

DataMind at AKBC Shared Task 2026: Relation-Specific Multiple Prompting for Closed-Book Knowledge Base Completion

Tanaree Patcharatrakul¹, Peerapon Vateekul^{2*}

¹International School of Engineering, Faculty of Engineering,
Chulalongkorn University, Thailand

²Department of Computer Engineering, Faculty of Engineering,
Chulalongkorn University, Thailand

6738108421@student.chula.ac.th, peerapon.v@chula.ac.th

Abstract

We describe a system for the AKBC Shared Task 2026. The task is to predict complete sets of knowledge base objects from parametric memory alone, with no retrieval and no fine-tuning, across six relations and within a 32B parameter budget. Our system combines relation-specific multiple prompting, consistency-based aggregation, and error-driven prompt refinement. For each relation, multiple prompt variants and samples elicit predictions, with cross-prompt agreement reducing wording-specific errors and cross-sample agreement filtering unstable guesses. Aggregation is tailored to each relation, including cluster-median aggregation for numeric values and voting for categorical and set-valued outputs. The system achieves a validation macro-F1 of 0.5577 and a test macro-F1 of 0.5699, compared to organizer baselines of 0.3131 (validation) and 0.2964 (test).

1 Introduction

Large language models store a considerable amount of factual knowledge from pre-training. We ask how reliably that knowledge supports prediction without retrieval augmentation or task-specific fine-tuning. Closed-book knowledge base construction (KBC) tests this. When given a (subject entity, relation) pair, the system must list all correct objects from parametric memory, with no external database. AKBC Shared Task 2026 (AKBC 2026 Shared Task Organizing Committee, 2026) turns this into a benchmark. It requires complete and precise predictions across six relations within a 32B total-parameter budget.

Our system is built on qwen/qwen3.6-27b (Qwen Team, 2026), with a gemma-3-4b-it (Gemma Team et al., 2025) auxiliary for language identification, and applies four techniques: (1) **relation-specific few-shot prompting**; (2) **self-consistency decoding** (Wang et al., 2023) with

confidence-escalated sampling for numeric relations; (3) **cross-prompt majority voting**, where several prompt variants predict independently and a prediction is kept only when at least two agree; and (4) **error-driven prompt refinement**, where the model reflects on its own errors and proposes better prompts. The system achieves a validation macro-F1 of **0.5577**, an improvement over the 0.3131 baseline. Our submission appears on the Codabench leaderboard under the team name **DataMind**.¹

2 Task and Setup

Relations. AKBC Shared Task 2026 (AKBC 2026 Shared Task Organizing Committee, 2026) covers six relations in three answer types. First, **numeric relations** return a single numeric value as a string, scored with 5% relative tolerance:

- `hasArea` asks for the total area of a geographic entity in square kilometers.
- `hasCapacity` asks for the maximum spectator capacity of a venue.

Next, a **single-object relation** returns exactly one string or an empty prediction:

- `personHasCityOfDeath` asks for the city where a person died; living persons or unknown cases return `[]`.

Finally, **set-valued relations** return zero or more objects:

- `countryLandBordersCountry` asks for all countries sharing a land border with a given country.
- `companyTradesAtStockExchange` asks for all stock exchanges where a company is independently listed.

¹Code and prompts: <https://github.com/tanaree-pat/akbc-2026>

* Corresponding author.

- `awardWonBy` asks for all recipients of a given award by name.

Evaluation. Performance is measured by F1 averaged over all test instances. For each (subject, relation) pair, F1 is computed against the gold set, then average across all pairs. Numeric predictions count as correct within a 5% relative tolerance. Gold answers are lists of canonical names and aliases, and the evaluator accepts any alias.

Constraints. The task sets three strict constraints. First, systems are **closed-book**, which means no retrieval augmentation, web search, or access to external knowledge bases at inference. Second, **no fine-tuning** is allowed. All methods run at inference time. Third, the **total parameter count** across all models must stay under 32B. Non-neural post-processing, such as normalization, deduplication, voting, and numeric clustering, is allowed, as is multi-step agentic inference.

Data and model. The dataset has training, validation, and test splits. We use the training split only as a source of few-shot examples. `qwen/qwen3.6-27b` (Qwen Team, 2026) (27B) is the main model. In addition, a `gemma-3-4b-it` (Gemma Team et al., 2025) model (4B) is used together with Qwen for selecting a prompt language when working on the `hasArea` relation. The total parameter count for both models is 31B, staying within the 32B limit.

Baseline. The organizers provide a reference system using a Qwen 3.5 9B model with simple prompting. It achieves an overall macro-F1 of 0.3131 on validation and 0.2964 on test. Per-relation baseline scores appear alongside our results in Table 2.

3 Methods

Table 1 lists the technique used for each relation. The subsections below describe each one, while the entire set of prompts can be found in Appendix A.

3.1 Numeric Relations

hasArea. Our prompt asks the model to return the total area of a geographic entity in square kilometers as a single numeric string, and to always give a number, never an empty answer. Few-shot examples cover countries, medium-sized islands, and small islands with decimal values, so the model is calibrated across the full output range.

To predict an area, we designed a nested ensemble technique where the model answers the same question many times and those answers are calculated into a single number. This happens in two stages. First, within a single run, the model is sampled repeatedly and combined into one area estimate. Second, six independent runs each produce an area estimate, and the 6 estimates are combined into the final area estimate.

For a single run, we use confidence-escalated self-consistency (Wang et al., 2023), where the model is sampled 5 times at temperature 0.7 and the samples are pooled by a *cluster-median*. To compute the cluster-median, we group the samples within 5% of one another and return the median of the cluster with the most members. A run’s confidence is the size of its largest cluster divided by the number of samples, which measures how strongly the samples agree. The sampling escalates to nine draws if the run’s confidence is below 80%.

For the six independent runs, the six numbers are pooled by the same cluster-median to give the final area estimate. Runs 1–5 use the *same* base prompt and differ only by random sampling at temperature 0.7. Run 6 uses an **anchored native-language** variant as we believe that the model may have seen area figures in local-language sources that are absent from English text, so querying in the entity’s native language may return a more precise value. The prompt first names the entity in English (Step 1) to fix its identity, then asks for its area in the target language (Step 2). This stops the model from mapping the name to a different extent in the target language (for example, a sub-national territory read as the whole country when queried in Chinese). To pick the language, `qwen/qwen3.6-27b` first names the entity’s primary local language together with a confidence in $[0, 1]$. If that confidence is below 0.7, `gemma-3-4b-it` is asked the same question, and its answer replaces qwen’s when gemma is more confident. The prompt is issued in the chosen language only if this confidence reaches 0.7.

hasCapacity. The prompt asks the model to name the venue’s country and venue type, then estimate its capacity using typical size based on the region and type given in the prompt. The prompt separates national stadiums, top professional leagues, and community grounds by typical size ranges. As with `hasArea`, we apply confidence-escalated self-consistency, drawing 5 samples and escalating to 9 when they disagree as the model may hold the

Relation	Technique summary
hasArea	Nested ensemble: per-run SC (5→9, cluster-median), then cluster-median over 6 runs (5 base + 1 native-lang)
hasCapacity	Single prompt with confidence-escalated SC (5→9)
personHasCityOfDeath	Ensemble using agreement from all 4 prompts, else abstain
countryLandBordersCountry	Single exhaustive prompt, single-sample
companyTradesAtStockExchange	Ensemble vote, keep if ≥ 3 of 5 prompts agree, else abstain
awardWonBy	Union of alphabetical + year-by-year sweeps, each with per-name confidence vote (SC $\times$ 5, $\geq 2/5$)

Table 1: A summary of the relation-specific techniques. Note that SC refers to self-consistency.

capacity value in parametric memory but report it with variance, and sampling lets the cluster-median recover the stable value.

3.2 Single-Object Relation

personHasCityOfDeath. Four prompt variants run independently, and a city is output only if all the four variants return the same city. If any variant disagrees or abstains, the system returns `[]`. Each prompt variant is as follows.

The **recent_aware** variant drafts then revises. The model first identifies who the person is and when they were most active, then considers whether someone prominent in the 1980s–2010s and likely in their 70s–90s may have died recently. This makes the model name a death city for older figures it would otherwise assume are still alive. It commits to a city if reasonably confident, and returns `[]` only when sure the person is alive or when it has no information. The **recent_precise** variant uses the same recency reasoning plus explicit city-name rules: US sub-city districts such as Brooklyn or West Hollywood map to their enclosing city (New York City, Los Angeles), and UK deaths use the specific named town rather than the nearest major city. The **city_precision** variant uses the same city-name rules without the recency framing. It decides whether the person is living or dead, applies the naming rules if dead, and returns `[]` when it has no reliable information rather than guessing. The **meta_antidefault** variant comes from error-driven prompt refinement, similar to (Wu et al., 2024). In particular, the model was shown its past errors with gold labels and asked to find failure patterns. The resulting prompt tells the model to check whether its recalled city is the actual place of death or just a biographical link such as the career city, birthplace, or national capital, and to output a city only when it has a direct, specific memory of the death there.

Using four differently-framed prompts is the key to this relation. Each variant reasons in its own way

so each makes different thought process. Requiring all four to agree cancels out the errors any single framing would make on its own.

3.3 Set-Valued Relations

countryLandBordersCountry. A single prompt with a directional-exhaustiveness instruction (“check all directions: north, south, east, west; do not stop early”) and an explicit land-border-only constraint is used. Island nations with no land borders return `[]`. Few-shot examples include a two-neighbor case (Sierra Leone), an island nation (New Zealand), and a nine-neighbor case (Germany). We use a single model sample without ensembling because our observation shows that the same errors tended to recur across prompt variants, suggesting that majority voting may offer limited benefit.

companyTradesAtStockExchange. Five prompt variants run independently, and an exchange is kept only when at least three agree. Each prompt variant is as follows.

The **simple** variant is a baseline few-shot prompt with listed and unlisted companies. The **listed-check** variant runs two steps: Step 1 decides whether the company is independently listed under its own name, rejecting subsidiaries, private firms, cooperatives, and delisted companies; Step 2 names the exchange only if Step 1 passes. The **meta_precision** and **meta_verify** variants come from the same error-driven refinement and target over-prediction. Both prohibit inferring a listing from a company’s name, headquarters, or sector and instead require the model to recall an actual listing. The **anti_confusion** variant targets two error patterns: NYSE–Nasdaq confusion, which it addresses through ticker-length heuristics, and incorrect country-based defaults, which it addresses by requiring a recalled listing rather than one inferred from the company’s headquarters.

awardWonBy. Two runs are combined for this relation. The **alphabetical** run tells the model to recall recipients by first letter (A through Z). Meanwhile, the **year-by-year** run enumerates recipients from the award’s first year through 2026 (decade by decade for cohort-elected fellowships), which reaches deeper on long-history awards. Both prompts set a recall target: “aim for 50 or more names; a partial list of 100 names scores higher than a tidy list of 10.”

Each of the two runs is filtered by a per-name confidence vote where the model is sampled five times, and a name is kept only if it appears in at least two of the five samples. Here the output is a list of names rather than one answer, so we vote on each name on its own. When average per-name confidence is low, sampling escalates to nine draws. The two filtered runs are then unioned.

For this relation, we observed that some responses exhausted their generation budget while still reasoning, and never produced the output array. Therefore, to recover an answer in this case, we use a second call that asks the model to extract every recipient name from its own reasoning text. If that call also fails, a regex pass look for quoted, bulleted, or numbered names from the reasoning text.

4 Results

Table 2 reports macro-F1 per relation on validation and test sets, with the organizer baseline.

Overall. The system achieves a test macro-F1 of 0.5699 and a validation macro-F1 of 0.5577, compared to baseline scores of 0.2964 on test and 0.3131 on validation—a test gain of 0.2735 points. Every relation improves over its organizer baseline on both splits, with test-set gains ranging from +0.1021 (*hasCapacity*) to +0.4941 (*companyTradesAtStockExchange*).

Performance range. Validation F1 runs from 0.1802 to 0.9892 across the six relations, and the test F1 presents in a similar range, from 0.2245 to 0.9440.

Per-relation characterization. *country-LandBordersCountry* is the easiest, sitting near-ceiling on both splits. Three relations show moderate performance. *companyTradesAtStockExchange* and *hasArea* both gain from prompt ensembling, and *personHasCityOfDeath* gains from a deliberate precision-recall

tradeoff (below). The two hardest are *awardWonBy*, which is recall-bound because the model enumerates only part of the recipient set, and *hasCapacity*, where errors come from entity confusion on obscure venues.

Precision-recall tradeoff in *personHasCityOfDeath*. Unanimity voting outputs a city only when all four variants agree. On the test set, this achieves 0.8700 precision and 0.5500 recall. By abstaining when the variants disagree, the method reduces incorrect predictions but also lowers recall. This precision-recall tradeoff leads to higher F1.

Validation vs. test shifts. *companyTradesAtStockExchange* and *awardWonBy* improve a lot on test (0.6752→0.7668 and 0.1802→0.2749). Differences in split composition may contribute to these changes.

5 Analysis

From our experiments, we analyze why the remaining errors occur.

Entity prominence and identification errors. From our inspection, *hasArea* and *companyTradesAtStockExchange* errors suggests difficulties with some less prominent entities. *hasCapacity* adds a related pattern, where the model anchors on a more famous venue that shares a name or city with the target and returns a plausible value for the wrong venue.

Incomplete enumeration in *awardWonBy*. Some awards have hundreds of recipients. The model recalls some reliably but cannot list the full set. We observe that missed recipients are not less well-known than recalled ones. Hence, the limit is possibly how many names the model can produce.

***personHasCityOfDeath*.** We inspected that errors include failure to produce a death location, associated location substitution, such as a career city or birthplace, and other incorrect city predictions. Some missed deaths occurred well before the model’s reported training cutoff, indicating that recency alone cannot explain these failures.

6 Related Works

Language models store factual knowledge from pre-training text (Petroni et al., 2019), and that knowledge is uneven. Entities that appear more

Relation	Base (val)	Ours (val)	Base (test)	Ours (test)
hasArea	0.2900	0.5800	0.3400	0.5700
hasCapacity	0.1800	0.2200	0.1224	0.2245
personHasCityOfDeath	0.2100	0.5000	0.1700	0.4900
countryLandBordersCountry	0.6650	0.9892	0.7315	0.9440
companyTradesAtStockExchange	0.3543	0.6752	0.2727	0.7668
awardWonBy	0.1008	0.1802	0.1510	0.2749
Overall	0.3131	0.5577	0.2964	0.5699

Table 2: Macro-F1 per relation on validation and test splits, with organizer baseline (Base) and our system (Ours).

often are recalled more reliably (Mallen et al., 2023; Kandpal et al., 2023). This literature motivates entity prominence as a possible explanatory factor in our setting.

Prompt engineering and output aggregation matter for reliable factual recall. Jiang et al. (2020) show that prompt phrasing strongly affects what a model recalls. Self-consistency (Wang et al., 2023) improves accuracy by sampling several outputs and choosing by majority. We apply this technique to numeric clustering and per-name voting. Also, automatic prompt optimization (Li et al., 2025) improves the prompts based on erroneous cases. We apply it to build the meta_antidefault and meta_precision prompt variants.

Selective prediction, i.e., abstaining on uncertain inputs to raise precision, has been studied in QA (Kamath et al., 2020). We show the same idea works for KBC, where requiring unanimity across four prompt variants for personHasCityOfDeath raises test precision to 0.8700 at the cost of recall.

7 Conclusion

We presented a closed-book KBC system for the AKBC Shared Task 2026 that reaches a test macro-F1 of 0.5699, within a 32B budget and without retrieval or fine-tuning. Its main strength is agreement. By asking each question with several prompts and several samples and keeping only the answers that recur, the system draws out reliable knowledge and filters errors. We also tailor the techniques to each relation. The techniques improve performance over the organizer baseline, but errors remain in entity identification, enumeration, and factual prediction. These results reflect the limitations of our current configuration, which calls for future research to address them.

Acknowledgments

We would like to thank Piyawat Lertvittayakumjorn for his advice throughout this work.

References

- AKBC 2026 Shared Task Organizing Committee. 2026. AKBC shared task 2026: Predicting complete knowledge base entries from language models. <https://lm-kbc.github.io/challenge2026/>. AKBC Workshop at EMNLP 2026, Budapest.
- Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, and 192 others. 2025. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*.
- Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. [How can we know what language models know?](#) *Transactions of the Association for Computational Linguistics*, 8:423–438.
- Amita Kamath, Robin Jia, and Percy Liang. 2020. [Selective question answering under domain shift](#). In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5684–5696. Association for Computational Linguistics.
- Nikhil Kandpal, Haikang Deng, Adam Roberts, Eric Wallace, and Colin Raffel. 2023. Large language models struggle to learn long-tail knowledge. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 15696–15707. PMLR.
- Wenwu Li, Xiangfeng Wang, Wenhao Li, and Bo Jin. 2025. A survey of automatic prompt engineering: An optimization perspective. *arXiv preprint arXiv:2502.11560*.
- Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating

effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. [Language models as knowledge bases?](#) In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Qwen Team. 2026. Qwen3.6-27B. <https://huggingface.co/Qwen/Qwen3.6-27B>. Official model card.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yurong Wu, Yan Gao, Bin Benjamin Zhu, Zineng Zhou, Xiaodi Sun, Sheng Yang, Jian-Guang Lou, Zhiming Ding, and Linjun Yang. 2024. [StraGo: Harnessing strategic guidance for prompt optimization](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10043–10061, Miami, Florida, USA. Association for Computational Linguistics.

A Prompts

This appendix gives the exact prompts used in the final system. {subject} is replaced with the subject entity at inference time. The full source, including all prompt variants, is in the code repository.

A.1 hasArea

Runs 1–5 use the base prompt. Run 6 uses the anchored native-language prompt.

base: direct area query, with few-shot examples spanning countries to small islands (runs 1–5).

```
What is the total area of {subject} in square kilometers? For countries use total area (land + inland water). Return a JSON array with a single number as a string, no units.
Examples:
- Israel: ["20770"]
- Mangareva: ["15.4"]
- Wellington Island: ["5556"]
Always give your best numeric estimate -- never return an empty answer. No explanation.
```

native_lang_anchored: anchors the entity in English, then asks for its area in the local language (run 6).

```
What is the total area of {subject} in square kilometers?
Step 1: Identify what {subject} is -- its type (island, lake, country, province, etc.) and location. If it is an island or sub-national territory, confirm it refers to that specific geographic feature ONLY, not any larger administrative region or country it belongs to.
Step 2: Identify the primary local language of where {subject} is located.
Step 3: Recall what local-language sources say about this specific entity's area -- they often have more precise figures for lesser-known places than English sources.
Step 4: Apply the correct rule:
countries → land + inland water;
islands → the island's land area only; lakes → water surface area.
Always give your best numeric estimate -- never return [].
Return a JSON array with a single number as a string on the last line. No units.
```

Language identification (run 6, Step 1): asked of qwen/qwen3.6–27b, and of google/gemma–3–4b–it as a fallback when qwen’s confidence is below 0.70. The higher-confidence answer is used.

```
What is the primary local language of the place, country, or origin associated with "{subject}"? Respond ONLY as JSON: {"language": "<language name>", "confidence": <0 to 1>}
```

Translation (run 6, Step 2): when the language confidence is at least 0.70, qwen/qwen3.6–27b translates the filled-in anchored prompt ({prompt}) into the identified language ({language}), keeping the entity name in English so its identity stays anchored.

```
Translate the following text into {language}. Keep every number, JSON array, and the entity name EXACTLY as written. Output ONLY the translation, with no preamble or commentary:

{prompt}
```

A.2 hasCapacity

country_tier: identify the venue’s country and type, then estimate from region-specific capacity norms.

What is the maximum spectator capacity of {subject}?

Step 1: What country and city is {subject} in, and what type and level of venue is it?

Step 2: Estimate capacity using regional norms for that venue type:

- South America (Brazil, Argentina, Colombia): often large, 30,000-100,000
- Western Europe top division: 20,000-80,000; lower divisions: 5,000-20,000
- North America (NFL/MLB/NBA): 17,000-82,500 depending on sport
- East Asia (Japan, South Korea, China): 15,000-70,000
- Africa, South/Southeast Asia, smaller nations: 5,000-40,000

Step 3: Give your best specific number within that range.

Always give your best estimate -- never return [].

Return a JSON array with a single integer as a string on the last line.

A.3 personHasCityOfDeath

We use four prompt variants, and a city is output only if all four agree.

recent_aware: reasons about whether a long-active person may have died recently before committing to a city.

In which city did {subject} die?

Step 1: Who is {subject}? What are they known for, and when were they most active?

Step 2: As of 2026, consider deaths through 2025. If {subject} was prominent in the 1980s-2010s and was likely in their 70s-90s, they may have passed away recently -- commit to a city if you have reasonable confidence.

Step 3: If deceased, name the city. Return [] only if you are confident the person is still alive, or you have no information at all about their death.

City granularity only -- not country or region.

Examples:

- Egbert Mulder: ["Groningen"]
- Frédéric Chopin: ["Paris"]
- Dave Keon: []

Return only the JSON array on the last line.

meta_antidefault: separates a real death-place memory from a biographical default (birth-place, career city, capital).

In which city did {subject} die?

Step 1: Identify {subject} and their vital status. If alive or status unknown, go to Step 3.

Step 2: Recall the exact city of death. Separate confirmed fact from inference: is the recalled city the ACTUAL place of death, or merely where they were born, worked, or are culturally associated? Geographic defaults (national capitals, major hubs, career bases) are not substitutes for the real death location.

Step 3: Output the city ONLY if you have a direct, specific memory of where they died. If you know they died but not where, or you are inferring from association, return [].

City granularity only.

Examples:

- Egbert Mulder: ["Groningen"]
- Frédéric Chopin: ["Paris"]
- Dave Keon: []

Return only the JSON array on the last line.

city_precision: applies obituary-style city-naming rules (e.g. US boroughs map to the enclosing city).

In which city did {subject} die?

Step 1: Is this person living or deceased?

Step 2: If deceased, which city? Use the specific city name as it appears in obituaries.

Precision rules:

- US: Brooklyn, Queens, Manhattan, Bronx, Staten Island → 'New York City'
- US: West Hollywood, Beverly Hills, Santa Monica → 'Los Angeles'
- US: Cambridge MA is 'Cambridge', not 'Boston'
- UK: use the specific town or city, not the nearest major city
- Elsewhere: use the municipality where the death occurred

Step 3: Return [] if still living, or if you have no reliable information.

Examples:

- Egbert Mulder: ["Groningen"]
- Frédéric Chopin: ["Paris"]
- Dave Keon: []

Return only the JSON array on the last line.

recent_precise: combines the recency reasoning with the city-naming rules.

In which city did {subject} die?

Step 1: Who is {subject}? What are they known for, and when were they most active?

Step 2: Consider deaths through 2025. If {subject} was prominent in the 1980s-2010s and was likely in their 70s-90s, they may have passed away recently -- commit to a city if you have reasonable confidence.

Step 3: If deceased, name the city using obituary-standard precision:

- US: Brooklyn, Queens, Manhattan → 'New York City'; West Hollywood, Beverly Hills, Santa Monica → 'Los Angeles'
- UK: use the specific town (not nearest major city)
- Elsewhere: use the municipality

Return [] only if you are confident the person is still alive, or you have genuinely no information. Do not return [] just because you are uncertain -- a best guess is better than nothing.

Examples:

- Egbert Mulder: ["Groningen"]
- Frédéric Chopin: ["Paris"]
- Dave Keon: []

Return only the JSON array on the last line.

A.4 countryLandBordersCountry

base: one exhaustive directional sweep, land borders only, single-shot.

List every single country that shares a land border with {subject}. Be exhaustive -- do not stop early. Check all directions: north, south, east, west. Only land borders count, not maritime. Island countries with no land borders return [].

Examples:

- Sierra Leone: ["Guinea", "Liberia"]
- New Zealand: []
- Germany: ["Austria", "Belgium", "Czech Republic", "Denmark", "France", "Luxembourg", "Netherlands", "Poland", "Switzerland"]

Return only a JSON array of country names. No explanation.

A.5 companyTradesAtStockExchange

Five variants; an exchange is kept only when at least three predict it.

simple: baseline few-shot query for the listing exchange(s).

At which stock exchange does {subject} trade? A company can be listed on multiple exchanges.

Examples:

- Farfetch: ["New York Stock Exchange"]
- West Japan Railway Company: ["Tokyo Stock Exchange", "Nagoya Stock Exchange", "Fukuoka Stock Exchange"]
- Bangladesh Cement Manufacturers Association: []

If not publicly traded or unknown, return []. Return only a JSON array of exchange names. No explanation.

listed_check: first checks the company is independently listed, then names the exchange.

Does {subject} have its own shares that trade on a stock exchange, and if so where?

Step 1 -- Eligibility. Decide whether {subject} ITSELF has publicly traded shares under its own name RIGHT NOW. Answer 'no' (→ []) if any of these apply:

- It is a subsidiary whose shares are not separately listed (only its parent trades). e.g. a brand or unit owned by a larger listed group.
- It is privately held (family/founder/private-equity owned, no public shares).
- It is a cooperative, mutual, association, government body, or non-profit.
- It was taken private, acquired, or delisted and no longer trades on its own.

Only answer 'yes' if the company files as an independent public issuer under its own name.

Step 2 -- Exchange. If and only if eligible, name the exchange(s) it is listed on (primary listing plus any cross-listings).

Be strict: if you are not confident the company is independently listed, return [].

Examples:

- Farfetch (independent public company): ["New York Stock Exchange"]
- West Japan Railway Company: ["Tokyo Stock Exchange", "Nagoya Stock Exchange", "Fukuoka Stock Exchange"]
- Kintetsu Real Estate (subsidiary of listed Kintetsu Group): []
- Bangladesh Cement Manufacturers Association (industry body): []

Return only a JSON array of exchange names on the last line. If not independently listed, return [].

meta_precision: high-precision recall; forbids inferring a listing from name, HQ, or sector.

Return a JSON array of stock exchanges where {subject} is listed.

Strictly enforce precision: only include an exchange for which you have definitive knowledge of an actual listing. Do not infer a listing from the company's name, headquarters, or sector. Do not include private companies, or subsidiaries/acquired companies whose shares are not separately traded. If the company is not publicly traded or the listing is unknown, return [].

Examples:

- Farfetch: ["New York Stock Exchange"]
- West Japan Railway Company: ["Tokyo Stock Exchange", "Nagoya Stock Exchange", "Fukuoka Stock Exchange"]
- Bangladesh Cement Manufacturers Association: []

Return only a JSON array of exchange names. No explanation.

meta_verify: verifies public-trading status and actual recall before answering.

Identify the stock exchange(s) for {subject}.

Before outputting, verify two things: (1) the company is publicly traded (not private or an unlisted subsidiary), and (2) the exchange is a listing you actually recall -- not a guess based on the company's name or region. If either check fails, return [].

Examples:

- Farfetch: ["New York Stock Exchange"]
- West Japan Railway Company: ["Tokyo Stock Exchange", "Nagoya Stock Exchange", "Fukuoka Stock Exchange"]
- Bangladesh Cement Manufacturers Association: []

Return only a JSON array of exchange names on the last line.

anti_confusion: targets NYSE-Nasdaq confusion and wrong-country defaults.

At which stock exchange does {subject} trade?

Two common mistakes to avoid:

1. NYSE vs Nasdaq confusion. These are DIFFERENT exchanges. Use the ticker symbol to identify which one -- NYSE tickers are usually 1-3 letters (e.g. DLB, EIX, IBM, AXP); Nasdaq tickers are usually 4+ letters (e.g. MSFT, AAPL, CSCO, AMZN). Do NOT guess -- if you are unsure which US exchange, return [].
2. Wrong-country default. A company may be listed in a country DIFFERENT from its headquarters. Do not return an exchange solely because the company is based there. Only name an exchange you actually recall the company trading on.

Rules:

- Only include exchanges where the company has its OWN shares listed independently (not a subsidiary or private company).
- Return [] if not publicly traded, or if you are not confident of the exchange.

Examples:

- Farfetch: ["New York Stock Exchange"]
- West Japan Railway Company: ["Tokyo Stock Exchange", "Nagoya Stock

Exchange", "Fukuoka Stock Exchange"]

- Bangladesh Cement Manufacturers Association: []

Return only a JSON array of exchange names. No explanation.

A.6 awardWonBy

Two runs, alphabetical and year-by-year, each filtered by the per-name confidence vote, then unioned.

alphabetical: sweeps recipients by first letter to avoid year-by-year loops.

List every person or organisation that has received the {subject}.

Recall by NAME, not by year. Going year-by-year makes you loop and second-guess a single year forever. Instead, sweep your memory by first letter: think of recipients whose surname starts with A, then B, then C, and so on to Z. Write each name down as soon as it comes to mind and move on -- do not stop to verify the exact year.

Rules:

- Include ONLY the recipient's name -- no years, no winning works, no categories.
- Aim for 50 or more names. A partial list of 100 names scores far higher than 10.
- Do not deliberate about any single name -- if you are fairly sure someone received it, include them and keep going.
- Only return [] if you are certain you know no recipients whatsoever.

Write the full JSON array of names as your FIRST output, before any reasoning.

Examples:

- Nobel Peace Prize: ["Martin Luther King Jr.", "Desmond Tutu", "Nelson Mandela", "Malala Yousafzai", "Elie Wiesel", "Mother Teresa", "Jimmy Carter", "Barack Obama", "Wangari Maathai", "Henry Kissinger", "Rigoberta Menchú", "Albert Schweitzer"]

Return a JSON array of names. No explanation.

year_sweep: enumerates recipients year by year for long-history awards.

List every recipient of the {subject}, going year by year from when it was first awarded through 2026.

Format each year you are confident about as: YEAR: Name

Skip years you are genuinely uncertain about -- do not guess a name just to fill a gap.

For fellowships or memberships elected by cohort, go decade by decade instead.

Rules:

- Confirmed recipients only -- not nominees, shortlisted candidates, or people merely associated with the award.
- Names only -- no categories, no winning works, no extra commentary.
- Annual prizes with one winner: one name per year. Multiple-winner prizes: list all you know.

After the year-by-year section, compile every name into a single JSON array on the last line.

Return only the JSON array if you cannot recall any specific year-recipient pairs.

Example (Turing Award excerpt):

1966: Alan Perlis

1967: Maurice Wilkes

1968: Richard Hamming

["Alan Perlis", "Maurice Wilkes", "Richard Hamming", ...]