Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Naming the deliverable raises convention coverage in large language model outputs

Jiawei Jing

jiawei.jing@mkhr.co.jp

Makuhari Development Corporation https://orcid.org/0009-0000-8365-367X

Research Article

Keywords: Prompt wording, Domain terminology, Vocabulary problem, Genre, Large language models, Checklist evaluation, AI literacy

Posted Date: November 4th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11602543/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Naming the deliverable raises convention coverage in large language model outputs

Jiawei Jing
Makuhari Development Corporation, Chiba, Japan
ORCID: 0009-0000-8365-367X
Correspondence: jiawei.jing@mkhr.co.jp

Abstract

Practitioners compress conventions into a name such as RACI matrix or term sheet, whereas outsiders can only describe the deliverable. We tested whether the name matters when large language models write professional deliverables. Each request had three versions differing in one phrase, the practitioner name or one of two lay descriptions. Blind model judges scored outputs against convention checklists from public normative sources. On 40 curated terms, the name raised convention coverage in all six models, by 0.11 [95% CI 0.07, 0.16] to 0.29 [0.22, 0.37], and the gain survived masking the name, rebuilt checklists, and length control. On 150 terms sampled by rule from O*NET and Wikidata, it replicated in four models (0.11 [0.08, 0.14] to 0.19 [0.16, 0.23]) and was largest for method names. Gains were largest in models with the lowest lay coverage, a pattern confounded with vendor and output length. On curated terms, log-probabilities from an open model correlated moderately with per-term gains ($\rho = 0.37$). Using a name obtained from the model recovered about half to three quarters of the gain, mostly when the name was correct. We propose knowing that deliverables have names, asking for them, and checking them as an AI-literacy skill.

Keywords: Prompt wording; Domain terminology; Vocabulary problem; Genre; Large language models; Checklist evaluation; AI literacy

1 Introduction

Large language models (LLMs) have become a first stop for drafting professional text such as project plans, clinical notes, investment terms, and incident reviews. The same need can be put to a model in two ways. A project manager writes “Create a RACI matrix for launching a new company website.” Someone without project-management training writes “Create a chart showing who does what for launching a new company website.” Both point to the same deliverable, but the first carries an unwritten set of conventions with the name: tasks as rows, roles as columns, every filled cell holding R, A, C, or I, and exactly one A per task. The second says only “who does what,” so the model must either guess the convention or return an ordinary division-of-labor table.

This is a new form of an old problem in information science. Users describe needs in their own words, while documents are indexed in a field’s vocabulary, and two people choose the same word for the same object less than one time in five (Furnas et al., 1987). Query expansion and thesauri bridge the two vocabularies (Carpineto & Romano, 2012), and the reference interview does the same through conversation (Taylor, 1968). Generative models turn finding a document into writing one, but the mismatch remains. To follow a field’s conventions, the model must first know which deliverable is wanted. What changes is who builds the

bridge. In retrieval it is the system or the librarian, whereas in conversational generation users often face the model alone.

Current evidence suggests that technical wording matters little. Raising the specificity of prompt vocabulary had no significant overall effect on STEM, legal, and medical question answering (Schreiter, 2025). Among beginning programmers, the information content of a prompt predicted success, whereas technical vocabulary showed only a correlational link with failure (Lucchetti et al., 2025). Rewriting medical terms into lay language lowered public-health question-answering accuracy by only 1.4 percentage points (Zhao & Yang, 2026). In medical question answering, technical versus plain style did not interact with question framing (Yun et al., 2026). When people analyzed household energy use with a model, groups formed by self-rated domain knowledge and AI literacy differed on only one of 20 measures (Jung et al., 2026). If these results generalize, users need not learn a field's vocabulary.

In all these studies the answer is fixed by the task content, so rewording neither adds nor removes information. Deliverable tasks are different, because their conventions are not in the prompt, an outsider may not know them, and only the name could bring them in. Outsiders may well lack such names, and novices in text-to-image prompting, for example, lacked style-specific vocabulary (Oppenlaender et al., 2025). Yet to our knowledge no study has compared, under control and scored by convention coverage, what a model produces from the name of a deliverable and from a description of it.

We frame the role of a name as a chain. Practitioners turn recurrent situations into genres and methods and give them names (C. R. Miller, 1984; Swales, 1990). A name lets a set of conventions travel in one word (Brown & Lenneberg, 1954). If a name compresses a set of conventions, what it adds over a description should be what the description cannot express. Models have seen names and conventions together in human text, so saying the name may decompress the set. How much it adds should then depend on how much the model fills in on its own. The chain closes with the user. That outsiders lack the name is a premise we do not test, so we ask instead whether they can get it from the model, either by asking on the spot or by picking it up in use.

We ask three research questions (RQs). RQ1: Does naming a deliverable, rather than describing it, raise the coverage of its conventions in LLM outputs, across models and on terms sampled by rule? RQ2: What does the name carry, for which models does it matter most, and can its value be estimated before generation? RQ3: Can users who lack the name obtain it from the model?

Our contributions are threefold. First, we show that the name raised convention coverage in all six models on 40 curated terms and in all four models on 150 rule-sampled terms, and we explain why this fits null results in question answering (RQ1). Second, we show that the gain was largest for method names and in models with low lay coverage, and that a generation-free probability measure correlated moderately with per-term gains on the curated terms (RQ2). Third, we show that using a name obtained from the model recovered about half to three quarters of the gain, mostly when the name was right, and we document domain terms passing from an assistant to its user in one user's logs (RQ3).

2 Related work

2.1 Names, genres, and the vocabulary problem

Frame semantics holds that a word evokes a structure of background knowledge, not only a referent (Fillmore, 1982). Genre theory carries this idea to texts, where a genre is a typified response to a recurrent rhetorical situation (C. R. Miller, 1984) and a discourse community owns a set of genres and a specialized lexis (Swales, 1990). Organizations likewise maintain genre repertoires that their members routinely enact (Orlikowski & Yates, 1994). Applied to LLM output, genre theory shows that a model can label its text as a genre without accomplishing the genre's social action (Banting, 2026). We measure conventions only at the level of form and required elements.

The other side of naming is codability. A concept with a short shared name is communicated more accurately within a community (Brown & Lenneberg, 1954), and what is codable differs across communities (Majid et al., 2018). A practitioner name is thus a community's compressed shorthand for a set of conventions, and outsiders who lack the shorthand must assemble a description. In Putnam's account, such names are products of a division of linguistic labor (Putnam, 1975), in which many speakers use them but a few experts fix what they refer to.

Information retrieval has long recognized that users often cannot specify what they need, because a need arises from an anomalous state of knowledge (Belkin, 1980). Domain experts' web searches differ from non-experts' in vocabulary, sites visited, and success (White et al., 2009). Retrieval answers the vocabulary side of this problem with thesauri and query expansion (Carpineto & Romano, 2012) and with reference librarians who translate a patron's description into search terms (Taylor, 1968). Both assume that a field's names are the entry point to its knowledge and that the system or an intermediary supplies them. We ask whether the same entry point exists for generative models and whether users can obtain it from the model itself.

2.2 Prompt wording and its prediction

Prompt wording changes outputs (Leidinger et al., 2023; Wahle et al., 2024), although prompt-based models can learn as fast from irrelevant instructions as from good ones (Webson & Pavlick, 2022). Direct comparisons of technical and lay wording use tasks with reference answers and find null or small effects (Lucchetti et al., 2025; Schreiter, 2025; Yun et al., 2026; Zhao & Yang, 2026). In generation tasks, a correlational analysis found that instructions with domain terminology performed better, although terminology was confounded with explicit action directives (Xie et al., 2026). Naming the algorithm was the most reliable cue for steering algorithm choice in code generation (Narula et al., 2026), and genre words were the most reliable channel from prompt to output in music generation (Park & Arthur, 2026). Closest to our setting, an evaluation on financial deliverables found that roles with sparse model priors fell much further short of professional rubrics, but it did not manipulate wording (FinProBench; B. Wang et al., 2026). The idea that a cultural anchor carries a complex intent appeared as the memetic proxy in prompt programming (Reynolds & McDonell, 2021), and control codes for text types were built into a conditional language model at training time (Keskar et al., 2019). We ask whether names already in natural language play this role at inference time.

Model performance often tracks pretraining frequency (Merullo et al., 2025; Razeghi et al., 2022; Udandarao et al., 2024). Other work scores prompts with the model's own probabilities. Examples include mutual information between input and output (Sorensen et al., 2022), the log-probability of a desired answer (Yang et al., 2024; Zhou et al., 2023), prompt perplexity (Gonen et al., 2023), and pointwise mutual information (PMI) between context and question (T. Liu et al., 2025). Across semantically equivalent prompts, however,

neither perplexity nor word frequency explained performance (Leidinger et al., 2023). Our predictor differs in both target and use, since it scores convention statements rather than a reference answer and predicts how a wording effect varies across terms rather than selecting a prompt.

2.3 Users, AI literacy, and checklist evaluation

Non-experts design prompts opportunistically (Zamfirescu-Pereira et al., 2023), and beginning programmers struggle to write and revise prompts (Nguyen et al., 2024). Text-to-image users lack style vocabulary (Oppenlaender et al., 2025), and word palettes lead novices to use more domain terms (Tao et al., 2025). Prompting skill predicts output quality (Knoth et al., 2024), and so does broad subject knowledge (Bierbrauer et al., 2026). Users without domain knowledge cannot judge model-supplied terminology (Heinisch, 2025), and rewriting non-experts' questions into a profession's framing has been proposed as a task (Atwell et al., 2026). AI literacy has been defined as a set of competencies for evaluating, communicating with, and using AI (Long & Magerko, 2020). The OECD and European Union framework for schools describes prompting in terms of context, specificity, and constraints but does not list domain vocabulary as a competence (OECD & European Union, 2026).

Expert rubrics are used to evaluate professional text (ProfBench; Z. Wang et al., 2026). Decomposing evaluation into binary checklist questions improves the consistency of model judges (Cook et al., 2024; Lee et al., 2025), although model judges can recognize and favor their own generations (Panickssery et al., 2024). We adopt checklist coverage as the outcome and use it to compare two ways of asking for the same deliverable.

3 Methods

3.1 Design and term sets

We used a model-only design because the question concerns what a model does with a fixed request, which can be manipulated exactly only when the request text is controlled. We call each target deliverable a term, and the expression practitioners use for it its name. Each term was paired with one deliverable task written as a request with a slot, for example "Create a {X} for launching a new company website." Filling the slot with the name or one of two lay descriptions gave three versions that differed only in that phrase (Table 1, Fig. 1). Each output was scored against the term's convention checklist, coverage was the share of items met, and the naming gain Δ was the coverage of the name version (name coverage) minus the mean coverage of the two lay versions (lay coverage). Δ is a difference in proportions, reported as such or in percentage points.

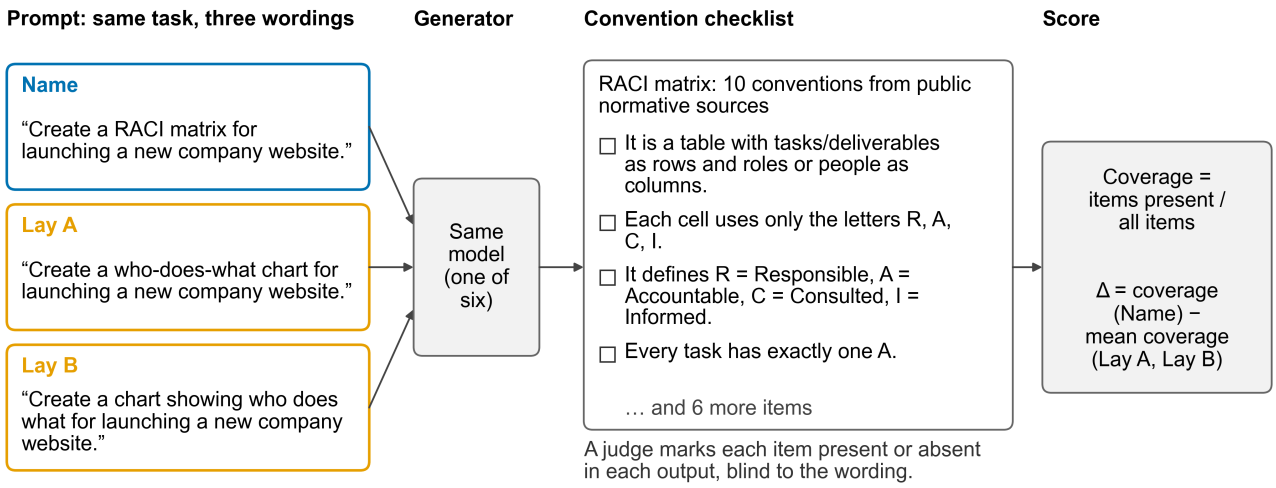


Fig. 1. Study design. One request (RACI matrix, launching a company website) is written in three versions that differ only in the slot phrase. Each tested model answers each version, and a judge blind to the version scores each output against the term's convention checklist (four of its items shown). Coverage is the share of items met. The naming gain Δ is name coverage minus the mean coverage of the two lay versions. In the curated set each version was also crossed with a rich and a thin scenario (not shown).

Table 1. Prompt versions. All versions of a term share the task sentence and differ only in the slot phrase. Examples are from the curated set.

Version	Slot content	Example (RACI matrix)	Mean prompt length, rich scenario (words)
Name	Practitioner name	Create a RACI matrix for launching a new company website.	15.2
Lay A	Short generic label (open-weight model)	Create a who-does-what chart for launching a new company website.	15.8
Lay B	Description with purpose (separate model instance)	Create a chart showing who does what for launching a new company website.	20.5

We ran this design on two samples of terms (Table 2). The curated set had 40 English terms that we chose in three pre-specified groups. These were 10 method names such as SWOT analysis and Gantt chart, 10 specialist genres such as term sheet and discharge summary, and 20 common workplace genres such as pitch deck and press release. Group labels were fixed when the terms were chosen and were not revised afterwards, so the common group retains SOAP note and the procedure name Whipple procedure even though both have fixed forms. Keeping them there works against the expected advantage of method names. The curated set was tested on six models, and each task had a rich scenario giving setting and purpose and a thin scenario stripped of genre cues, such as "Write a {X} for a 60-seat bistro" instead of "Write the checklist for a {X} before dinner service in a 60-seat bistro." Its main outcome Δ_{all} averages the rich-scenario and thin-scenario gains (Δ_{rich} and Δ_{thin}). Curated terms may favor large effects, and only 7 of the 40 appeared in the rule-based frame described next. The sampled set therefore drew 150 terms by a pre-specified rule and was tested on four models in the rich scenario.

Table 2. The two term sets. Model details are in Section 3.4.

	Curated set	**Sampled set**
Selection	Chosen by us	Stratified random sample from O*NET and Wikidata candidates
Terms	40	150
Groups	10 method names, 10 specialist genres, 20 common workplace genres	50 method names, 49 specialist genres, 51 common genres
Checklist items (per term)	442 (8–12, median 11)	1,721 (7–12, median 12)
Normative sources	119 (2–3 per term)	422 (2–5 per term)
Scenarios	Rich and thin	Rich
Tested models	27B, Haiku, Sonnet, Opus, GPT-6.1-Sol, GPT-6-Luna	Haiku, Sonnet, Opus, GPT-6.1-Sol
Outputs	1,680 (480 from 27B, 240 per closed model)	1,800 (450 per model)

3.2 Rule-based sampling

We extracted noun phrases after production verbs such as write, draft, and prepare from task statements in the O*NET 31.0 database (National Center for O*NET Development, 2026). We added Wikidata items that are instances of classes such as document, method, diagram, and plan (Vrandečić & Krötzsch, 2014; Wikidata, 2026), which gave 4,665 candidates after merging. A term was eligible if its deliverable is text or a text-expressible diagram and if public normative sources from at least two publishers define it. Its name also had to be attested in a large web corpus, and the term had to admit a lay description sharing no content word with the name. Two independent blind model judges screened each candidate for eligibility, type, and domain, and a third adjudicated disagreements. A term was classed as a method name when its name points to a fixed internal structure, such as an acronym's components or a noun like matrix or canvas, and as a genre otherwise. From the 580 eligible terms we drew 50 method names and 100 genres in quotas across six domains, and we split genres at the median corpus count into common and specialist. All criteria were fixed before generation (flow in Table B.1), and Appendix B gives the protocol, agreement, replacements, and deviations.

3.3 Lay descriptions and checklists

Each term had two lay descriptions, both of which had to avoid the name and its content words, say only what kind of thing the deliverable is, and fit the slot. Lay A, written by the open-weight model (27B, Section 3.4), was usually a short label such as "schedule" (mean 2.6 words). Lay B, written by a separate Claude instance that knew neither the study's aim nor the checklists, usually stated a purpose, such as "timeline showing which tasks happen when" (mean 7.25 words). Names averaged 1.93 words. Averaging the two styles reduces dependence on one way of paraphrasing, although both were written by models rather than by real outsiders.

Each checklist was extracted from public normative sources such as government and university guides, professional bodies, practitioner guides, and standards. A Claude instance following a fixed protocol extracted the conventions that distinguish the deliverable from ordinary text, covering structure, required sections, order, format, tone, and required elements. It recorded a supporting quotation for each and merged them into binary items linked to their sources (example in Appendix A). Checklists were frozen before

generation in the curated set and before judging in the sampled set, and builders never saw outputs. Sampled-set materials followed the same protocol, with differences listed in Appendix B.

3.4 Models, generation, and scoring

The curated set was run on six models. These were an open-weight Qwen3.8-27B variant with its refusal direction removed (Huihui-Qwen3.8-27B-abliterated, hereafter 27B), Anthropic's Claude Haiku 5.5, Sonnet 5.5, and Opus 5.5, and OpenAI's GPT-6.1-Sol and GPT-6-Luna. Removing the refusal direction may alter the model in other ways, so conclusions do not extend to the original weights. We served 27B locally with llama.cpp at temperature 0.7, with reasoning disabled and two outputs per cell. Claude models ran as agent instances in Anthropic's command-line agent environment, and OpenAI models ran through OpenAI's command-line agent client, both at high reasoning effort. Each Claude instance answered 40 requests, and a Latin square ensured that no instance saw two versions of a term, whereas each OpenAI call ran in a fresh context. Neither route is a bare API, since both add their environment's system prompt and neither lets us set the temperature. Closed models produced one output per cell (term × version × scenario). The sampled set used Haiku, Sonnet, Opus, and GPT-6.1-Sol under the same settings, and Appendix C gives further details.

Judges saw only the output and the checklist and scored an item 1 only when the text clearly does what the item says (Appendix A). Sonnet instances judged all closed-model outputs blind to version, and in the sampled set also blind to model, while 27B judged its own outputs blind to version. Without a human gold standard, we checked reliability in three ways (Table 3). Two Sonnet judges rescored 120 of 27B's outputs, and Sonnet and GPT-6.1-Sol rescored 60 curated-set outputs with each "yes" requiring a verbatim quotation. A script found 415 of 415 Sonnet quotations and 361 of 364 GPT-6.1-Sol quotations in the outputs. GPT-6.1-Sol judged more strictly than Sonnet but in the same direction, giving name coverage 0.62 and lay coverage 0.48 on these outputs. An independent Opus judge rescored a subsample of the sampled set. All judges are language models, and Claude judges scored outputs that included Claude outputs, so self-preference is a risk (Panickssery et al., 2024).

Table 3. Judge agreement on binary checklist items. Original judgments are Sonnet's for closed-model outputs and 27B's own for 27B outputs.

Comparison	Outputs	Items	Agreement	κ
27B self-judgment vs two Sonnet judges (curated set, thin scenario)	120	1,326	—	0.77
Original vs Sonnet with quotations (curated set)	60	661	0.941	0.873
Original vs GPT-6.1-Sol with quotations (curated set)	60	661	0.825	0.639
Sonnet with quotations vs GPT-6.1-Sol (curated set)	60	661	0.823	0.637
Sonnet vs Opus (sampled-set subsample)	160	1,837	0.928	0.835

3.5 Robustness checks

Three checks addressed alternative explanations in the curated set. First, to test whether the judge rewards the name, we replaced the name and its variants in closed-model outputs by `[TERM]`, using a variant list fixed before reading any output, and fresh Sonnet judges rescored them. We also dropped items whose text contains the name or its distinctive part. Second, to test whether the checklists reflect their builder, two Opus instances without access to the originals or to results rebuilt the 40 checklists. Sonnet then rejudged the outputs of 27B, Sonnet, and Opus against them, which also put 27B under a judge from another family. Third, to test whether the gain is only length, we regressed the item score within each model on the name version, log output words, and term fixed effects, using an item-level linear probability model with a term-clustered bootstrap. In the sampled set, outputs for half of the terms (75) were rescored with the name masked, and sensitivity analyses dropped terms with known material issues (Appendix B).

3.6 Analyses

Structure coding. An Opus instance without access to results coded each checklist item as structure and format (S, such as "four quadrants"), content (C, such as "states the risks"), or tone, length, and audience (T). It also judged whether each curated lay description named the deliverable's structure. Second coders agreed at $\kappa = 0.69$ on 90 curated items and at $\kappa = 0.80$ on 287 sampled items in the final set. The confirmatory test compared, within terms, the gain on S items with the gain on C items.

Generation-free predictor. On 27B we read the token log-probabilities of each checklist item after a prefix such as "Notes on what a good {name} looks like:" and after the same prefix with each lay description in place of the name (Appendix A). A term's PMI was the mean, over its checklist items, of each item's log-probability after the name prefix minus its mean log-probability after the two lay-description prefixes. PMI uses only the checklist and the model, with no generation or judging. Closed models do not expose log-probabilities, so we tested whether PMI from 27B, computed for the curated set only, predicts per-term gains in other models.

Two-step naming. On Sonnet and GPT-6.1-Sol, the model first saw each rich-scenario lay prompt of the curated set and was asked for the standard name practitioners use for the deliverable (Appendix A). The returned name was inserted into the same task sentence, and the same model generated in a new context. A separate model instance judged whether each returned name was correct, counting partial matches as wrong. In the sampled set, we repeated the procedure on Sonnet for all 300 lay prompts. We report recovery in two ways and treat the first as primary. The unselected ratio divides the mean two-step gain (two-step coverage − coverage of that lay prompt) over all lay prompts by the same model's mean rich-scenario naming gain. The per-prompt ratio, (two-step coverage − lay coverage) / (name coverage − lay coverage), is unstable when the denominator is near zero. It was therefore computed only where name coverage exceeded lay coverage by more than 0.05. This selection biases it upward, because the lay output that decides selection also enters the ratio and regression to the mean inflates the apparent recovery.

Lexical predictors. We also related the gain to corpus frequency (J. Liu et al., 2024; Soldaini et al., 2024; Team OLMo et al., 2025) and to lexical resources and word prevalence norms (Baker et al., 1998; Brysbaert et al., 2019; International Computer Science Institute, n.d.; G. A. Miller, 1995), with details in Appendix C.

Usage case. We analyzed our own multi-month logs of conversations with an LLM-based coding and writing assistant. The user is a non-native English speaker who writes mostly in another language, so English terms

can be extracted mechanically. Here "term" has its everyday sense of a word or phrase from a field's vocabulary, not an item of the two term sets. We extracted candidate terms that the user typed at least three times, excluding pasted text and tool output, and a Sonnet instance classified them, leaving 65 domain terms. A term counted as adopted if the assistant used it first and at least three times and the user later typed it at least three times. The analysis is descriptive, and only aggregate counts are released.

Statistics. Coverage and Δ were computed within terms and bootstrapped over terms (percentile bootstrap, 5,000 resamples). Correlations are Spearman ρ with term-level bootstrap confidence intervals (CIs). For both sets, the main criteria were written in dated internal analysis plans before the corresponding outputs were generated. Pooling across models and the cross-model account in Section 4.2 are exploratory. The plans were not publicly registered, and deviations are listed in Appendix B. Language models also wrote lay descriptions and thin scenarios, extracted checklists, and did all judging, coding, screening, and classification, and their inputs and outputs are in the supplementary material.

4 Results

4.1 Naming raises convention coverage (RQ1)

The name raised coverage in every model and in both sets (Table 4, Fig. 2). In the curated set, Δ_{all} ranged from +0.112 for Opus to +0.293 for GPT-6.1-Sol, and every lower bound was above zero. The pooled gain was positive for 39 of 40 terms. In the sampled set, gains ranged from +0.112 for Opus to +0.194 for GPT-6.1-Sol, and every lower bound again exceeded zero, as the pre-specified criterion required. The name version also beat each lay version separately (Table 5). For the three Claude models, sampled-set gains were close to their curated rich-scenario gains (Fig. 2), whereas GPT-6.1-Sol gained less on sampled terms. The effect therefore does not depend on curation, although curation may have enlarged it for GPT-6.1-Sol.

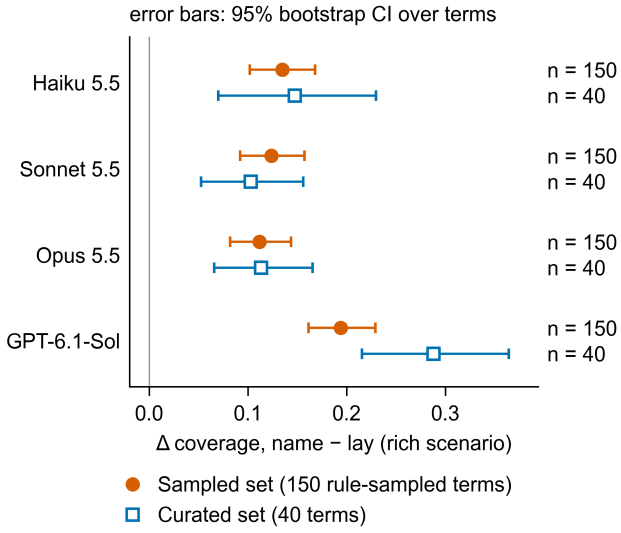


Fig. 2. Naming gain by model in the sampled set and the curated set, rich scenario (Δ_{rich} for the curated set, which differs from the Δ_{all} in Table 4). Points are means with 95% bootstrap CIs over terms (5,000 resamples). Sampled set: n = 150 terms per model. Curated set: n = 40 terms. Only the four models tested in both sets are shown.

Table 4. Main effect by model. Coverage is the share of checklist items met. Curated set: coverage averages rich and thin scenarios and, for lay, the two lay versions, and Δ_{all} has 95% bootstrap CIs over 40 terms. Sampled set: rich scenario, 150 terms. "Terms $\Delta > 0$" counts terms with a positive gain.

Model	Curated: name coverage	Curated: lay coverage	Curated: Δ_{all} [95% CI]	Curated: terms $\Delta > 0$	Sampled: name coverage	Sampled: lay coverage	Sampled: Δ [95% CI]	Sampled: terms $\Delta > 0$
27B	0.739	0.566	+0.173 [0.116, 0.234]	35	—	—	—	—
Haiku 5.5	0.729	0.573	+0.156 [0.089, 0.232]	25	0.714	0.580	+0.135 [0.102, 0.168]	100
Sonnet 5.5	0.809	0.667	+0.142 [0.091, 0.200]	30	0.836	0.712	+0.124 [0.092, 0.157]	95
Opus 5.5	0.832	0.720	+0.112 [0.066, 0.162]	31	0.889	0.777	+0.112 [0.082, 0.144]	93
GPT-6.1-Sol	0.693	0.400	+0.293 [0.222, 0.365]	37	0.640	0.446	+0.194 [0.161, 0.229]	114
GPT-6-Luna	0.616	0.349	+0.267 [0.202, 0.333]	36	—	—	—	—
Pooled (6)	—	—	+0.190 [0.141, 0.245]	39	—	—	—	—

No robustness check changed the conclusion (Table 5). Masking the name moved the closed models' curated gains by at most 0.011 and their sampled gains by 0.010 or less. Dropping items that contain the name, rebuilding the checklists, and controlling for output length each left every curated point estimate above zero, and every length-controlled lower bound was above zero. Term-level gains under original and rebuilt checklists correlated at $\rho = 0.64$ to 0.81 across the three rejudged models. In the sampled set, sensitivity analyses left gains between 0.108 and 0.199. They dropped the 22 terms whose checklist items were not written in English, the terms whose Lay A came from another model, or the one term that overlaps the curated set (Appendix B). These checks make it unlikely that the gain reflects the judge seeing the name, items spelling out the name, one particular checklist, or longer text.

Table 5. Robustness checks. Curated set (Δ_{all}): Masked, closed-model outputs rescored with the name masked. Name items dropped, items containing the name or its distinctive part removed. Rebuilt, independently rebuilt checklists judged by Sonnet. For 27B the rebuilt value uses the first output per cell, where the original judgments give 0.146. Length-controlled, name-version coefficient with log output length and term fixed effects (term-clustered bootstrap, 2,000 resamples). Sampled set (rich scenario): Masked half compares unmasked and masked judgments on the same 75 terms. Name − Lay A and Name − Lay B give the gain over each lay version.

Model	Curated: masked Δ	Curated: name items dropped Δ	Curated: rebuilt Δ	Curated: length-controlled [95% CI]	Sampled: masked half, unmasked → masked	Sampled: name − Lay A	Sampled: name − Lay B
27B	—	0.171	0.172	0.146 [0.085, 0.207]	—	—	—
Haiku 5.5	0.149	0.154	—	0.146 [0.082, 0.221]	0.118 → 0.113	0.151	0.119
Sonnet 5.5	0.148	0.142	0.153	0.131 [0.083, 0.188]	0.085 → 0.095	0.140	0.107
Opus 5.5	0.101	0.107	0.111	0.104 [0.057, 0.158]	0.090 → 0.093	0.137	0.086
GPT-6.1-Sol	0.295	0.288	—	0.226 [0.151, 0.302]	0.171 → 0.177	0.212	0.176
GPT-6-Luna	0.260	0.267	—	0.203 [0.139, 0.271]	—	—	—

4.2 What the name carries and for which models (RQ2)

Lay descriptions conveyed purpose or medium but rarely structure. Only 12 of the 80 curated descriptions named any structure, and even these named only a fragment, as when “chart showing who does what” gives the axes of a RACI matrix but not the rule of one A per task. Gantt chart became “schedule” and user story “description”. We had pre-specified that terms whose descriptions named a structural fragment would gain less. Instead, the 9 such terms (12 descriptions in all) gained more than the other 31 (0.30 against 0.16). Six of the 9 are method names or specialist genres, so a fragment may mark a large fixed structure of which a short description captures only part.

The gain followed term type in both sets (Table 6, Fig. 3). Method names gained most in both sets, and method names minus common genres was +0.170 [0.034, 0.319] and +0.145 [0.079, 0.212], meeting the sampled set’s pre-specified criterion. The largest gains were for terms whose deliverables have a fixed structure or prescribed notation, such as SWOT analysis and term sheet in the curated set and IDEF0 and program evaluation and review technique in the sampled set. Near-zero gains clustered on common genres such as pitch deck and hot wash. For these, the lay description or the task sentence may already convey the purpose or medium, or the deliverable may have no fixed form. Specialist genres gained more than common genres in the curated set but not in the sampled set (+0.013 [−0.034, 0.061]), although the two sets drew this line differently, by our judgment in the curated set and by median corpus count in the sampled set.

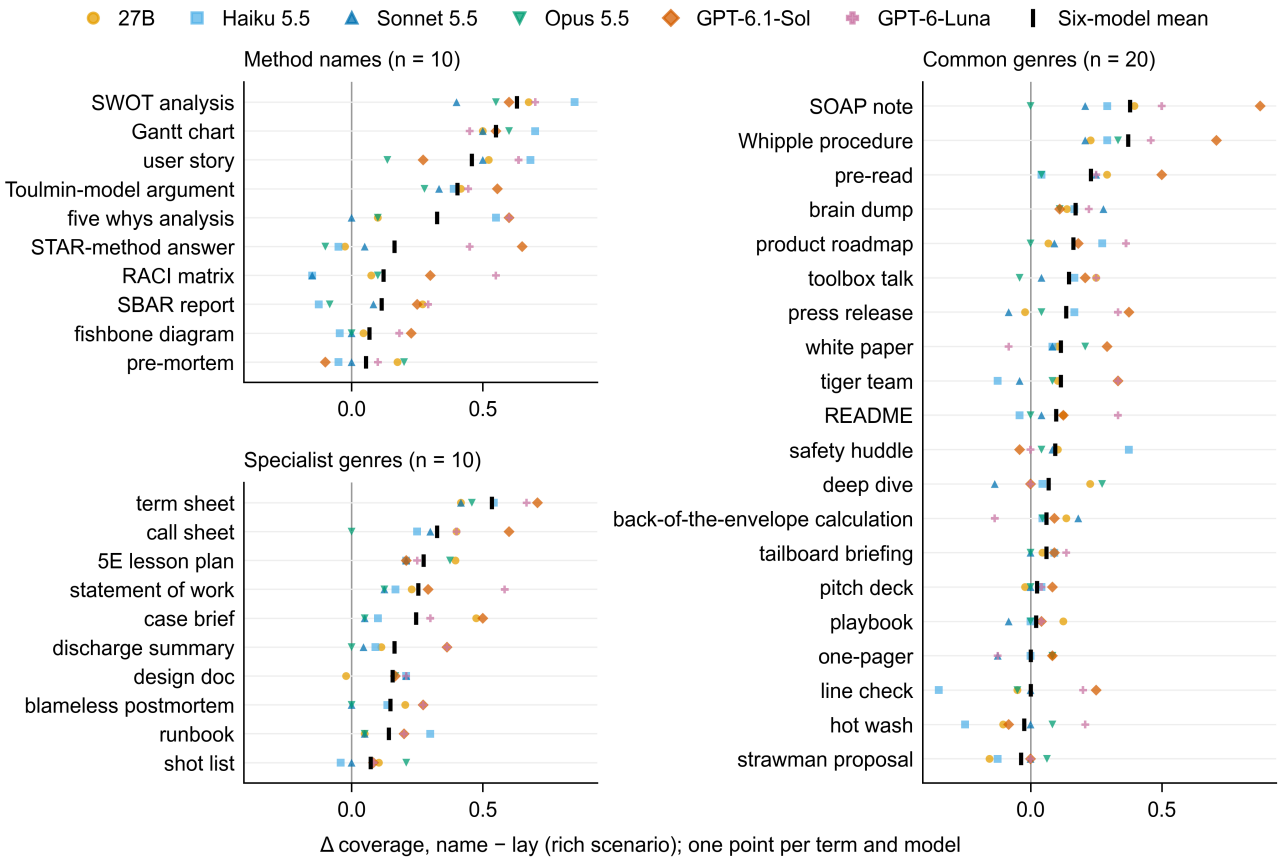


Fig. 3. Naming gain per term in the curated set (rich scenario, Δ_{rich}; group means and term values given in the text are Δ_{all}, which also averages the thin scenario). Colored markers are individual models and black bars the six-model mean. Terms are grouped into method names (n = 10), specialist genres (n = 10), and common workplace genres (n = 20, including one procedure name) and sorted by the mean within group. Each marker is a single term-model value (one output per version for closed models, the mean of two for 27B), so no intervals are shown.

Table 6. Naming gain by term type and checklist item class. The curated set pools six models over Δ_{all} (exploratory pooling). The sampled set pools Sonnet, Opus, and GPT-6.1-Sol in the rich scenario (Appendix B). Brackets are 95% bootstrap CIs over terms (5,000 resamples). For curated groups, CIs are given only for the pre-specified contrast.

	Curated set	**Sampled set**
Method names	0.290 (n = 10)	0.235 [0.180, 0.294] (n = 50)
Specialist genres	0.231 (n = 10)	0.104 [0.071, 0.140] (n = 49)
Common genres	0.120 (n = 20)	0.091 [0.059, 0.126] (n = 51)
Method − common	+0.170 [0.034, 0.319]	+0.145 [0.079, 0.212]
Structure items, share of checklist	58 of 442 (13%)	534 of 1,721 (31%)
Gain on structure items	0.319 (n = 26)	—
Gain on content items	0.180 (n = 26)	—
Structure − content (paired within terms)	+0.139 [0.066, 0.216] (n = 26)	+0.020 [−0.015, 0.054] (n = 134)

The name lifted more than structure alone. In the curated set, the gain on structure items exceeded the gain on content items by +0.139 [0.066, 0.216]. The name added quadrants, role letters, the "As a… I want… so that…" template, and time bars. In the sampled set the difference was +0.020 [−0.015, 0.054], and the pre-

specified criterion failed, although structure items made up a larger share of those checklists. Structure was the most visible part of what a curated name carried, but sampled names lifted structure and content alike.

A name may also need context to fix its sense. For the polysemous line check, 27B's name coverage fell from 0.55 (rich) to 0.10 (thin). Otherwise thinning mattered little, and the pre-specified expectation that names matter more in thin scenarios held only for Sonnet.

The size of the gain tracked the lay baseline (Table 4). Lay coverage ranged from 0.35 to 0.72 across models, with the two GPT models lowest, while name coverage differed less. The GPT models also wrote shorter outputs, averaging 437 and 375 words against 1,014 and 1,213 for Sonnet and Opus. In the sampled set, GPT-6.1-Sol again had the lowest lay coverage and the largest gain. One account is that models which elaborate vague requests supply many conventions unprompted and leave less for the name to add. This account is exploratory and confounded with output length and with the vendors' system prompts. The ranking of terms was more stable, as term-level gains correlated across the 15 model pairs at a median ρ of 0.55. In an exploratory comparison of tiers, Haiku given the name did not differ detectably from Opus given a lay description on curated terms but fell short on sampled terms, so naming does not reliably substitute for a stronger model (Appendix C.4).

The binding between names and conventions was visible without generation, since on 27B convention statements were more probable after the name than after the lay descriptions for 37 of the 40 curated terms. PMI correlated moderately with the six-model mean gain (ρ = 0.37 [0.04, 0.63], Fig. 4). Per model, lower bounds were above zero for 27B, Haiku, GPT-6.1-Sol, and GPT-6-Luna but not for Sonnet or Opus. Which models cleared zero changed when Δ_{rich} replaced Δ_{all}, whereas dropping the few items that contain the name left PMI nearly unchanged (Appendix C.3). Predictors that do not consult a model, such as corpus frequency and dictionary coverage, did not track the gain (Appendix C.5).

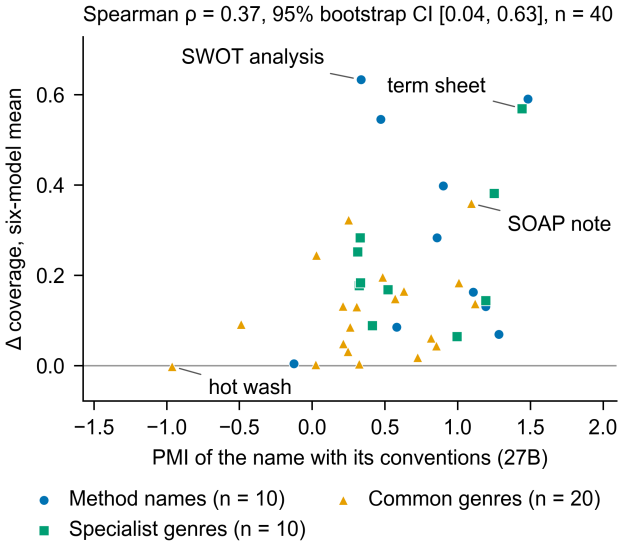


Fig. 4. Generation-free PMI computed on 27B against the six-model mean Δ_{all} in the curated set (n = 40 terms). Colors and shapes mark term groups. Labeled terms are the largest gain in each group and the smallest gain overall. Spearman ρ = 0.37, 95% bootstrap CI [0.04, 0.63] (5,000 resamples).

4.3 Getting the name from the model (RQ3)

Asked to name the deliverable behind a lay request, models returned the practitioner name for 63–70% of requests, fewer on sampled than on curated terms (Table 7). Generating with the returned name recovered 0.72 and 0.73 of the curated rich-scenario naming gain for Sonnet and GPT-6.1-Sol, and 0.49 of Sonnet's gain on the sampled set. Recovery depended on the name being right. In the sampled set, mean recovery on selected prompts was 0.82 with a correct name and 0.11 [−0.14, 0.36] with a wrong one, and the curated set showed the same split. The curated correct-name ratios above 1 reflect noise between generations and the upward bias of selection. Wrong names were real practitioner names for other convention sets, such as run of show for Gantt chart. An outsider has little chance of noticing such an error, so a name obtained from the model needs checking.

Table 7. Two-step naming (rich scenario). Brackets are 95% bootstrap CIs. The unselected ratio divides the mean two-step gain over all lay prompts by the model's rich-scenario naming gain. Per-prompt recovery ratios use only the selected prompts, where name coverage exceeded lay coverage by more than 0.05 (in the sampled set this selects the same prompts as any positive gap, Appendix B.6), a selection that biases them upward (Section 3.6). In the sampled set, naming accuracy was 0.46 by string matching alone.

	Sonnet 5.5, curated set	**GPT-6.1-Sol, curated set**	**Sonnet 5.5, sampled set**
Correct names	56 of 80	54 of 80	189 of 300
Two-step − lay coverage	+0.074 [0.026, 0.120]	+0.211 [0.152, 0.273]	+0.061 [0.041, 0.084]
Unselected ratio (two-step gain / naming gain)	0.72 (0.074 / 0.103)	0.73 (0.211 / 0.288)	0.49 (0.061 / 0.124)
Selected prompts	42	65	163
Recovery ratio, median / mean	1.00 / 0.86	1.00 / 0.78	0.67 / 0.56
Mean recovery ratio, correct name (n)	1.37 (27)	1.05 (43)	0.82 (103)
Mean recovery ratio, wrong name (n)	−0.06 (15)	0.26 (22)	0.11 (60)
Example wrong names	Gantt chart → run of show	deep dive → root cause analysis, term sheet → pitch deck	—

The usage case shows a second route, in which 46 of 65 domain terms in one user's logs were used first by the assistant and later by the user. The median interval between the assistant's first use of a term and the user's first use was 34 days, and the assistant had used a term a median of 9.5 times before the user did. Most adopted terms were concepts, such as novelty and ground truth, and only 12 were deliverable names, such as deck and playbook. When we restricted the count to terms first seen after a fixed cutoff, which removes terms the user may have known beforehand, 19 of 24 were adopted. The case shows that domain terms, including some deliverable names, flow from model to user. It does not show that knowing them makes later work more effective.

5 Discussion

5.1 Why the name matters for deliverables but not for answers

Our results and the null results on question answering follow one principle. A word in a prompt matters when it brings information that the rest of the prompt lacks (Lucchetti et al., 2025). In question answering the answer is fixed by the question, so a plainer phrase leaves the information intact (Schreiter, 2025; Zhao &

Yang, 2026). In deliverable tasks the conventions are not in the prompt, and a lay description conveys a purpose while the name points to a convention set. The gain therefore appears where descriptions carry least. It was largest for method names, whose deliverables follow prescribed notation, and smallest for common genres, whose descriptions already say what the deliverable is for. The small gain on common genres is the counterpart, within deliverable tasks, of the question-answering null.

The sampled set sharpens what a name compresses. On curated terms the name lifted structure items more than content items, although structure made up only 13% of those checklists, and on sampled terms structure and content gained alike. The general claim therefore concerns the whole convention set rather than layout. The same principle may explain why corpus frequency and dictionary coverage failed as predictors while the model's own probabilities on convention statements correlated moderately with the mean gain. The gain appears to depend less on how often a model has seen the name than on whether a convention set stands behind it and whether the model supplies that set unprompted.

5.2 Models as decompressors and as dictionaries

One might expect stronger models to read intent from vague descriptions and make names unnecessary, but our data support this only in part. Within the Claude family, lay baselines rose with capability, but Opus still gained 11 percentage points in both sets. The GPT models gained about twice as much as the Claude models in the curated set, tracking their lower lay baselines and shorter outputs. The value of a name thus appears to depend on whether a model writes a convention set out unprompted, not only on whether it knows the set. Elaboration depends on training, system prompts, and product choices about brevity. Users control none of these, and all of them change across releases. Naming narrowed this dependence, since name coverage varied less across models than lay coverage did.

Practitioner names are products of the division of linguistic labor described in Section 2.1, and an LLM occupies both sides of it. As a decompressor it stands with the experts, consistent with a binding learned from practitioner text. The positive PMI for most curated terms shows that the binding is present, not where it came from. As a dictionary it maps descriptions to names. These functions used to belong to different people, since experts held the reference while teachers, colleagues, and manuals passed names to newcomers. One system now does both, with a new risk. In retrieval terms, the model acts as thesaurus and reference librarian, but its mapping has no editor, and a wrong name looks as credible as a right one.

5.3 Implications for users, system design, and AI literacy

For users, the results suggest simple rules. When the name is known, say it, especially for methods and genres with fixed structure. When it is not, ask the model what practitioners call the deliverable, use that name, and check it against a definition, because three to four in ten returned names were wrong. For common genres a clear purpose may suffice, and when a name has several senses a line of context fixes the intended one.

For system design, the two-step procedure can be built in. When a request seems to point to a named deliverable without naming it, a system can propose candidate names with one-line definitions and ask for confirmation. This follows the logic of query suggestion and of word-palette interfaces (Tao et al., 2025). A

generation-free indicator such as PMI might help find the terms in a domain with the largest gains, once it has been tested on sampled terms.

For AI-literacy education, the target is a meta-skill rather than each field's conventions. Learners should know that many deliverables have names standing for convention sets, ask for a name when they do not know it, and check it. The OECD and European Union framework does not include this competence (OECD & European Union, 2026). Our results give it a quantitative basis, since in every model tested the name added 11 to 29 percentage points of convention coverage, with the lowest 95% lower bound at 7 percentage points.

5.4 Limitations

Several pre-specified expectations failed, and they bound the conclusions. These were the thin-scenario effect beyond Sonnet, smaller gains for lay descriptions with a structural fragment, the structure advantage in the sampled set, and PMI for Sonnet and Opus. The method-minus-common contrast was individually significant in only three of six curated-set models. Corpus frequency, public lexical resources, and the predicted ordering within the Claude family also failed (Appendix C).

First, all judging was done by models without human expert calibration. Cross-vendor agreement ($\kappa = 0.64$) shows that judges differ in strictness. Self-preference is a risk throughout, because Sonnet judged outputs that included its own, 27B judged its own outputs, and Claude instances built all checklists. In the curated set the direction of the gain held under every judge, but its size depends on the judge. The sampled set had no cross-vendor judging, and its Opus subsample covered all three versions for only one term–model pair. Calibration against human practitioners would fix the size of the effect.

Second, lay descriptions were written by models, and the claim that outsiders lack the name is a premise, not a finding. Real outsiders may write longer, vaguer, or more structural descriptions, which could move the gain in either direction. By design, our descriptions listed no contents, so descriptions that list some contents would likely narrow the gain. Coverage measures conformity to the named convention and does not show that a lay requester, who did not ask for that convention, is better served. A test of whether real outsiders know the names, prompts written by real outsiders, and a user experiment on teaching the ask-first procedure would close this end of the chain.

Third, closed models were accessed through vendor agent environments rather than bare APIs. Each added its own system prompt, temperature could not be set, and Claude instances handled several requests per context while OpenAI calls were made singly. These differences are confounded with the elaboration account. The open model was a modified variant, and closed models produced one output per cell. Bare APIs at fixed temperature and fixed output length would separate elaboration from length.

Fourth, only English was tested, and both term sets have limits. The curated set was chosen by us, while the sampled set covered only four models and concentrates its method names in two domains. It represents the normative vocabulary of O*NET and Wikidata rather than workplace jargon such as user story or runbook, which the frame misses. PMI was computed only on the curated set and on a model other than most of those it predicted. Testing PMI within one open model that both scores and generates, on sampled terms and in other languages, is the next step.

Finally, the two-step procedure was tested on two models in the curated set and one in the sampled set. The usage case has one person, covers only English terms, relies on one model classifier, and treats repeated use as adoption, which is not understanding.

6 Conclusion

Prior work found that technical and lay wording differ little when the answer is fixed by the question. On deliverable tasks, the name matters. Practitioners compress a whole set of conventions into a name, and a lay description conveys the purpose but not the set. Consistent with language models having learned this compression from human text, saying the name decompressed the set. The name raised convention coverage by 11 to 29 percentage points in six models on curated terms. The gain replicated in four models on 150 terms sampled by rule, where it was largest for method names. It was largest where lay coverage was lowest and outputs shortest, a descriptive pattern confounded with vendor. On the curated terms, the conditional probabilities of one open model correlated moderately with which terms gained most. That outsiders lack the name is our premise, not a finding, since no human outsiders took part. The data show that the name can be obtained by asking the model for it. Using the returned name recovered about half to three quarters of the gain, mostly when the name was correct. One user's logs also show domain terms, mostly concepts, passing from the assistant to the user. For users and for AI-literacy education, we propose as the skill to master not each field's conventions but knowing that deliverables have names, asking for them, and checking them. Future work should test this skill with real users and calibrate the measurements against human practitioners.

Data availability

The term lists, convention checklists with source citations, prompts, all model outputs and judgments, and analysis code will be deposited in a public repository with a DOI and are available from the author on request until then. Raw assistant logs from the usage case are not shared because they contain private conversation content; aggregate counts are included in the supplementary material.

CRediT authorship contribution statement

Jiawei Jing: Conceptualization, Methodology, Software, Investigation, Formal analysis, Data curation, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The author declares no competing interests.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

Large language models are the object of this study and also served as research instruments, including generating materials and judging outputs; these roles are described in Section 3. During the preparation of this work the author used Claude Opus 5.5 (Anthropic) in order to draft and edit text. After using this tool, the author reviewed and edited the content as needed and takes full responsibility for the content of the published article.

References

Atwell, K., Gray, M. A., Savelka, J., Rial, L., Linardi, S., & Alikhani, M. (2026). Asking the right questions: Can expert-prompted LLMs reformulate legal queries from non-experts? In *Proceedings of the Second Workshop on Customizable NLP: Progress and Challenges in Customizing NLP for a Domain, Application, Group, or Individual (CustomNLP4U)* (pp. 167–181). Association for Computational Linguistics. https://doi.org/10.18653/v1/2026.customnlp4u-1.15

Baker, C. F., Fillmore, C. J., & Lowe, J. B. (1998). The Berkeley FrameNet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1* (pp. 86–90). Association for Computational Linguistics. https://doi.org/10.3115/980845.980860

Banting, S. (2026). Simulated social action: Assessing generative AI's attempts at the rhetoric of literary criticism. *Discourse and Writing/Rédactologie*, *35*, 78–100. https://doi.org/10.31468/dwr.1149

Belkin, N. J. (1980). Anomalous states of knowledge as a basis for information retrieval. *Canadian Journal of Information Science*, *5*, 133–143.

Bierbrauer, C., Vogelsang, M., Staab, T., & Qayyum, A. (2026). *Ask AI, and thou shalt receive? Subject knowledge, prompting skill, and the production of AI output quality* [Preprint]. SSRN. https://doi.org/10.2139/ssrn.7432358

Brown, R. W., & Lenneberg, E. H. (1954). A study in language and cognition. *The Journal of Abnormal and Social Psychology*, *49*(3), 454–462. https://doi.org/10.1037/h0057814

Brysbaert, M., Mandera, P., McCormick, S. F., & Keuleers, E. (2019). Word prevalence norms for 62,000 English lemmas. *Behavior Research Methods*, *51*(2), 467–479. https://doi.org/10.3758/s13428-018-1077-9

Carpineto, C., & Romano, G. (2012). A survey of automatic query expansion in information retrieval. *ACM Computing Surveys*, *44*(1), 1–50. https://doi.org/10.1145/2071389.2071390

Cook, J., Rocktäschel, T., Foerster, J., Aumiller, D., & Wang, A. (2024). *TICKing all the boxes: Generated checklists improve LLM evaluation and generation* (arXiv:2410.03608) [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2410.03608

Fillmore, C. J. (1982). Frame semantics. In The Linguistic Society of Korea (Ed.), *Linguistics in the morning calm: Selected papers from SICOL-1981* (pp. 111–137). Hanshin Publishing.

Furnas, G. W., Landauer, T. K., Gomez, L. M., & Dumais, S. T. (1987). The vocabulary problem in human-system communication. *Communications of the ACM*, *30*(11), 964–971. https://doi.org/10.1145/32206.32212

Gonen, H., Iyer, S., Blevins, T., Smith, N. A., & Zettlemoyer, L. (2023). Demystifying prompts in language models via perplexity estimation. In *Findings of the Association for Computational Linguistics: EMNLP 2023* (pp. 10136–10148). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.findings-emnlp.679

Heinisch, B. (2025). Large language models for terminology work: A question of the right prompt? *Journal for Language Technology and Computational Linguistics*, *38*(2), 13–30. https://doi.org/10.21248/jlcl.38.2025.280

International Computer Science Institute. (n.d.). *FrameNet* (Release 1.7) [Data set]. Retrieved November 2, 2026, from https://framenet.icsi.berkeley.edu/

Jung, W., Jeon, K., & Babon-Ayeng, P. (2026). *Human-AI collaboration in large language model-integrated building energy management systems: The role of user domain knowledge and AI literacy* (arXiv:2602.16140) [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2602.16140

Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., & Socher, R. (2019). *CTRL: A conditional transformer language model for controllable generation* (arXiv:1909.05858) [Preprint]. arXiv. https://doi.org/10.48550/arXiv.1909.05858

Knoth, N., Tolzin, A., Janson, A., & Leimeister, J. M. (2024). AI literacy and its implications for prompt engineering strategies. *Computers and Education: Artificial Intelligence*, *6*, Article 100225. https://doi.org/10.1016/j.caeai.2024.100225

Lee, Y., Kim, J., Kim, J., Cho, H., Kang, J., Kang, P., & Kim, N. (2025). CheckEval: A reliable LLM-as-a-judge framework for evaluating text generation using checklists. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing* (pp. 15782–15809). Association for Computational Linguistics. https://doi.org/10.18653/v1/2025.emnlp-main.796

Leidinger, A., van Rooij, R., & Shutova, E. (2023). The language of prompting: What linguistic properties make a prompt successful? In *Findings of the Association for Computational Linguistics: EMNLP 2023* (pp. 9210–9232). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.findings-emnlp.618

Liu, J., Min, S., Zettlemoyer, L., Choi, Y., & Hajishirzi, H. (2024). Infini-gram: Scaling unbounded n-gram language models to a trillion tokens. In *Conference on Language Modeling (COLM 2024)*. https://doi.org/10.48550/arXiv.2401.17377

Liu, T., Qi, J., He, P., Bisazza, A., Sachan, M., & Cotterell, R. (2025). Pointwise mutual information as a performance gauge for retrieval-augmented generation. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)* (pp. 1628–1647). Association for Computational Linguistics. https://doi.org/10.18653/v1/2025.naacl-long.78

Long, D., & Magerko, B. (2020). What is AI literacy? Competencies and design considerations. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (pp. 1–16). Association for Computing Machinery. https://doi.org/10.1145/3313831.3376727

Lucchetti, F., Wu, Z., Guha, A., Feldman, M. Q., & Anderson, C. J. (2025). Substance beats style: Why beginning students fail to code with LLMs. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)* (pp. 8541–8610). Association for Computational Linguistics. https://doi.org/10.18653/v1/2025.naacl-long.433

Majid, A., Roberts, S. G., Cilissen, L., Emmorey, K., Nicodemus, B., O'Grady, L., Woll, B., LeLan, B., de Sousa, H., Cansler, B. L., Shayan, S., de Vos, C., Senft, G., Enfield, N. J., Razak, R. A., Fedden, S., Tufvesson, S., Dingemanse, M., Ozturk, O., … Levinson, S. C. (2018). Differential coding of perception in the world's languages. *Proceedings of the National Academy of Sciences*, *115*(45), 11369–11376. https://doi.org/10.1073/pnas.1720419115

Merullo, J., Smith, N. A., Wiegreffe, S., & Elazar, Y. (2025). On linear representations and pretraining data frequency in language models. In *International Conference on Learning Representations (ICLR 2025)*. https://proceedings.iclr.cc/paper_files/paper/2025/hash/b35c6f99df526a798433fabff039bd50-Abstract-Conference.html

Miller, C. R. (1984). Genre as social action. *Quarterly Journal of Speech*, *70*(2), 151–167. https://doi.org/10.1080/00335638409383686

Miller, G. A. (1995). WordNet: A lexical database for English. *Communications of the ACM*, *38*(11), 39–41. https://doi.org/10.1145/219717.219748

Narula, A., Rafique, M. B., & Bindschaedler, L. (2026). *The invisible lottery: How subtle cues steer algorithm choice in LLM code generation* (arXiv:2606.04057) [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2606.04057

National Center for O*NET Development. (2026). *O*NET 31.0 database* [Data set]. U.S. Department of Labor, Employment and Training Administration. https://www.onetcenter.org/database.html

Nguyen, S., Babe, H. M., Zi, Y., Guha, A., Anderson, C. J., & Feldman, M. Q. (2024). How beginning programmers and code LLMs (mis)read each other. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems* (pp. 1–26). Association for Computing Machinery. https://doi.org/10.1145/3613904.3642706

OECD, & European Union. (2026). *Empowering learners for the age of AI: An AI literacy framework for primary and secondary education*. OECD Publishing. https://doi.org/10.1787/65cd27d4-en

Oppenlaender, J., Linder, R., & Silvennoinen, J. (2025). Prompting AI art: An investigation into the creative skill of prompt engineering. *International Journal of Human–Computer Interaction*, *41*(16), 10207–10229. https://doi.org/10.1080/10447318.2024.2431761

Orlikowski, W. J., & Yates, J. (1994). Genre repertoire: The structuring of communicative practices in organizations. *Administrative Science Quarterly*, *39*(4), 541–574. https://doi.org/10.2307/2393771

Panickssery, A., Bowman, S., & Feng, S. (2024). LLM evaluators recognize and favor their own generations. In *Advances in Neural Information Processing Systems 37* (pp. 68772–68802). https://doi.org/10.52202/079017-2197

Park, S., & Arthur, C. (2026). *From prompting to describing: A cross-cultural study of language for AI-generated music* (arXiv:2608.06634) [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2608.06634

Putnam, H. (1975). The meaning of "meaning". In K. Gunderson (Ed.), *Language, mind, and knowledge* (pp. 131–193). University of Minnesota Press.

Razeghi, Y., Logan IV, R. L., Gardner, M., & Singh, S. (2022). Impact of pretraining term frequencies on few-shot numerical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2022* (pp. 840–854). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.findings-emnlp.59

Reynolds, L., & McDonell, K. (2021). Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems* (pp. 1–7). Association for Computing Machinery. https://doi.org/10.1145/3411763.3451760

Schreiter, D. (2025). *Prompt engineering: How prompt vocabulary affects domain knowledge* (arXiv:2505.17037) [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2505.17037

Soldaini, L., Kinney, R., Bhagia, A., Schwenk, D., Atkinson, D., Authur, R., Bogin, B., Chandu, K., Dumas, J., Elazar, Y., Hofmann, V., Jha, A., Kumar, S., Lucy, L., Lyu, X., Lambert, N., Magnusson, I., Morrison, J., Muennighoff, N., … Lo, K. (2024). Dolma: An open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 15725–15788). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.acl-long.840

Sorensen, T., Robinson, J., Rytting, C., Shaw, A., Rogers, K., Delorey, A., Khalil, M., Fulda, N., & Wingate, D. (2022). An information-theoretic approach to prompt engineering without ground truth labels. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 819–862). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.60

Swales, J. M. (1990). *Genre analysis: English in academic and research settings*. Cambridge University Press.

Tao, S., Liang, I., Peng, C., Wang, Z., Palani, S., & Dow, S. P. (2025). DesignWeaver: Dimensional scaffolding for text-to-image product design. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems* (pp. 1–26). Association for Computing Machinery. https://doi.org/10.1145/3706598.3714211

Taylor, R. S. (1968). Question-negotiation and information seeking in libraries. *College & Research Libraries*, *29*(3), 178–194. https://doi.org/10.5860/crl_29_03_178

Team OLMo, Walsh, P., Soldaini, L., Groeneveld, D., Lo, K., Arora, S., Bhagia, A., Gu, Y., Huang, S., Jordan, M., Lambert, N., Schwenk, D., Tafjord, O., Anderson, T., Atkinson, D., Brahman, F., Clark, C., Dasigi, P., Dziri, N., … Hajishirzi, H. (2025). 2 OLMo 2 furious. In *Conference on Language Modeling (COLM 2025)*. https://doi.org/10.48550/arXiv.2501.00656

Udandarao, V., Prabhu, A., Ghosh, A., Sharma, Y., Torr, P. H. S., Bibi, A., Albanie, S., & Bethge, M. (2024). No "zero-shot" without exponential data: Pretraining concept frequency determines multimodal model performance. In *Advances in Neural Information Processing Systems 37* (pp. 61735–61792). https://doi.org/10.52202/079017-1973

Vrandečić, D., & Krötzsch, M. (2014). Wikidata: A free collaborative knowledgebase. *Communications of the ACM*, *57*(10), 78–85. https://doi.org/10.1145/2629489

Wahle, J. P., Ruas, T., Xu, Y., & Gipp, B. (2024). Paraphrase types elicit prompt engineering capabilities. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing* (pp. 11004–11033). Association for Computational Linguistics. https://doi.org/10.18653/v1/2024.emnlp-main.617

Wang, B., Zhou, K., Guo, L., Chen, F., & Zhang, C. (2026). *FinProBench: Evaluating financial AI agents with role-grounded rubrics derived from professional deliverables* (arXiv:2608.04077) [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2608.04077

Wang, Z., Jung, J., Lu, X., Diao, S., Evans, E., Zeng, J., Molchanov, P., Choi, Y., Kautz, J., & Dong, Y. (2026). ProfBench: Multi-domain rubrics requiring professional knowledge to answer and judge. In *International Conference on Learning Representations (ICLR 2026)*. https://proceedings.iclr.cc/paper_files/paper/2026/hash/09c4960d66e6182e57062c5d71ce1d3d-Abstract-Conference.html

Webson, A., & Pavlick, E. (2022). Do prompt-based models really understand the meaning of their prompts? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 2300–2344). Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.naacl-main.167

White, R. W., Dumais, S. T., & Teevan, J. (2009). Characterizing the influence of domain expertise on web search behavior. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining* (pp. 132–141). Association for Computing Machinery. https://doi.org/10.1145/1498759.1498819

Wikidata. (2026). *Wikidata* [Data set; SPARQL query results retrieved November 2, 2026 (UTC)]. Wikimedia Foundation. https://query.wikidata.org/

Xie, Q., Liang, Z., Wu, J., Chen, Y., Wang, W., Ma, W., Ming, Z., Yang, H., & Wu, K. (2026). Beyond prompt engineering: A systematic analysis of prompt lexical sensitivity and its impacts on quality. In *Findings of the Association for Computational Linguistics: ACL 2026* (pp. 41998–42012). Association for Computational Linguistics. https://doi.org/10.18653/v1/2026.findings-acl.2084

Yang, S., Kim, J., Jang, J., Ye, S., Lee, H., & Seo, M. (2024). Improving probability-based prompt selection through unified evaluation and analysis. *Transactions of the Association for Computational Linguistics*, *12*, 664–680. https://doi.org/10.1162/tacl_a_00666

Yun, H. S., Kapoor, G., Mackert, M., Kouzy, R., Xu, W., Li, J. J., & Wallace, B. C. (2026). *This treatment works, right? Evaluating LLM sensitivity to patient question framing in medical QA* (arXiv:2604.05051) [Preprint]. arXiv. https://doi.org/10.48550/arXiv.2604.05051

Zamfirescu-Pereira, J. D., Wong, R. Y., Hartmann, B., & Yang, Q. (2023). Why Johnny can't prompt: How non-AI experts try (and fail) to design LLM prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (pp. 1–21). Association for Computing Machinery. https://doi.org/10.1145/3544548.3581388

Zhao, C., & Yang, H. (2026). Evaluating LLM robustness under domain-specific prompt perturbations in public health applications. In *Proceedings of the 2026 2nd International Conference on Big Data and Intelligent Medicine* (pp. 304–308). Association for Computing Machinery. https://doi.org/10.1145/3816325.3843642

Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., & Ba, J. (2023). Large language models are human-level prompt engineers. In *International Conference on Learning Representations (ICLR 2023)*. https://doi.org/10.48550/arXiv.2211.01910

Appendix A. Materials and prompt samples

A.1 Prompt versions

All versions of a term share one task sentence. The slot is filled after removing any leading article from the lay description, and the preceding article is set to "a" or "an" from the first letter of the filled phrase. No system prompt is added beyond the access environment's own. Rich task sentences give a setting and purpose without listing the deliverable's sections, although some name the medium (script, checklist, slides).

- Curated set, RACI matrix (rich). Name: "Create a RACI matrix for launching a new company website." Lay A: "Create a who-does-what chart for launching a new company website." Lay B: "Create a chart showing who does what for launching a new company website."

- Curated set, line check. Rich: “Write the checklist for a {X} before dinner service in a 60-seat bistro.” Thin: “Write a {X} for a 60-seat bistro.”
- Sampled set, cost-benefit analysis (rich). Task: “Write a {X} for a mid-sized logistics company deciding whether to replace its paper-based delivery tracking with handheld scanners across its twelve regional depots.” Lay A: “report weighing the pros and cons.” Lay B: “write-up weighing whether the switch is worth the money.”

Claude agent instances were told to answer each request directly as in a normal chat. The OpenAI client received each request prefixed by “Answer the following user request directly in your reply, as you would in a normal chat. Do not run commands or create files.”

A.2 Lay descriptions

Both lay versions had to (a) contain neither the name nor any of its content words, (b) say what kind of thing the deliverable is without listing its contents or procedure, and (c) fit the slot grammatically. Lay A was requested as a short label. Lay B was written by an instance told neither the purpose of the study nor the checklist. One Lay A was itself a practitioner name. Five whys analysis was described as “deep dive,” which is also the name of a curated term.

A.3 Checklist example

The curated checklists drew on 26 government, 24 university, 14 professional-body, 40 practitioner-guide, 4 textbook, 3 standards, and 8 other sources. The first five of the RACI matrix items, each linked to its supporting sources (two university guides and one practitioner guide), are as follows.

1. It is a table with tasks or deliverables as rows and roles or people as columns (or the transpose).
2. Each cell uses only the letters R, A, C, I (or is left blank).
3. It defines R = Responsible (does the work), A = Accountable (owns or approves the outcome), C = Consulted (gives input beforehand), I = Informed (kept updated).
4. Every task has exactly one A.
5. Every task has at least one R.

A.4 Judge instruction

“You are checking a document against a checklist. Judge each item strictly from the document text: answer 1 only if the document clearly does what the item says (explicitly present, not merely implied or hinted); otherwise 0. Do not reward length or style.” The document and the numbered checklist follow, and the judge answers with one JSON key per item. In the cross-vendor rescoring, each answer of 1 must include a verbatim quotation from the document, and a script checked each quotation after normalizing whitespace and case.

A.5 Two-step naming and PMI prefix

Step one: “Here is a request someone wrote without knowing the professional term: «{lay prompt}». What is the standard term practitioners use for the deliverable being requested? Reply with the term only.” Step two

inserts the returned name into the task sentence and generates in a new context. Outputs are scored against the original checklists by version-blind judges, and the naming judge records reasons.

PMI prefix: “Notes on what a good {name} looks like:” followed by a line break, a hyphen, and a space, then the checklist item text, with the name replaced by Lay A or Lay B in the comparison conditions.

Appendix B. Sampling protocol and deviations for the sampled set

B.1 Candidates

One route took task statements and detailed work activities from the O*NET 31.0 database. It extracted the noun phrase after production verbs such as write, draft, and prepare, and after develop, create, or design when the object was a communicative or cognitive noun. The phrase was truncated before the first preposition, comma, or conjunction and normalized. The other route took Wikidata items that are direct instances of classes such as document, technique, method, diagram, chart, model, report, and plan and have an English Wikipedia article, from SPARQL queries on 2 November 2026 (UTC). Merging gave 4,665 candidates: 1,922 from O*NET only, 2,661 from Wikidata only, and 82 from both.

B.2 Eligibility and screening

- E1: the deliverable is text, or a diagram that can be written in text (a Markdown table, ASCII layout, or diagram markup).
- E2: at least two public normative sources from different publishers define the deliverable’s conventions.
- E3: the English form has an infini-gram count of at least 100 in Dolma, which removes extraction debris rather than filtering by frequency (J. Liu et al., 2024; Soldaini et al., 2024).
- E4: a lay description sharing no content word with the name can be written, and the name itself is not everyday lay usage.
- Exclusions: proper names, deliverables that are not primarily textual, statutory forms, generic heads on their own (report, plan, form), curated-set terms, and terms with safety-refusal risk.
- Type rule: method name if the name points to a fixed internal structure applicable to many documents (components of an acronym, a person’s name, or a structural noun such as matrix, diagram, canvas, or analysis), genre otherwise.

Generic heads (15) and curated-set terms (7) were removed mechanically. The mechanical match required identical strings, so SBAR remained in the sampled set although the curated set included SBAR report. Two independent blind judges, an Opus and a Sonnet instance, screened the remaining 4,643 candidates in a seeded random order for eligibility, type, and domain. They agreed on eligibility for 0.930 of candidates ($\kappa = 0.733$). On the 557 both judged eligible, type κ was 0.968 and domain κ 0.898. A third instance adjudicated the 373 disagreements under the rules in B.3. After 4,049 screening exclusions and 14 by the count criterion, 580 terms were eligible, 92 method names and 488 genres.

B.3 Adjudication rules

The third judge applied rules written before adjudication. Data graphics are not text deliverables. Programs, strategies, and plans count only if usually delivered as a written document with public writing guidance. Generic words and extraction fragments fail E1. Single-author or single-vendor frameworks fail E2. A name that is itself everyday lay usage fails E4.

B.4 Quotas and replacement

Eligible terms were drawn in the fixed order into cells of type by six domains: management and business; computer, engineering, and science; health care; education, legal, and social services; arts, media, and communication; and service, construction, production, and transportation. Quotas were nine or eight method names (50 in all) and 17 or 16 genres (100 in all) per domain, assigned to domains in a seeded random order. Shortfalls of method names in some domains were filled from others by rule, so method names concentrate in engineering and science (19) and management and business (16). Genres were split at the median count of all eligible genres (103,018.5) into common and specialist. A term was replaced by the next eligible term in its cell when no checklist could be built from two publishers, fewer than six items were supported, or an eligibility violation emerged. Builders flagged 40 terms in the first round. The third judge upheld 26 and overturned 14. For six of the overturned terms the name is not the most canonical one but is not a lay word either, and these terms were kept and examined in a sensitivity analysis. Three rounds replaced 26, 6, and 4 terms, with the second and third rounds replacing replacements that were themselves flagged. The final set had 50 method names, 49 specialist genres, and 51 common genres.

B.5 Materials and checklist construction

A model instance that saw only the term, its type, and its domain wrote each task sentence. Lay A was written by 27B for 123 terms and by Claude Haiku 5.5 with the same prompt for 27, and Lay B by a Claude instance without access to checklists. Builders followed a fixed protocol. They found at least two public normative sources by different publishers, with at most one vendor source and at most one Wikipedia source. From each source they extracted the conventions that distinguish the deliverable from ordinary text, with a short supporting quotation, and merged them into binary items linked to their sources. They recorded source type and access date and froze the checklist before judging. Builders saw only the protocol and the term's inputs, never model outputs. The 422 sources were 157 government, 82 university, 49 association, 43 practitioner-guide, 31 Wikipedia, 28 textbook, 16 vendor, and 16 standards sources. The builder agents' task protocol was not written in English, and for 22 terms a builder wrote the items in that language. Judges applied them as written to the English outputs, and English translations are in the supplementary material.

Structure coding of the sampled set gave 534 S, 1,135 C, and 52 T items. A second coder recoded 337 items from 30 terms coded before the last replacements ($\kappa = 0.81$). Of these, 25 terms with 287 items are in the final set, with $\kappa = 0.80$.

B.6 Pre-specified criteria and deviations

Four criteria were fixed before any output was generated. Q1 required each model's Δ to have a 95% lower bound above zero. Q2 required the pooled difference between method names and common genres to have a lower bound above zero. Q3 required the same of the pooled difference between structure and content items. Q4 required $\rho(\text{PMI}, \Delta) \geq 0.3$ with a lower bound above zero. The plan named 27B as one of three tested

models. 27B was dropped when its compute host became unavailable, and the 33 outputs it had generated by then were not analyzed. Before analysis, Opus replaced it in the pooled criteria, which pool Sonnet, Opus, and GPT-6.1-Sol. Haiku was added as a fourth model, and Q4, which needs open-model log-probabilities, was not run. Adding Haiku to the Q2 pool gave a method-minus-common difference of +0.155.

For two-step naming, the per-prompt ratio was computed where name coverage exceeded lay coverage (gap > 0). With at most 12 items per checklist, the smallest nonzero coverage difference is $1/12 \approx 0.08$. The curated-set threshold of 0.05 therefore selects the same 163 prompts and gives the same ratios. Naming was first judged by string matching alone. To match the curated set, the model judgment of synonymy was added after the coverage judgments were available but before any synonymy judgment was made. String matching alone is reported as a lower bound. Sonnet named 138 lay prompts correctly by automatic matching, and 51 more were judged synonymous. A step-two generator revised numbers in two of its outputs by script before submitting them, and these outputs were not excluded.

B.7 Sensitivity analyses

Dropping the six terms with non-canonical names changed each model's gain by at most 0.004. Restricting to the 123 terms with 27B-written Lay A left gains between 0.108 and 0.199. Dropping the 22 terms with checklist items not written in English left gains between 0.113 and 0.199. Dropping SBAR, which overlaps the curated set's SBAR report, gave 0.125 for Sonnet, 0.113 for Opus, 0.137 for Haiku, and 0.196 for GPT-6.1-Sol (n = 149). The most negative four-model gains were slight, the lowest being −0.10 for graphic charter.

B.8 Flow

Table B.1. Sampling flow for the sampled set, from candidates to the final term set.

Step	Count
Candidates after merging O*NET and Wikidata routes	4,665
Removed mechanically (generic heads 15, curated-set terms 7)	22
Screened by two blind judges	4,643
Ineligible at screening	4,049
Removed by the count criterion (E3)	14
Eligible (92 method names, 488 genres)	580
Drawn into type × domain cells	150
Flagged in round 1 / upheld / overturned	40 / 26 / 14
Replaced in rounds 1, 2, 3 (rounds 2 and 3 replaced flagged replacements)	26, 6, 4
Final set (method names, specialist genres, common genres)	150 (50, 49, 51)

Appendix C. Additional settings and analyses

C.1 Generation settings

27B was run with up to 3,000 output tokens, and its 480 outputs included eight that reached the length limit, three name-version and five lay outputs from two terms. Truncation could bias Δ in either direction, but the

affected cells are too few to matter. Claude instances received requests with opaque identifiers in random order. OpenAI calls used a configuration directory containing only credentials, and none failed. Claude models were selected by the agent environment's model aliases (haiku, sonnet, opus), and OpenAI models by the identifiers gpt-6.1-sol and gpt-6-luna through version 0.163 of the OpenAI client. All outputs were generated on 2 and 3 November 2026. In the curated set a Haiku instance revised one output after its judging batch had been built, so the earlier draft was judged, and the output was not excluded. Counts of terms with positive or negative gain treat differences within 10^{-12} of zero as ties.

C.2 Structure coding of the curated set

The 442 curated items were coded as 58 S, 352 C, and 32 T. A second coder recoded 90 items from 8 terms ($\kappa = 0.69$) and agreed on all 16 recoded lay judgments. The structure advantage held in five of six models, with Haiku's CI spanning zero.

C.3 PMI sensitivity

Against Δ_{all}, PMI correlated with per-term gains at $\rho = 0.34$ for 27B, 0.37 for Haiku, 0.36 for GPT-6.1-Sol, and 0.41 for GPT-6-Luna, each with a lower bound above zero. It gave 0.23 for Sonnet and −0.07 for Opus. Against the rich-scenario gain Δ_{rich}, lower bounds were above zero only for Sonnet ($\rho = 0.39$) and GPT-6-Luna (0.41). 27B fell to 0.16 [−0.19, 0.47], and the six-model mean gave 0.40 [0.07, 0.66]. Nine checklist items from five terms contain the name, which the lay prefixes lack by design. Dropping them left PMI nearly unchanged ($\rho = 0.993$ with the original). Against the six-model mean gain it gave $\rho = 0.35$ [0.02, 0.62]. The lower bound stayed above zero for 27B, GPT-6.1-Sol, and GPT-6-Luna but not for Haiku (−0.02).

C.4 Exploratory comparisons across models

Within the Claude family, point estimates of Δ_{all} fell from Haiku to Sonnet to Opus (0.156, 0.142, 0.112). This ordering was predicted in an amendment to the analysis plan written after the Sonnet and Opus results were known. The paired Haiku–Opus difference spanned zero (Δ_{all} +0.044 [−0.008, 0.101], rich +0.034 [−0.027, 0.098]).

Name coverage of a smaller model was also compared with lay coverage of Opus. On curated terms, the paired differences were +0.009 [−0.048, 0.071] for Haiku and +0.018 [−0.048, 0.084] for 27B. Intervals that span zero do not establish equivalence, and the 27B comparison also crosses judges. GPT-6-Luna reached 0.62 with the name, below Opus with lay descriptions. On sampled terms, against Opus with lay descriptions, name coverage differed by −0.063 [−0.099, −0.027] for Haiku, +0.059 [0.029, 0.089] for Sonnet, and −0.137 [−0.181, −0.094] for GPT-6.1-Sol (n = 150).

C.5 Public lexical resources and corpus frequency

An Opus instance without access to results computed for each curated name its lexicalization (presence in WordNet 3.1, Wiktionary, FrameNet 1.7, and English Wikipedia, 0–4), number of senses, Wiktionary domain labels, and Wikipedia page views. It also looked up word prevalence norms (Baker et al., 1998; Brysbaert et al., 2019; International Computer Science Institute, n.d.; G. A. Miller, 1995). Corpus frequency summed three capitalizations in infini-gram indexes of Dolma and of the OLMo 2 training mixture (J. Liu et al., 2024;

Soldaini et al., 2024; Team OLMo et al., 2025). Predicted directions were registered before computation. Over the 40 terms, lexicalization correlated with the mean gain at $\rho = 0.05$ and number of senses at -0.30, both CIs spanning zero. Prevalence norms list single lemmas and matched only one name as a whole.

Frequency was related to coverage and gain in an earlier 27B run on the 20 common workplace genres. That run used the same task sentences, lay descriptions, and checklists, the rich scenario only, five outputs per version at temperature 0.7 with up to 2,000 output tokens and reasoning disabled, and 27B judging its own outputs. Corpus frequency correlated with 27B's name coverage at $\rho = 0.11$ and with its gain at -0.08. Of its 300 name-version and lay outputs, 60 reached the length limit. Excluding them left both correlations near zero ($\rho = -0.14$ and -0.04).