.4s** • Accuracy: **79% (T2)** • **Cost skyrockets**, and accuracy gains are small (+4–5 points). • In T3, GPT-ReAct has massive cost increases (18k–50k tokens, 38–50s latency) with **zero accuracy benefit**. → GPT-ReAct is **not** Pareto-efficient. **Solar: Largely Pareto-efficient** • **Solar-CoT:** cheapest but least accurate (66–67%). • **Solar-ReAct:** moderate cost (2.5–3s, ~2.3–2.5k tokens) with meaningful (but not significant) accuracy gains (74–76%). • Solar-ReAct essentially matches **GPT-CoT accuracy at a fraction of the cost**. → Solar-ReAct appears to be **near-Pareto optimal**: small cost increase → measurable benefit. **Conclusion:** • **H2 holds for Solar**, where ReAct produces a good accuracy–cost trade-off. • **H2 is contradicted for GPT**, where ReAct explodes token usage with diminishing returns. → Overall: **partially supported**.

VI. Conclusion

Summary of Findings

- **Overall Goal:** Compare CoT vs. ReAct for Korean temporal-reasoning tasks.
- **T1 & T2 — Single-event reasoning**
  - ReAct outperforms CoT by **4-6 percentage points**.
  - ReAct shows **lower latency** and **lower token usage**.
  - → **Clear accuracy–cost advantage** for ReAct in simple normalization tasks.
- **T3 — Multi-event, document-level scheduling**
  - Both CoT and ReAct show **low accuracy (17–22%)**.
  - **No significant difference** between the two methods.
  - → Neither approach scales well to multi-constraint schedule generation.
- **Ablation Study**
  - Switching to LLM-driven tools significantly reduced failures, increasing accuracy to **31.4%**.
    → Suggests that the original low performance stemmed from **"Tool Misuse" due to rigid specifications**, rather than a lack of reasoning power.

Implications

- Evaluation of reasoning frameworks must account for **accuracy, cost-efficiency, and alignment with task structure**, not just raw correctness.
- **ReAct is effective for lightweight, single-event reasoning (T1/T2)** — delivers strong accuracy gains with *lower* latency and token cost.
- **ReAct is not sufficient for multi-event temporal reasoning (T3)** — tasks require **explicit event segmentation, constraint propagation, and hierarchical planning**, which ReAct does not provide.
- **Ablation findings show tool usage is not always beneficial** — performance improved *without* tool invocation, indicating that **stable temporal reasoning depends more on orchestration and planning** than on adding external tools.
- Future systems should prioritize **structured controllers**, **planning modules**, and **temporal decomposition pipelines** for complex Korean scheduling tasks.

Limitations & Future Work

- Current evaluation focuses primarily on **accuracy**, leaving deeper **qualitative analysis** of reasoning traces and failure patterns for future study.
- Task difficulty varies: **T1/T2 contain simpler temporal structures**, which may partly explain the performance gap compared to T3.
- T3 exposes the need for richer temporal modeling; future work should explore:
  - **Multi-event pipeline architectures** for complex temporal queries
  - **Adaptive tool-use policies** and **error-recovery mechanisms**
  - New metrics capturing **temporal reasoning stability** beyond exact match
  - Broader coverage of **Korean temporal expressions**, including discourse-level and culturally specific time references

Thank you.

References

[1] Cho, M. J., Namgung, S., & Kim, M. Y. (2022). A Rule-based Approach for Recognition and Normalization of Temporal Information in Korean Text. *Journal of the Korea Institute of Information and Communication Engineering*, 26(11), 1539-1547.

[2] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). *ReAct: Synergizing reasoning and acting in language models.* In The Eleventh International Conference on Learning Representations (ICLR). arXiv. https://arxiv.org/abs/2210.03629

[3] Jeong, Y.-S., Kim, Z. M., Do, H.-W., Lim, C.-G., & Choi, H.-J. (2015). *Temporal information extraction from Korean texts.* In Proceedings of the 19th Conference on Computational Natural Language Learning (CoNLL 2015) (pp. 250–260). Association for Computational Linguistics. https://aclanthology.org/K15-1028.pdf

[4] Piryani, B., Abdallah, A., Mozafari, J., Anand, A., & Jatowt, A. (2025). *It's high time: A survey of temporal information retrieval and question answering.* arXiv. https://arxiv.org/abs/2505.20243

[5] Eldan, O., & Yahav, A. (2025). *LM-Polygraph: Unveiling Instability in Large Language Model Leaderboards*. arXiv preprint arXiv:2506.21595.

[6] Cherukuri, M. (2025). Cost, Complexity, and Efficacy of Prompt Engineering Techniques for Large Language Models. *International Journal on Science and Technology (IJSAT)*, 16(2).

[7] IBM. (2024). *What is a ReAct agent?*. Retrieved October 21, 2025, from https://www.ibm.com/think/topics/react-agent

[8] Agent Patterns Documentation. (2024). *ReAct Pattern*. Retrieved October 21, 2025, from https://agent-patterns.readthedocs.io/en/stable/patterns/re_act.html

[9] Lim, C. G., Jeong, Y. S., & Choi, H. J. (2018). *Korean TimeBank Including Relative Temporal Information*. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018).

Appendix: Related Works

Temporal Information Extraction from Korean Texts

(Jeong, Y.-S., Kim, Z. M., Do, H.-W., Lim, C.-G., & Choi, H.-J.; 2015) [3]

Why Korean Temporal Expressions Are Challenging

- Korean's linguistic structure complicates normalization:
  - Verb morphology with stems + endings (어간/어미)
  - Frequent omission of subjects/objects
  - Multiple numeral systems (삼십 / 서른)
  - Use of lunar-calendar expressions
- These characteristics increase ambiguity for LLMs and rule-based systems.
- Accurate normalization is critical for age/interval questions and context-dependent references used by AI agents.

Main Contributions of K15-1028

- One of the earliest systems dedicated to **Korean** temporal expression recognition.
- Proposes a **two-stage pipeline**:
  - Segment text into minimal semantic units.
  - Apply **finite-state patterns** to identify temporal expressions.
- Achieves high precision/recall even in noisy, domain-specific Korean text.
- Demonstrates feasibility of structured detection despite Korean-specific linguistic challenges.

Limitations / Gap

- Rule-based method struggles with conversational or informal Korean.
- Doesn't address deep temporal reasoning or multi-step interval logic.
- Leaves open the need for **LLM-based** Korean temporal normalization.

EWHA WOMANS UNIVERSITY

It’s High Time: A Survey of Temporal QA

(Piryani, B., Abdallah, A., Mozafari, J., Anand, A., & Jatowt, A.; 2025) [4]

Why Korean Temporal Reasoning Needs Attention

- Temporal QA is complex even in English; even more so in Korean due to:
  - Morphological variation and omitted arguments
  - Multiple numeral systems
  - Relative or contextual expressions (“지난번”)
  - Lunar-calendar dates
- LLMs frequently misinterpret these forms, limiting accurate time anchoring for Korean tasks.

Main Points From the Survey

- Identifies universal challenges in Temporal QA:
  - Normalizing relative expressions
  - Anchoring events to timelines
  - Reasoning over intervals and sequences
- Notes that even advanced LLMs struggle with temporal consistency and multi-step logic.
- Rates current systems as inadequate for languages with complex temporal morphology.

Relevance to Our Project

- Survey highlights the need for **structured reasoning** frameworks to improve temporal QA.
- Korean-specific challenges amplify this need, as ambiguity and morphology make simple CoT unreliable.
- Motivates exploring **ReAct-style** reasoning loops for Korean temporal normalization and reasoning.

ReAct: Synergizing Reasoning and Acting

(Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y; 2023) [2]

- Integrates **Reasoning** + **Acting** through an iterative loop:
  Thought → Action → Observation
- Overcomes limitations of purely CoT or purely action-based methods
  - Reduces hallucinations
  - Adds intermediate verification
  - Enables adaptive decision-making

Advantages Shown in Prior Research

- Improved performance on:
  - Multi-hop QA
  - Fact verification
  - Interactive reasoning tasks
- Offers better factual grounding, interpretability, and self-correction.

Relevance to Korean Temporal Reasoning

- Temporal reasoning often requires:
  - Constraint checking
  - Step-by-step validation (e.g., calendars, dates, offsets)
  - Handling ambiguous or context-dependent expressions
- ReAct's structured loop may enhance the accuracy and robustness of Korean temporal normalization.