**Introduction**

Bayesian optimization (BO) has become a workhorse for data-efficient experimental design in chemistry, enabling autonomous platforms to discover high-yielding conditions with tens—not hundreds—of trials. Yet BO’s early iterations are notoriously brittle: with few data, a surrogate model (typically a Gaussian process, GP) is driven as much by its prior as by observations. Today, most chemistry BO systems default to weak, implicit priors (zero mean; generic kernels) and attempt to compensate with hand-picked seeds (good initial points) or by reusing old data (transfer/meta-learning). Those strategies help when suitable seeds or source tasks exist, but they do not address the fundamental problem: how to encode what we already know about a new reaction family—mechanistic trends, safe operating envelopes, known trade-offs—before we run any experiments.

We propose a simple but powerful alternative: use a large language model (LLM) to translate concise domain knowledge (reaction schema, bounds/roles of variables, safety notes, and a handful of context bullets) into a structured, auditable prior for BO. Concretely, the LLM emits a JSON readout that specifies (i) expected 1-D response shapes for each continuous knob (e.g., temperature has a peak; time is saturating); (ii) a short list of plausible interactions (e.g., temperature×time synergy); and (iii) optional similarity structure over categorical choices (e.g., bases, ligands, solvents). We map this readout into (a) a prior mean m0(x)m\_0(x)m0​(x) that nudges the surrogate toward chemically sensible trends in the cold-start regime and (b) a structure-aware feature map ϕ(x)\phi(x)ϕ(x) that makes residual learning easier. The result is language-shaped priors: a drop-in module for standard ask–tell BO loops that improves cold-start performance without requiring archival numeric data.

## Modeling framework

Let denote an experiment (continuous knobs such as temperature/time/concentration; categoricals such as base/ligand/solvent). We model the objective (e.g., yield) as

,

### LLM readout ⇒\Rightarrow⇒ prior mean m0(x)m\_0(x)m0​(x)

The LLM readout provides, for each continuous variable xix\_ixi​ normalized to zi ⁣∈ ⁣[0,1]z\_i\!\in\![0,1]zi​∈[0,1], an **effect type** ti∈{increase,decrease,saturating,peak,valley,flat}t\_i\in\{\text{increase},\text{decrease},\text{saturating},\text{peak},\text{valley},\text{flat}\}ti​∈{increase,decrease,saturating,peak,valley,flat}, a scale si∈[0,1]s\_i\in[0,1]si​∈[0,1], a confidence ci∈[0,1]c\_i\in[0,1]ci​∈[0,1], and optionally a **range hint** [ai,bi]⊂[0,1][a\_i,b\_i]\subset[0,1][ai​,bi​]⊂[0,1]. We map these to smooth basis functions gi(zi;ti)g\_i(z\_i;t\_i)gi​(zi​;ti​) (sigmoid/saturation/Gaussian bump) and combine them with interaction terms from the readout:

m0(x)=∑i∈contαi gi ⁣(zi;ti,[ai,bi])  +  ∑(i,j)∈Iβij h ⁣(zi,zj;synergy/antagonism).m\_0(x) = \sum\_{i\in\mathrm{cont}} \alpha\_i \, g\_i\!\big(z\_i; t\_i, [a\_i,b\_i]\big) \;+\; \sum\_{(i,j)\in \mathcal{I}} \beta\_{ij}\, h\!\big(z\_i,z\_j; \text{synergy/antagonism}\big).m0​(x)=i∈cont∑​αi​gi​(zi​;ti​,[ai​,bi​])+(i,j)∈I∑​βij​h(zi​,zj​;synergy/antagonism).

The **prior strength** αi\alpha\_iαi​ and βij\beta\_{ij}βij​ are calibrated from confidence (αi∝si⋅(12+12ci)\alpha\_i\propto s\_i\cdot(\tfrac12+\tfrac12c\_i)αi​∝si​⋅(21​+21​ci​)); low-confidence beliefs contribute weakly and are easy to override.

### LLM readout ⇒\Rightarrow⇒ structure-aware features ϕ(x)\phi(x)ϕ(x)

We augment standard numeric encodings with:

* **Polynomial features** of normalized continuous variables (up to degree 2–3), aligned with the readout shapes.
* **Categorical similarity**: the readout can provide a pairwise similarity SSS over category levels; we embed with one-hot + similarity-aware mixing or via a kernel kcat(xcat,xcat′)=S[xcat,xcat′]k\_{\mathrm{cat}}(x\_{\mathrm{cat}},x'\_{\mathrm{cat}})=S[x\_{\mathrm{cat}},x'\_{\mathrm{cat}}]kcat​(xcat​,xcat′​)=S[xcat​,xcat′​].
* Optional **interaction features** for the specific pairs listed in the readout.

This yields a design matrix that is easier for the residual GP to fit with few points.

### Residualized GP and pseudodata view

Fitting proceeds on the **residuals** r=y−m0(X)r = y - m\_0(X)r=y−m0​(X) with a standard GP on ϕ(X)\phi(X)ϕ(X). Equivalently, one may view the prior as adding **virtual observations** {(xp, yp=m0(xp))}\{(x\_p,\,y\_p=m\_0(x\_p))\}{(xp​,yp​=m0​(xp​))} with precision λ(c)\lambda(c)λ(c) that shrinks toward the readout where confidence is high. The posterior mean at a new point is

μ(x∣D)  =  m0(x)  +  kx⊤K−1 ⁣(y−m0(X)),\mu(x\mid\mathcal{D}) \;=\; m\_0(x) \;+\; k\_x^\top K^{-1}\!\big(y - m\_0(X)\big),μ(x∣D)=m0​(x)+kx⊤​K−1(y−m0​(X)),

so the readout influences predictions most when ∣D∣|\mathcal{D}|∣D∣ is small and vanishes asymptotically under standard regularity conditions.

## Hypotheses

* **H1 (Cold-start gain):** Language-shaped priors **increase early AUC of best-so-far** and reduce burn-in iterations relative to weak/flat priors, especially when initial seeds are random or intentionally poor.
* **H2 (Robustness to bad seeds):** Good priors **recover faster** from misleading initial data than flat priors by steering exploration toward chemically plausible regions.
* **H3 (Graceful failure):** When the readout is partially wrong, **calibrated confidence** limits bias; the residual GP rapidly overrides incorrect components.
* **H4 (Asymptotic neutrality):** As data accumulate, performance converges to that of an uninformative prior (the readout does not harm consistency).

## Relation to prior work

* **Warm-starts / seed design.** Choosing high-quality initial points improves BO but **changes the data, not the model**. Our approach is orthogonal: we keep seeds fixed and change the **beliefs** (the prior), or vice versa, to disentangle effects.
* **LLM-in-the-loop BO (ICL, sampling, scoring).** Recent systems use LLMs to propose candidates or act as text-based surrogates. We instead use the LLM once, up front, to create an **explicit, auditable prior** for a standard surrogate (GP), preserving uncertainty calculus and acquisition theory.
* **Knowledge-guided priors / preference BO.** Prior works inject hand-crafted beliefs (e.g., location/value of optima). We generalize this via an LLM that produces **full shape+interaction priors** under a strict schema, with confidence-to-strength calibration.
* **Transfer/meta-BO.** Pretraining priors from many numeric tasks improves cold-start when such tasks exist. Our method provides **zero-shot transfer from text** and can be combined with meta-BO by learning a mapping from “task card →\to→ readout parameters”.
* **Multi-task BO.** Joint modeling across tasks exploits inter-task kernels. Our current focus is single-task; however, the same LLM can emit **task descriptors** to build an inter-task kernel, enabling a multi-task extension.

## Contributions

1. **Language-shaped priors for BO.** A principled, schema-constrained method to turn domain knowledge (roles, bounds, hazards, mechanistic hints) into a **prior mean** m0(x)m\_0(x)m0​(x) and **structure-aware features** ϕ(x)\phi(x)ϕ(x) for residual GP surrogates.
2. **Confidence-calibrated strength.** A mapping from readout confidence to prior strength (pseudodata/regularization) that gives **graceful failure** when the prior is imperfect and **asymptotic neutrality** as data grow.
3. **Interpretable interactions and categorical similarity.** The readout encodes a **small, auditable set of interactions** and a **chemistry-consistent similarity** over categorical choices, improving information sharing in sparse regimes.
4. **Closed-loop evaluation protocol.** A benchmark that **decouples priors from seeds**, comparing NoPrior / Heuristic / LLM-Good / LLM-Bad / RandomPrior under **good/bad/random initialization**, reporting early-phase AUC and Best@k.
5. **Chemistry case studies.** We demonstrate gains on autonomous optimization of **aldol condensation** and **direct arylation** oracles, showing improved cold-start and robustness to adversarial seeds without sacrificing long-run optimality.

## Advantages over existing approaches

* **Auditable and editable.** The prior is explicit JSON, not a black box; chemists can inspect and adjust shapes, ranges, and interactions before running.
* **Drop-in to standard BO.** No change to acquisition functions or GP tooling; works with EI/UCB/TS and most model stacks.
* **Zero-shot applicability.** Works **without archival numeric data**—useful for unpublished or novel chemistry where meta-learning has nothing to train on.
* **Safety by design.** Prior can encode **hard envelopes** (e.g., polymerization/pressure limits) and down-weight exploration outside safe regions.
* **Robust to misspecification.** Confidence-scaled contributions and residual modeling mitigate harm when parts of the prior are wrong.

## Limitations and next steps

* **Confidence calibration.** We currently map LLM confidence heuristically to prior strength; future work will **learn this mapping** from retrospective campaigns (empirical Bayes) or via **conformal calibration**.
* **Meta-priors from text+data.** Train a lightweight model to map a task card to prior parameters using historical campaigns—merging our zero-shot approach with **meta-BO**.
* **Multi-task extension.** Use LLM-derived task descriptors to build an **inter-task kernel**, enabling **multi-task BO** when optimizing related substrates/solvents in parallel.
* **Richer structures.** Extend the readout to **monotonicity constraints**, **ridge/valley** in ratio spaces, and **physics-informed kernels**.
* **Human-in-the-loop editing.** Build UI to compare LLM-Good vs LLM-Bad vs Heuristic readouts, with **pre-run diagnostics** (posterior under synthetic draws) to catch pathologies early.
* **Theoretical guarantees.** Formalize finite-sample regret improvements under bounded prior misspecification and confidence scaling.

## Summary

Language-shaped priors transform BO for chemistry from “start blank, hope seeds are good” to **“start informed, learn fast, and stay auditable.”** By converting compact mechanistic context into a calibrated prior mean and structure, we accelerate early learning, recover from poor seeds, and retain the statistical discipline of GP-based BO. This makes autonomous experimentation more **data-efficient, safer, and easier to trust**—even when no prior numeric dataset exists.

## Q & A: Q1 = this prior that we make using LLM or heuristic, are different with initialization points that we feed them into the BO for optimization? what is prior? TL;DR

* **Prior** = your model’s **belief about the response surface before seeing data**.  
  In our setup it’s the **LLM/heuristic JSON** that shapes the surrogate’s prior mean m0(x)m\_0(x)m0​(x), interactions, and the feature map/similarities.
* **Initialization points (seeds)** = the **first real experiments** you feed the optimizer.  
  They’re just observed (x,y)(x,y)(x,y) pairs that condition the model (the “training data” at iteration 0).

They’re independent but interact in the optimizer.

## What “prior” means here

In Bayesian Optimization, the surrogate (e.g., GP) starts with a prior over functions  
f∼GP(m0(x),k0(x,x′))f \sim \mathcal{GP}(m\_0(x), k\_0(x,x'))f∼GP(m0​(x),k0​(x,x′)).

Our **language-shaped prior** provides:

* **m0(x)m\_0(x)m0​(x)**: a bias/mean function (e.g., “Temp has a peak”, “Time saturates”), created from the readout JSON.
* **ϕ(x)\phi(x)ϕ(x)**: extra features from the readout (polynomials of normalized knobs, one-hots, optional embeddings/similarities) so the residual model can learn faster.
* Optionally **category similarity**: so similar reagents share information early.

In code, that’s your readout → PriorBuilder → used by the **ResidualizedRegressor**:

* Predicts y^(x)=m0(x)+residual\_model(ϕ(x))\hat{y}(x) = m\_0(x) + \text{residual\\_model}(\phi(x))y^​(x)=m0​(x)+residual\_model(ϕ(x)).
* Before you have much data, m0(x)m\_0(x)m0​(x) strongly steers predictions and uncertainties → **better cold-start**.

**Important:** A prior does **not** run an experiment. It doesn’t add data.  
It only shapes what the surrogate believes before (and between) measurements.

## What “initialization points” are

These are the **actual designs you evaluate first** (your DoE / seeds):

* You choose ninitn\_{\text{init}}ninit​ points (e.g., good/top, bad/bottom, random, Latin hypercube).
* The oracle/experiment returns real **yields**.
* Those (Xinit,yinit)(X\_{\text{init}}, y\_{\text{init}})(Xinit​,yinit​) become the model’s first observations.  
  The model is **conditioned** on them to form a posterior p(f∣D0)p(f \mid \mathcal{D}\_0)p(f∣D0​).

Seeds can encode domain knowledge too (e.g., “start near known good”), but that’s **design**, not prior.

## How they interact in BO

At each “ask” step we pick the next xxx by an acquisition (EI/UCB), which uses:

* **Posterior mean/variance** = driven by **both** the **prior** and the **data so far**.
* With **few data**, the prior has more influence; as data grow, the model relies more on observations.

**Examples**

* **LLM-Good prior + bad seeds**: the prior can “pull” the model away from poor initial measurements and still explore promising regions → faster recovery.
* **NoPrior + good seeds**: strong start just from data; may still converge well.
* **LLM-Bad prior + bad seeds**: double whammy—can delay finding good regions (this is a stress test).
* **Heuristic prior + random seeds**: usually a safe baseline; better than NoPrior when the space is large.

## How to change each in our code

* **Prior**: edit/choose the readout JSON (LLM/heuristic/random). In the LLM case we generate it from bullets and cache it (llm\_good\_readout.json, llm\_bad\_readout.json). This affects m0m\_0m0​, interactions, and features everywhere in the space.
* **Initialization**: choose init\_mode (good, bad, random) and n\_init. This only selects the **first evaluated designs**; it doesn’t alter the prior itself.

They’re **orthogonal controls**: you can keep the same prior and compare different seeds, or keep the same seeds and compare priors.

## Practical guidance

* Use a **reasonable prior** (LLM-Good or Heuristic) when the space is big or initial data are scarce.
* Keep **seeds diverse** even if “good” (we did greedy farthest in normalized space) to avoid early overfitting.
* Always report **AUC(best-so-far)** and **Best@k** across runs, stratified by **init mode** and **prior type**—that shows both cold-start and robustness.

If you want, I can dump a tiny schematic figure or a toy numeric example that shows how the same 6 seeds behave under different priors (posterior means/uncertainties and EI choices).

Q2-how LLM can generate better readouts before doing certain experiment?  
Short answer: by turning everything you already know **before** running new trials—chemistry rules, constraints, analogs, and reagent properties—into a **structured prior** on the response surface. A good LLM prior isn’t magic text; it’s a disciplined way to encode pre-experimental signal into your surrogate’s mean function and structure.

Here’s how to make LLMs generate **better readouts** up front:

## 1) Feed the LLM the right scaffolding

Give a compact, unambiguous “task card”:

* **Schema**: each variable’s name, role, physical bounds, units.
* **Context bullets**: the specific reaction family, hazards/constraints (e.g., polymerization risk at high T), known mechanistic levers (nucleophile strength, electrophilicity), practicalities (solubility/phase, pressure).
* **Allowed output schema**: the exact JSON keys + allowed values (effect, scale, confidence, range\_hint, interactions).

This eliminates prose/hallucinations and forces a usable readout.

## 2) Ground with analogs (few-shot + retrieval)

Even if your exact system is unpublished, many **closely related** systems are. Build a tiny library:

* (input): a reaction card (family, knobs, bounds)
* (output): a vetted readout JSON + a short “why” note (not used at runtime)  
  Then do **few-shot prompting**: include 1–3 most similar exemplars (by reaction class or reagent taxonomy) so the model “pattern-matches” shapes:
* Temperature: often **peak**; Time: **increase-saturating**; Equivalents: **peak**; Ratios: **ridge**.

## 3) Add descriptor summaries, not just names

Give the model numbers it can reason about:

* For each categorical option (ligand/base/solvent), pass **precomputed descriptors** (basicity/pKa window, polarity indices, donor/acceptor, sterics) as **brief statistics** (mean ± sd, min/max across the candidate set).
* For continuous knobs, add **engineering constraints** (e.g., “>70 °C risks X”) as bullets.  
  This lets the LLM set **category similarity** and **range\_hint** more intelligently.

## 4) Ask for interactions explicitly

LLMs are good at proposing **mechanistically plausible interactions** (e.g., T×time synergy; equivalents×solvent antagonism). Restrict them to continuous–continuous pairs and limit to a small number (top-2) with confidences. This gives your model structure the residual learner can exploit early.

## 5) Calibrate strength and uncertainty

Tell the LLM to output **confidence** and keep it honest:

* Chemistry-certain knobs → higher confidence/scale (e.g., T shape = peak, medium–high confidence).
* Ambiguous knobs → lower scale/confidence.  
  Then **map confidence to prior strength** (pseudopoints) on your side, so weak beliefs don’t overrule data.

## 6) Use ensemble & self-consistency

Sample the LLM **K times** (different seeds or slight perturbations), then:

* Keep only JSONs that pass **schema/constraint checks** (bounds in [0,1], allowed strings, interaction pairs valid).
* **Aggregate** effects (median shape; mean scale/confidence).
* Optionally run a second “critic” prompt that spots contradictions (e.g., “you set temp=flat but also warned about high-T side reactions”).

## 7) Bake in domain constraints to prevent nonsense

* Hard floors/ceilings: impossible regions get low range\_hint or low confidence.
* Monotonicity where physics demands (e.g., catalyst loading never strictly decreases yield at very low loadings).
* Safety envelopes (pressure, solubility) → reduce confidence outside safe windows.

## 8) Tie category similarity to chemistry families

Ask the LLM to group options (e.g., bulky dialkylbiaryl phosphines together; carbonate bases together) and to express **within-family similarity** > **cross-family**. You can **blend** this with numerical descriptor cosine similarities so it isn’t purely text-based.

## 9) Cross-validate priors offline on proxy data

Before your new campaign, check the LLM prior on **archival datasets** from related reactions:

* Fit surrogate with/without the prior on those datasets at tiny data regimes (n=4–12).
* Keep the prompting pattern that reduces cold-start error and improves early AUC most consistently.

## 10) Keep priors and seeds orthogonal

Priors set **beliefs**; seeds are **initial data**. To show value, benchmark:

* Fixed seeds, vary priors (NoPrior vs Heuristic vs LLM-Good/Bad vs RandomPrior).
* Fixed prior, vary seeds (good/bad/random).  
  This disentangles “we started near a good point” from “the model believed the right shape”.

### A minimal recipe you can run today (no extra code needed)

1. Build **GOOD\_BULLETS** and **BAD\_BULLETS** from your manuscript + intuition (we drafted these).
2. Run the LLM readout generator with:
   * **Schema** (bounds/roles)
   * **Descriptor summaries** (optional but powerful)
   * **Bullets** (good or bad)
   * Strict JSON schema
3. **Validate** the JSON, down-weight confidence where uncertainty is high.
4. Plug into PriorBuilder → run the closed-loop benchmark.

### Why this works

LLMs are strongest at **pattern transfer**: they have read thousands of outcomes and mechanistic discussions. By **constraining the output space** (effect types, ranges) and **feeding the right context** (roles, descriptors, constraints, analog exemplars), you turn that diffuse knowledge into a **usable, calibrated prior** that accelerates learning before your first measurement in the new space.

If you want, I can take your latest schema and descriptor tables and produce a **production-grade prompting template** (with self-consistency + critic pass) that drops into your current llm\_readout call.