CIKM 2024
BOISE OCTOBER 21-25

Tutorial on Landing Generative AI in Industrial Social and E-commerce Recsys

Presenter: Da Xu

Organization: LinkedIn, Microsoft, Amazon, Meta, ByteDance

Xu, Da, et al. "Survey for Landing Generative AI in Social and E-commerce Recsys--the Industry Perspectives. "
(this tutorial will be reflected in V2 of the survey paper releasing in Nov 2024)

Contributors

Da Xu
Staff AI engineer
LinkedIn

Danqing Zhang
Founder
Stealth Startup
(with demo in this tutorial!)

Lingling Zheng
Principal scientist
Microsoft

Bo Yang
Applied Scientist
Amazon

Guangyu Yang
Senior ML engineer
Meta

Shuyuan Xu
Research Scientist
ByteDance

Cindy Liang
Head of Network Growth AI
LinkedIn

THE EVOLVING USER NEED FROM RECSYS

- Why people come to the social and e-commerce platforms nowadays --
  - information seeking and discovery (i.e. search & recommendation)
  - complete user tasks (i.e. buy merchandise, get updates, learn about something …)
- What people need from Recsys in these platforms nowadays:
  - better coping with information overload
  - provide explanation and reasoning to shape decisions
  - support action taking

THE EVOLVING USER NEED FROM RECSYS

- Why people need Recsys in these platforms **(THE WHY)**:
  - better coping with information overload
  - provide explanation and reasoning to shape decisions
  - support action taking
- It entails an upgrade from Personalized Suggestion to Personalized Assistance, however:
  - growing level of ambiguity in new problem definitions and complexity to develop the right capabilities
  - despite GenAI's seeming vast potential, what are the opportunity areas and how to facilitate the paradigm shift?

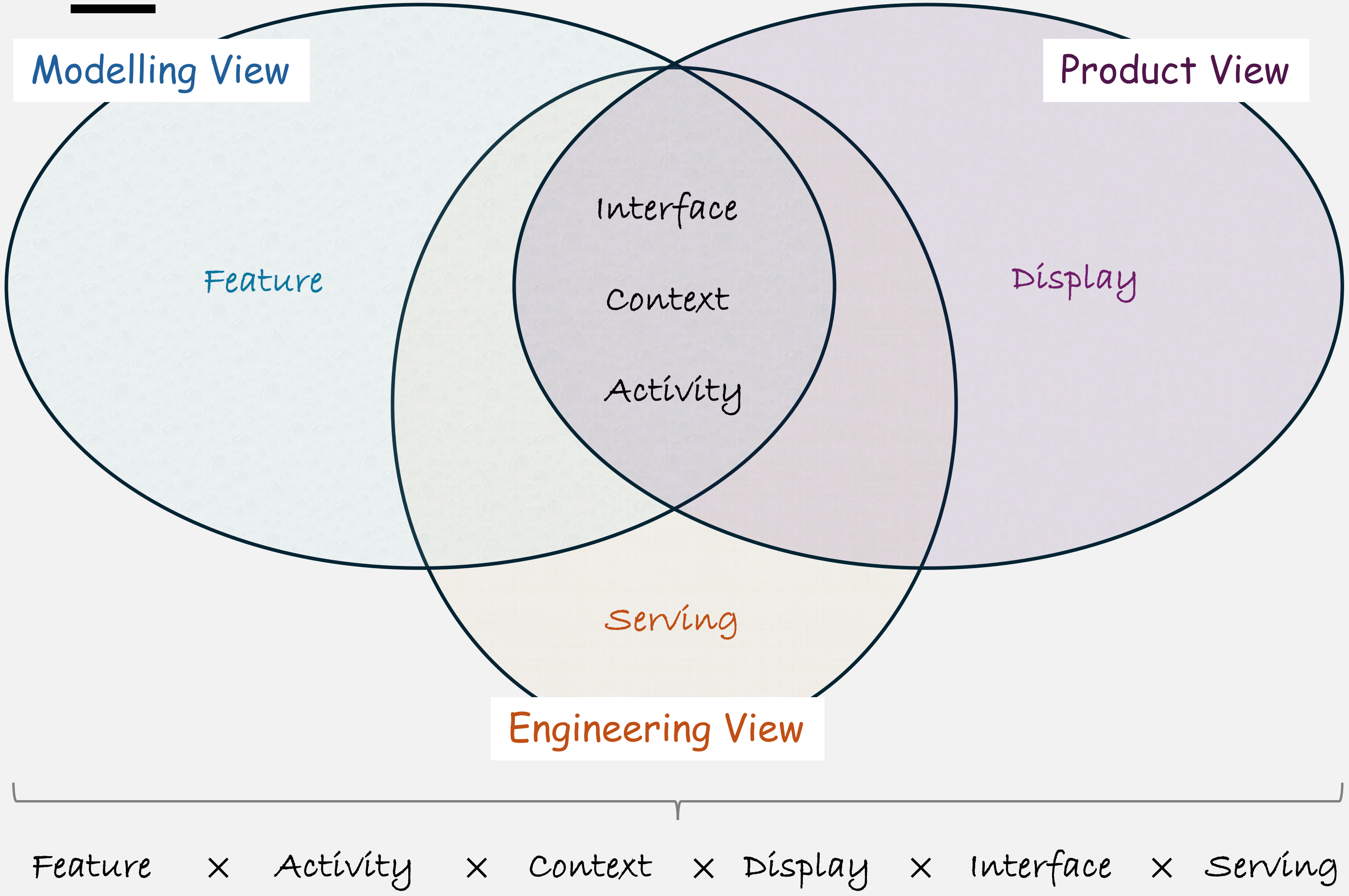
↑ ↑

two contributions of this tutorial: THE WHAT + THE HOW

A QUICK NOTE ON THE EVOLVING SOCIETAL NEED FROM RECSYS (WILL NOT BE COVERED IN THIS TUTORIAL)

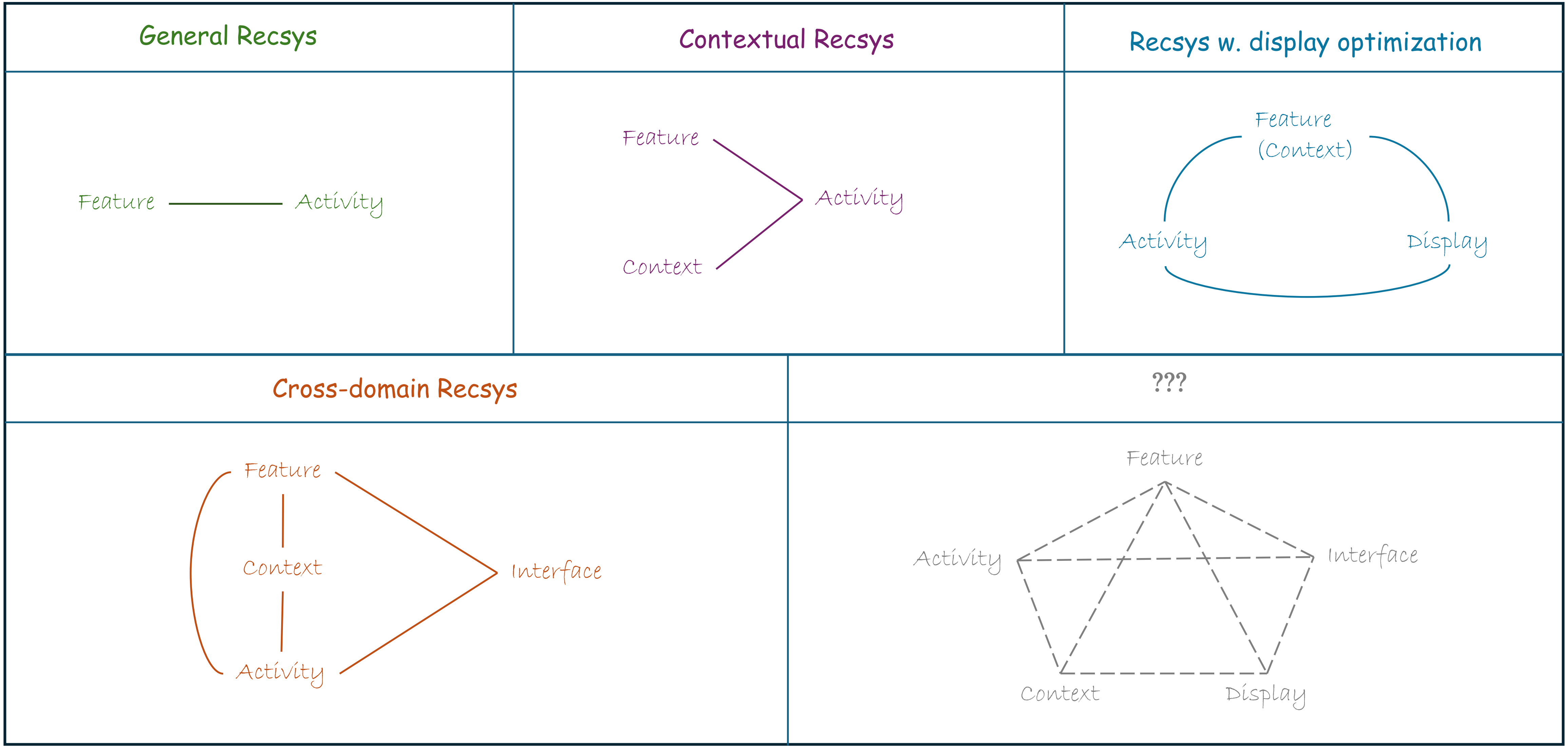
- Societal roles of Recsys nowadays:
  - Connecting creator to audiences (supply to demand)
  - Shaping content creation (supply strategy)
  - Impacting platform / creator economics (market dynamics)
- Similarly, GenAI opens new opportunity areas:
  - AI-assisted marketing analysis and monetization strategy, content and campaign creation …
  - Cross-platform integration
- Also, new challenges arise:
  - Understanding the redistributed competition landscape with AI and AI-guided strategy as the new players
  - The ethics framework need a significant upgrade to ensure the welfares of all parties

To Better Understand how to Support the Evolving User need, Let's Breaking Down Industry Recsys by Functional Pillars



- **Feature**: raveling the characteristics of the user / item, based on which the matching and predictions algorithms can be developed.
- **Activity**: serving as labels and / or signals (e.g. in sequence recommendation) for capturing user explicit / implicit preferences.
- **Context**: consisting of situational features that affects user preference and behavior, but are not part of the user and item characteristics.
- **Display**: the design of visual presentation of selected information to the user.
- **Interface**: the interactive elements and concepts (including feedback and task completion mechanisms) and navigation logics of the system.
- **Serving**: systematically delivering all the above elements and functionalities to users.

To Better Understand how to Support the Evolving User need, Let's Check Out the Evolvement of Recsys Design Patterns



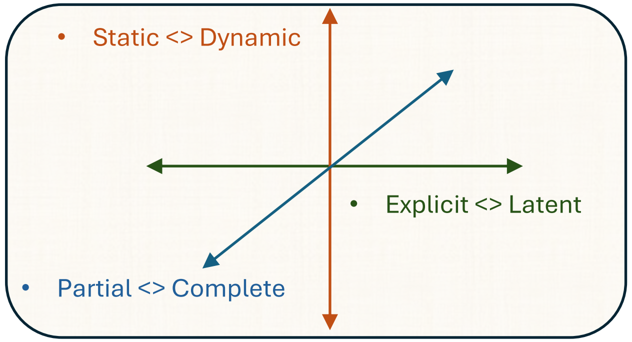
Well-understood as the primary focus area in pre-GenAI era

	Feature × Activity × Context × Display × Interface × Serving
Input	• Numerical • Categorical (inc. ID) • Pairwise • Hidden • Image etc. • … • Matrix • Set • Graph • Sequence • Review • Multi-turn Q&A • …
Method	• Latent factor methods • GNN methods • Sequential methods • Markov Chain methods • Multi-task / meta / online methods • …
NLP application	• Sequence encoder • Natural language encoder • Next item prediction • Conversational • …
Output	• Ranked list • Top-1 • Bundled • Grouped • …

Know contexts are important, but less focused in pre-GenAI era

Feature × Activity × Context × Display × Interface × Serving

Input



Method

- Contextual filtering
- Contextual modeling
- Exploration / Exploitation (RL)
- ...

......

NLP application

- Context-driven query / search
- Text encoding (e.g. review)
- ...

Output

Less understood, less explored in pre-GenAI era (but can significantly impact all sorts of user behaviors!)

	Feature	×	Activity	×	Context	×	Display	×	Interface	×	Serving
Input							• Non-textual, non-interactive (e.g. click, purchase) • …				
Method											
	……				……						
NLP application							• Knowledge extraction / standardization • …				
Output							• Natural language template hydrated with knowledge metadata (e.g. explanation) • …				

Focus on optimizing customized (small) models in pre-GenAI era

Feature × Activity × Context × Display × Interface × Serving

Input

Method

NLP application

Output

....

Input:

- Standard numerical features
- Embedded non-numerical features
- ...

Method:

- Model compression
- Quantization
- Distillation
- Compilation (GPU-based serving)
- Containerization
- API-based serving
- Adaptive batching
- Async processing
- Automated scaling
- ...

Output:

- Structured output
- ...

- better coping with information overload ALL THE TIME (?)
- provide GOOD ENOUGH explanation and reasoning to shape decisions (??)
- CAPABLE OF support action taking (???)

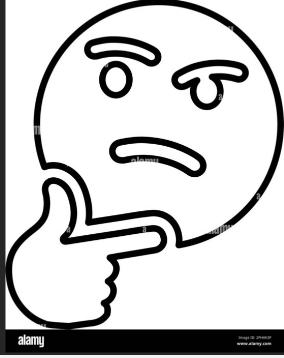
To summarize, existing industrial Recsys:

- Strong personalized filtering and prediction of the available information in existing corpus when abundant structured feature and data are available
- Exhibiting some level of contextual awareness
- Displaying templated knowledge-based justification (persuasion) and reasoning to accompany raw contents
- Focusing on passive preference elicitation interface design concepts (e.g. standardized, non-verbal interaction) with limited user-system interactivity

On the other hand, they suffer from:

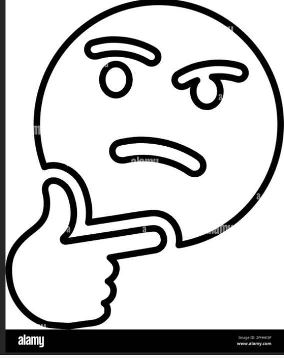
- Performance gap when structured feature and data is scarce (e.g. cold start, multi-modal)
- Lack interpreting nuanced natural language and other complex contexts for rapid adaptation to different scenarios
- No on-demand creation of complex outputs for enriched and personalized display of explanation, reasoning, and coherent content repurposing
- Less diverse, versatile, and engaging interface to enable interactive preference elicitation, critiquing, refinement, and user control

GenAI to the Rescue?



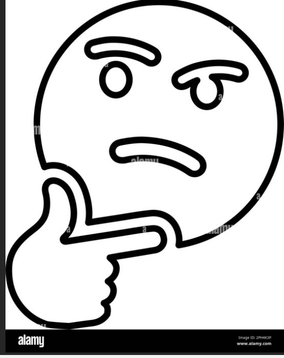
➢ Performance gap when structured feature and data is scarce (e.g. cold start)	⟸	➢ Fill the data gap with LLMs' open-world knowledge?
➢ Lack interpreting nuanced natural language and other complex contexts for rapid adaptation to different scenarios	⟸	➢ Leverage the semantic & multi-modal understanding capability, as well as the zero-shot capability of LLM?
➢ No on-demand creation of complex outputs for enriched and personalized explanation & reasoning display, and coherent content repurposing	⟸	➢ Introduce NL generation components with enhanced system control and reliability (grounded in retrieval)?
➢ Less diverse, versatile, and engaging interface to enable interactive preference elicitation, refinement, critiquing, and user control	⟸	➢ Add verbalized interactive experience (e.g. QnA) with both member-initiated and agent-initiated short-term actions (e.g. via chatbot UI) and long-term actions (e.g. via email / notification UI)?

Not so Fast …



What Personalized Recsys possess:	What Personalized Assistants need:
➢ Treat multi-modality data independently	➢ Jointly handle multi-modality data
➢ Retrieve from item corpus	➢ Retrieve from anything
➢ Specialized models for dedicated tasks	➢ Unified models for all (including zero-shot) tasks
➢ Multi-stage Systems (chain-based)	➢ Multi-component System (graph w. routing)
➢ Human-generated output	➢ AI-generated output
➢ Backend-focused optimizations	➢ Full-stack optimizations

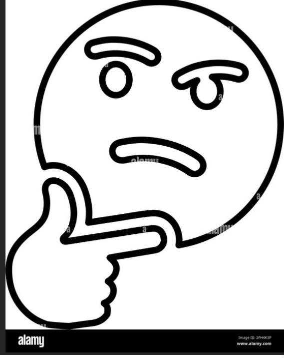
Not so Fast …

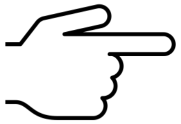


The perspectives are different even for those shared components / pillars:

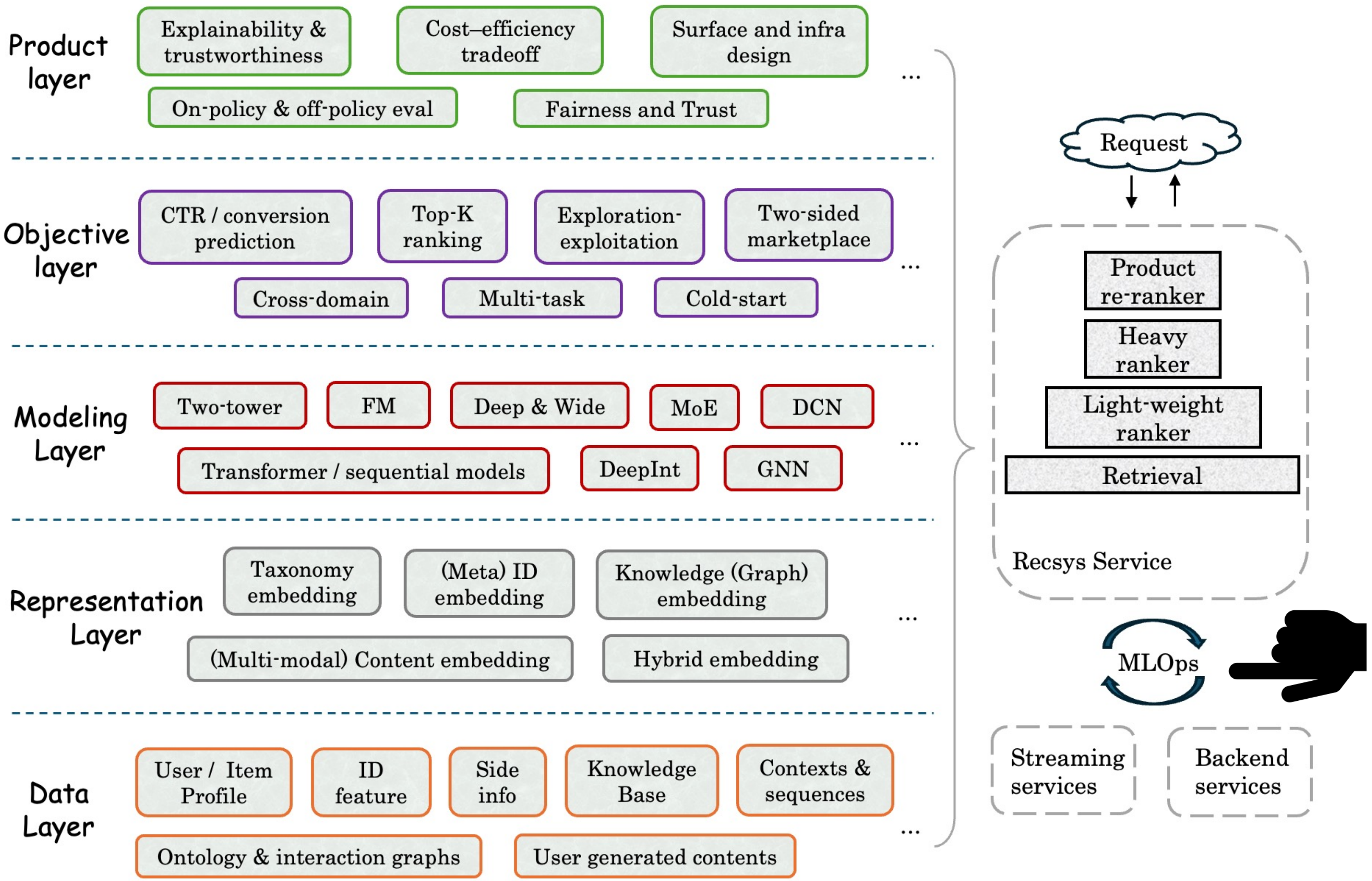
ML model input improvement	Feature engineering	LLM context improvement
Model-driven	Memorization	Database-driven
Structured	Data processing / understanding	Unstructured & multi-modal
Customized	Faster model inference	Transformer/MoE/…-centric
Developer-controlled domain logic	Robust serving	User+agent+developer co-controlled logics
Closed-ended, system-centric	Evaluation	Open-ended, user & agent-centric

And Let's not Forget the Unsung Hero (Hidden Boss?)



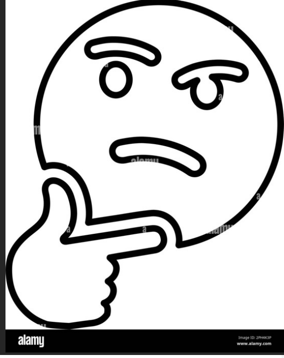


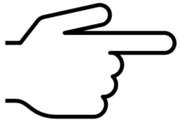
What people think makes us working overnight



What we are actually working on overnight

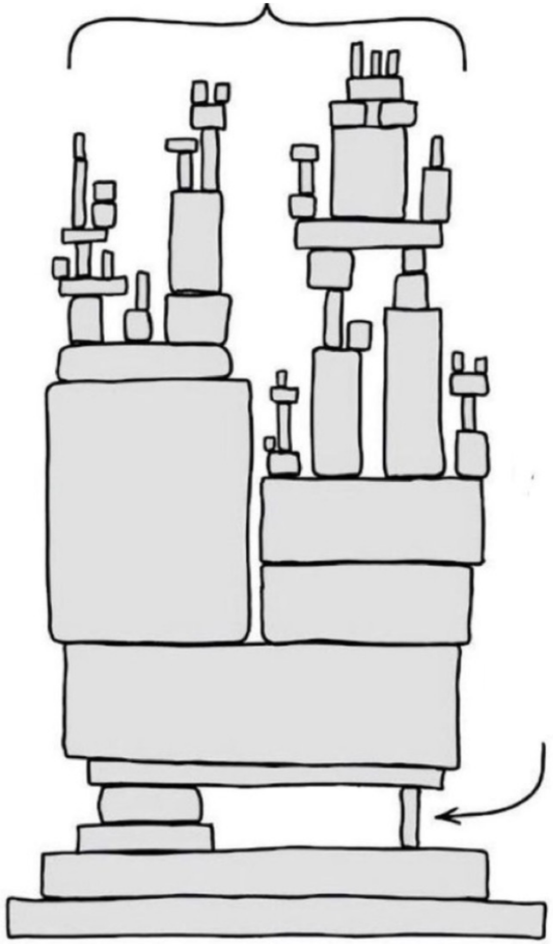
And Let's not Forget the Unsung Hero (Hidden Boss?)





If LLM follows NL instruction and behave as they did in the demos!

Impressive LLM capabilities



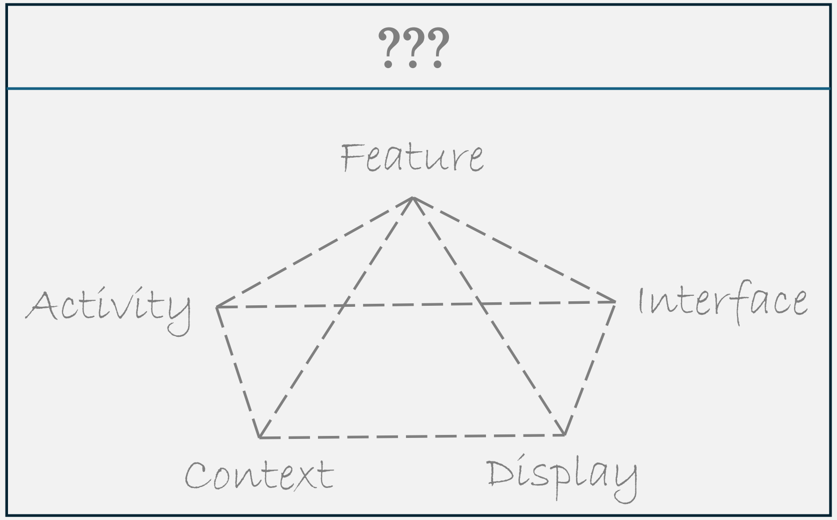
LLMOps

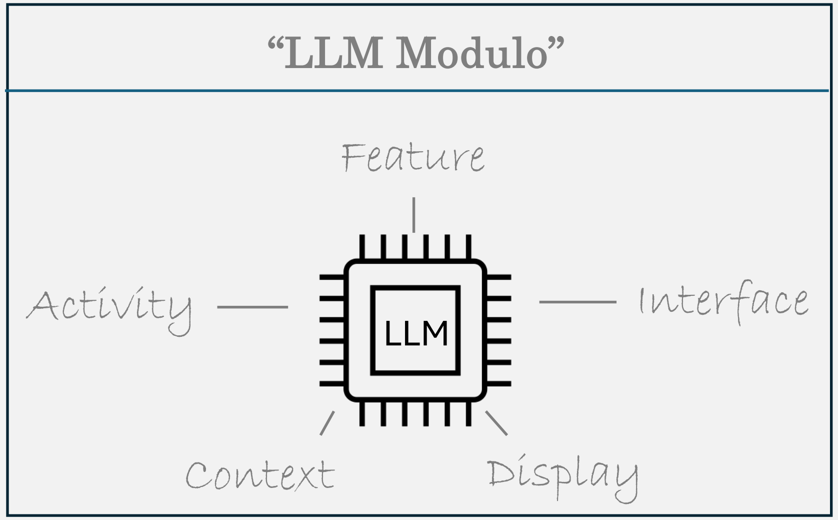
- …
- Prompt-response management
- Embedding / vectorstore / memory database ops
- API gateway management
- Skill registration
- Interface / tooling debuggability
- Messaging
- Autonomous agent orchestration
- Red / blue teaming
- Human-in-the-loop evaluation
- Observability / monitoring
- …

Putting Perspectives Together

Identified the opportunites

- LLM can enhance data & model for core Recsys tasks and applications
- LLM can produce diverse & complex outputs to power new display objectives beyond recommendation
- LLM can facilitate interactive design patterns and functions for advanced user tasks



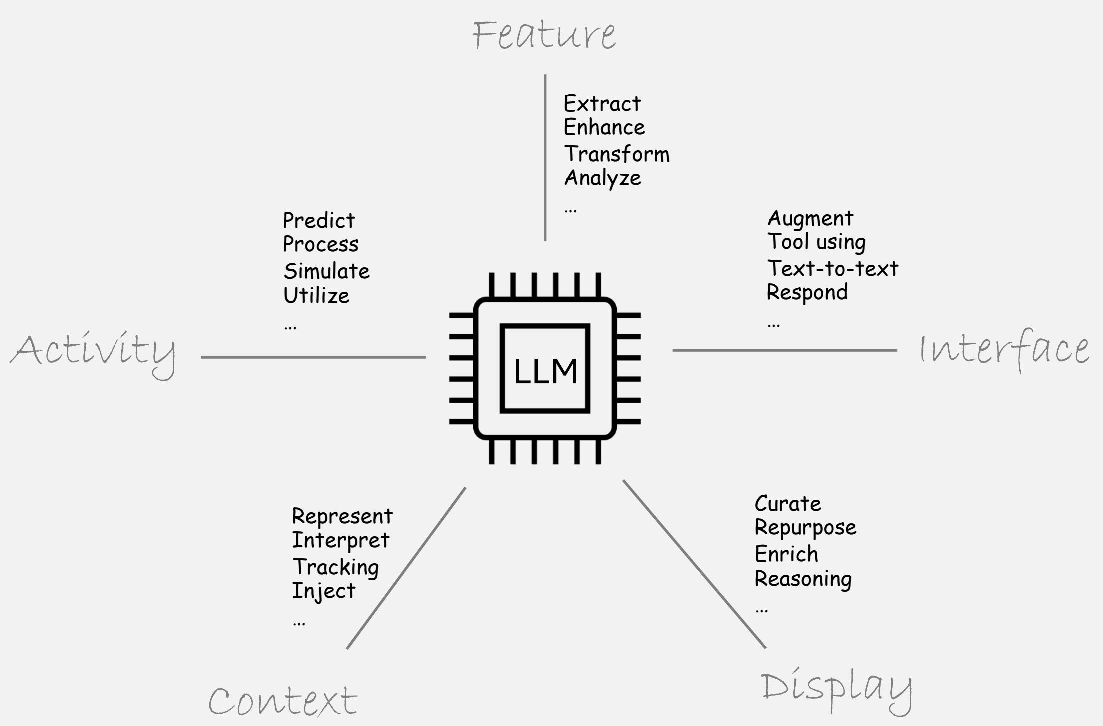


- Merging LLM into existing tasks & applications requires justifying the ROI / consolidating new tech stack with existing ones
- Serving LLM-powered components require dedicated backend / mid-tier / front-end solutions (algo. & infra.)
- Shifting to GenAI system requires new frameworks for design, develop, evaluate, and ops (and reliability, trust & safety)

Now need a roadmap to develop the capabilities

Putting Perspectives Together

The opportunity we identified (w. "LLM Modulo" solution)



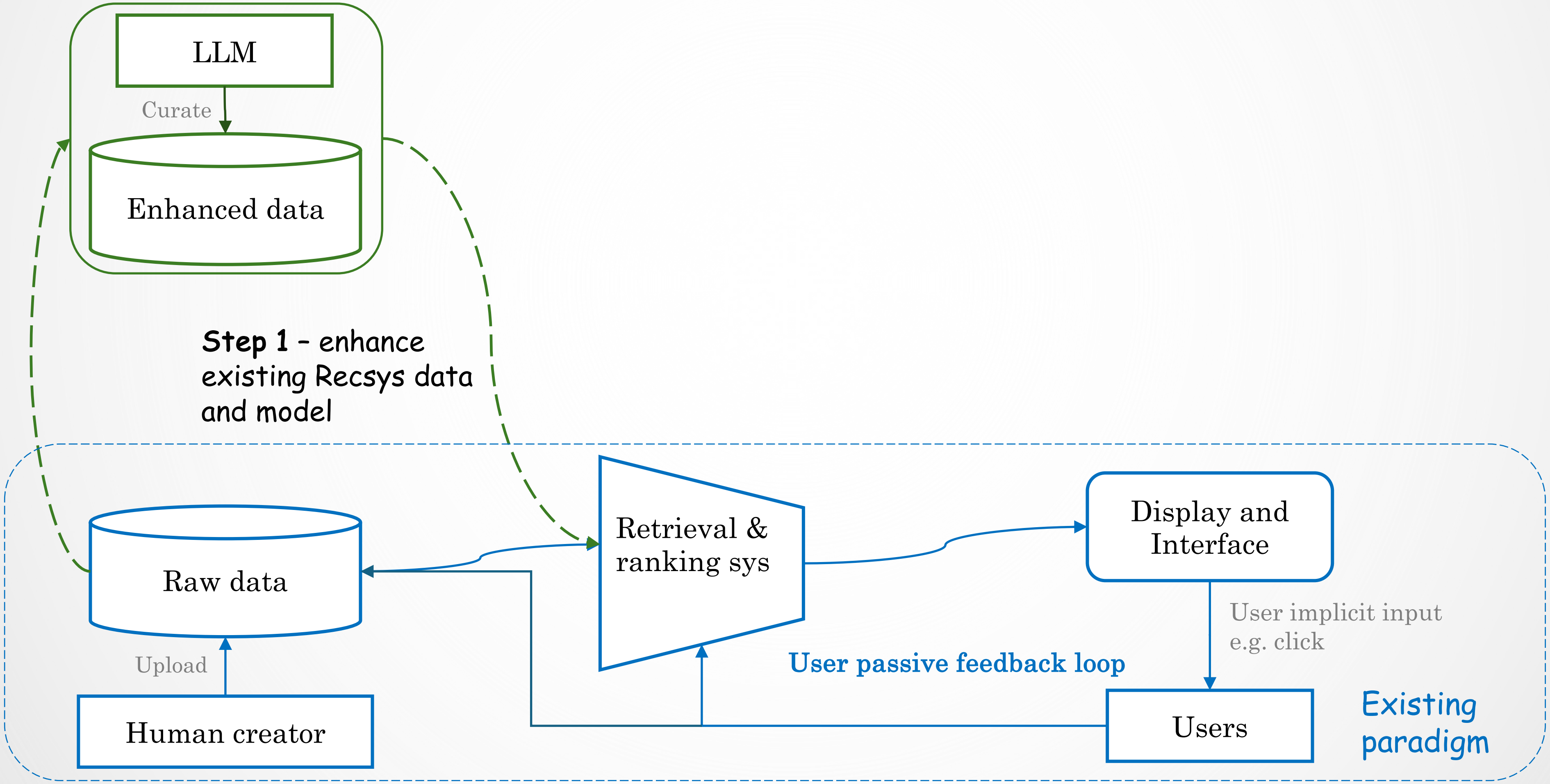
How to get there from where we are?

- The analogy question is: how to replace a running car's engine without stopping it (constraint) / posing safety concerns (risk) / getting pulled over (surveillance)?
- Q1 – how to breakdown the goal into minimum executable steps?
- Q2 - what are the prerequisites and interdependency of the breakdown steps?
- Q3 – risk-aware resource-constraint optimal planning and sequencing?

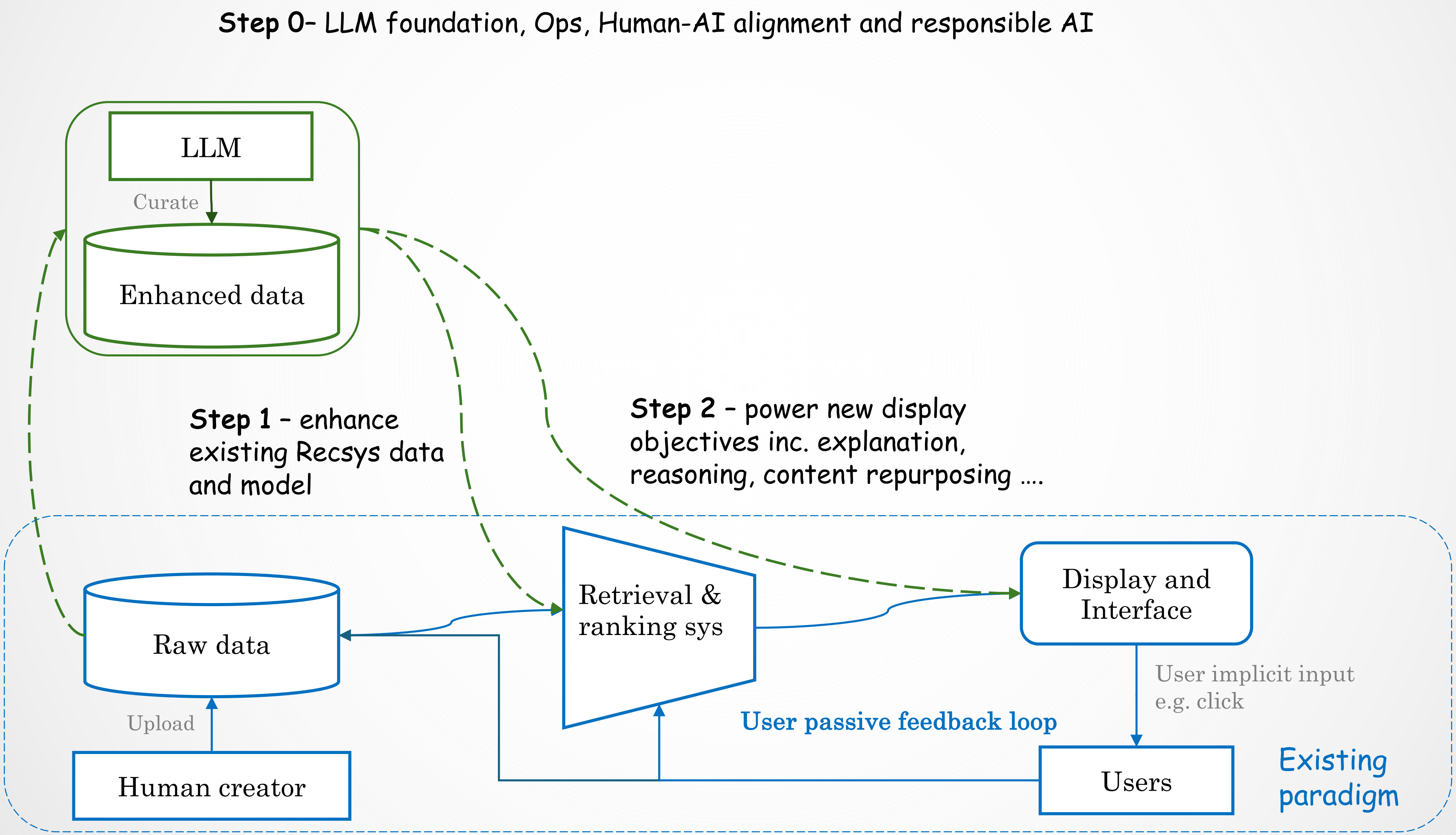
Can talk on it for hours but we bootstrap our solutions into what we call the "Tetralogy"

The “Tetralogy” for landing GenAI in Social and E-commerce Recsys

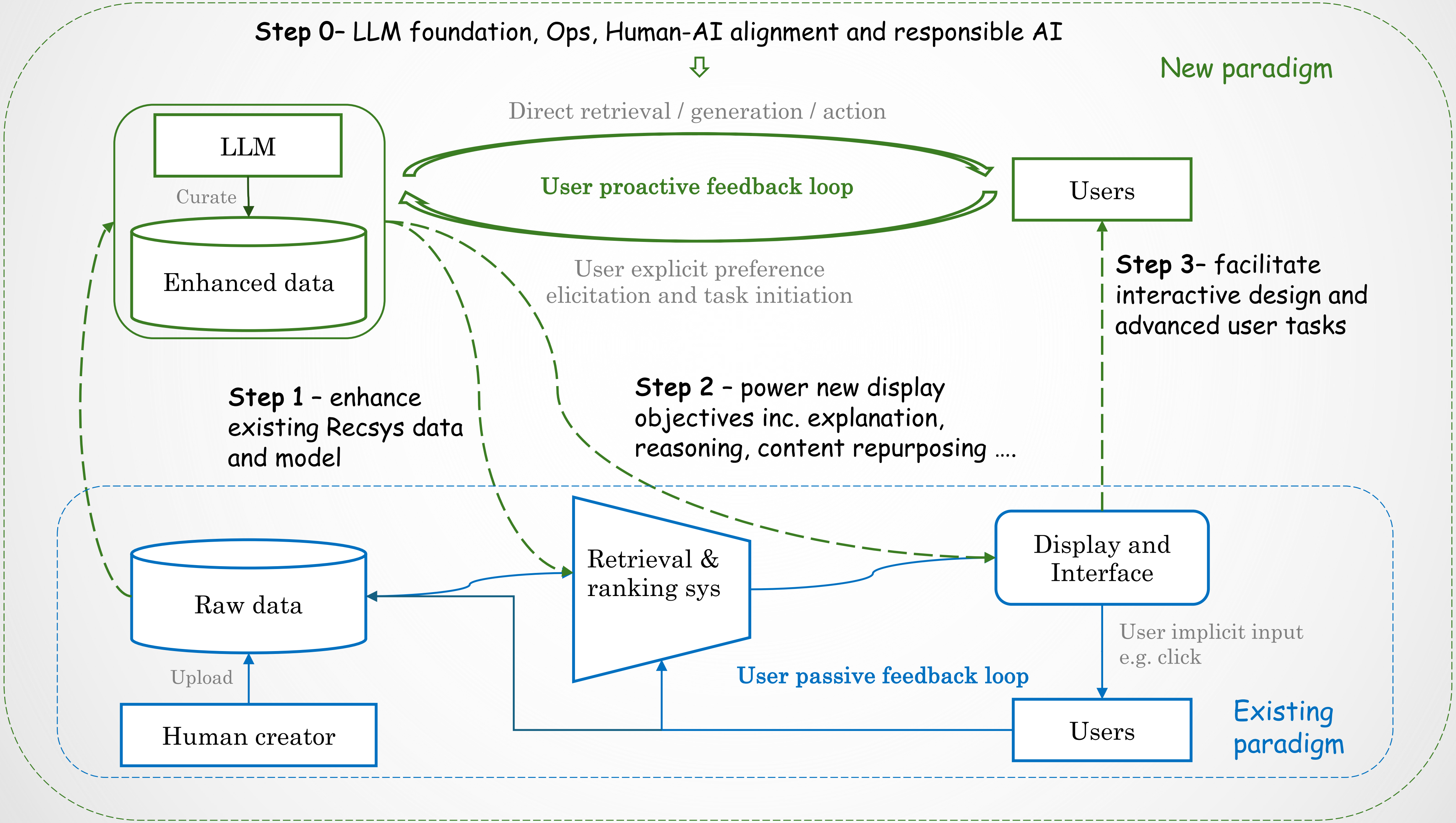
Step 0- LLM foundation, Ops, Human-AI alignment and responsible AI



The “Tetralogy” for landing GenAI in Social and E-commerce Recsys



The "Tetralogy" for landing GenAI in Social and E-commerce Recsys



Outline for the Next Sections

- Step 0 – LLM foundation and Ops
- Step 1 – Enhancing existing Recsys data and model
  - Case study 1: LLM as cold-start candidate generator
  - Case study 2: Semantic ID
  - Case study 3: Unifying semantic search and contextual recommendation
- Step 2 – Enabling complex display objectives
  - Case study 1: RAG for personalized explanation & reasoning
  - Case study 2: Display (creative) optimization with bandits
- Step 3 – Facilitating interactive design and complex user tasks
  - Case study 1: Register Recsys as Tools
  - Case study 2: Agent call patterns
- Step 0 (‘cont) – Alignment and Responsible GenAI
  - Case study: Multi-modal GenAI in Recsys

Heavily Abbreviated History

	NLP	IR & Recsys
50s	Shannon model	
60-80s	Earliest chatbot, statistical language models	Exact retrieval, indexing
90-00s	Dedicated language modeling	Vector space / probabilistic model
00-10s	Representations, word embeddings	Personalization, learning to rank
10-20s	LSTM, RNN, Transformer, BERT	Solving evolving user / business needs with new technologies
2020+	”Bitter lesson”, “scaling law”, “emerging capabilities”	

How LLMs are Built

Data cleaning: Preprocessing, Denoising, Outlier, Dedup, Synthetic, …

Tokenization: Byte Pair, Word Piece, Sentence Piece, …

Position encoding: Absolute position, Relative position, Rotary, With bias, …

Architecture: Encoder-only, Decoder-only, Encoder-decoder, …

Pre-training: Masked, Causal, Next sentence, MoE, …

Tuning: Supervised, Contrastive, Instruction following, …

Alignment: Supervised, RLHF, DPO, …

Adaptation: LoRA, Distillation, Prompting, …

Inference: Top-k sampling, Top-p sampling, Greedy search, Beam search, …

How LLMs are (usually) Categorized

By Size

- Small <1B
- Medium 1~10B
- Large 10~100B
- Mega 100B+

By Tuning

- Untuned (original)
- Foundational (tuned not for instruction following)
- Instruction
- Chat
- …

By Enhancement

- Multi-modal
- Long-context
- Expanded token
- Domain expertise (e.g. Text2sql, tool-using, planning, law, medical, educational, …)
- …

Some Known Limitations of LLM and Augmentation

- Stochastic
- Staled
- Lacking state tracking / memory
- Hallucinate
- Safety / privacy / integrity
- Very bulky to train / serve

Final sections

More on this next

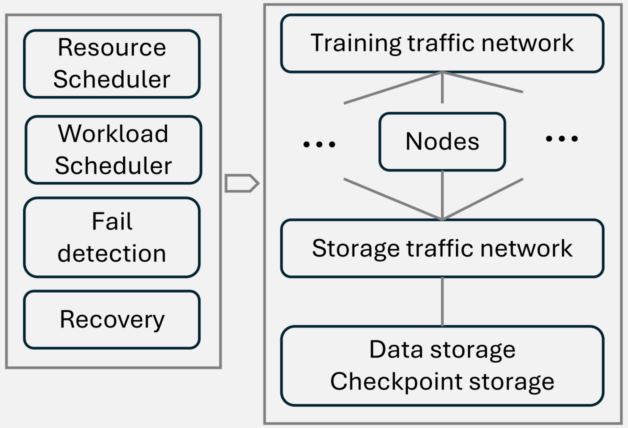
- Generic prompt augmentation
  - CoT, ToT
  - Self-consistency, reflection
  - Automatic prompt optimization
  - Rails (fact-checking, Jailbreaking)
  - …
- Add external knowledge (RAG)
  - Retrieval from Vectorstore / DB
  - Reranking / chunking …
  - RAG-aware prompt augmentation
  - …
- Use external tool
  - Tool registration / API calls
  - Tool-aware prompt augmentation
  - …
- Equip with Agentic flows
  - Reason and act, reasoning without observation
  - Multi-agent control system
  - …

LLM Training Optimizations

Goal:

- Less memory consumption
- Faster computation
- Better hardware utilization
- Higher success rate

Distributed training infra



Parallelism

- Data
- Tensor
- Pipeline
- Expert
- Sequence

Computation Optimization

- Operator optimization
  - Manual optimization (e.g. FlashAttn.)
  - Auto optimization (e.g. kernel level)
- Mixed precision training

Memory Reduction

- Activation re-computation
- Redundancy reduction
- Defragmentation (partially, fully)
- Offloading
  - CPU
  - SSD

Management Optimization

- Communication optimization (often the bottleneck for large GPU cluster!)
- Scheduler with network topology awareness
- Fault tolerance
  - Detection
  - Recovery

LLM Inference Optimizations

Goal:

- Lower computational cost
- Lower memory access cost
- Lower memory cost

⇩

- Better latency
- Better throughput
- Better storage

Data-level optimization

- Input compression
  - Prompt pruning
  - Context summary
  - Context compression
- Output organization

Model-level optimization

- Distillation
- Quantization
- Pruning
- Sparsification
- Structural optimization
- …

Algorithm-level optimization

- Speculative decoding
- Parallel decoding
- Early exiting
- …

System-level optimization

Memory management & caching

- Efficient KV cache
- Compression of KV cache
- Prompt cache
- Semantic cache
- Infinite LLM
- …

Task scheduling

- Request batching
- Disaggregated inference
- Distributed system
- …

Kernel optimization

- Sampling optimization
- Support variable length
- Graph and operator optimization
- …

LLM Evaluation and Observability

- Generic NL Evaluation
  - Entropy, perplexity, …
  - Functional correctness, relevancy, coherence
  - Similarity with reference data (BLUE, ROGUE …)
- Domain Evaluation
  - E.g. Recsys / IR task evaluation
  - E.g. QnA task evaluation
- Open-ended Generation Evaluation
  - Insutruction-following capability
  - Factual consistency, faithfulness, safety …
- New Evaluation Methods
  - LLM-as-a-judge
    (note: criteria ambiguity, inconsistency, cost …)
  - Comparative evaluation
    (note: lack scalability, standardization, interpretation …)
  - …

- Observability: Metrics
  - System metrics (e.g. throughput, memory usage, HW utilization, service availability / uptime)
  - Model performance metrics (e.g. accuracy, hallucination rate, length-related metrics …)
  - Latency metrics (e.g. time to first token, time between tokens, tokens per sec, time per output token, total latency …)
- Observability: Logs
  - Raw input / output
  - Hydrated prompts
  - Retrieved contexts
  - Intermediate steps
- Observability: Traces
  - Flow executions
  - API calls
  - … (see Langsmith!)

Putting Things Together (for now…)

LLM Foundation

	Development	Serving
Prompt engineering	Meta-prompts	Templating
	Design and publish prompt	In-context learning
	Prompt eval, tuning, testing	Grounding
Models	LLM pre-training / tuning	Algorithmic acceleration
	Offline evaluation	AI system and architecture optimization
	Learning from feedback Continual learning	Online monitoring / alarming
	Non-GAI models	Supplement / edge function

\+

LLMOps

Delivery pipeline

- LLM selection
- Adaptation to user task
- Database management
- External data integration
- GenAI evaluation
- Cost & latency optimization
- Observability

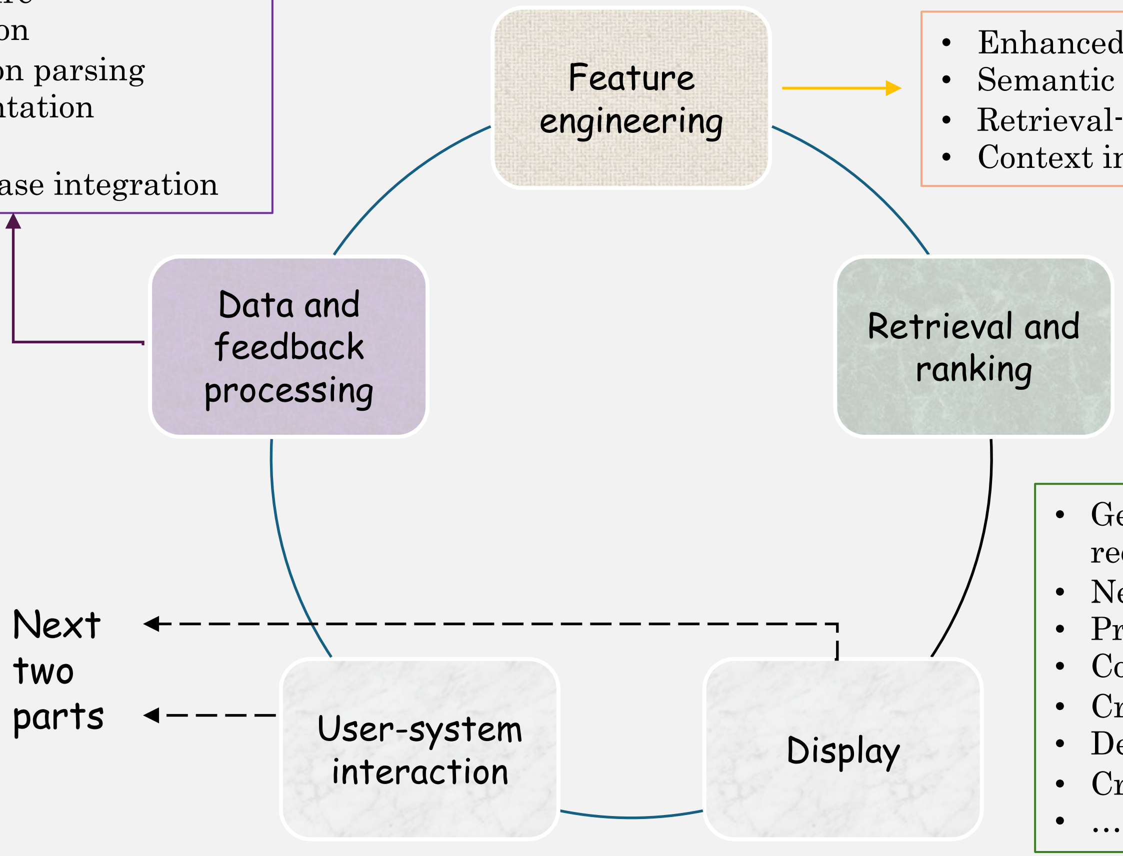
Feedback loop

⇨

Get ready to build your first LLM solution in Recsys!

Overview

- Textual feature transformation
- NL interaction parsing
- Data augmentation
- Simulation
- Knowledge base integration



- Enhanced (multi-modal) representation
- Semantic ID
- Retrieval-augmented
- Context interpretation

- Generative retrieval and recommendation
- Next-item prediction
- Prompting as ranking
- Cold-start/few-shot problem
- Cross-domain recommendation
- Dense retriever
- Cross-encoder methods
- …

Prompt engineering + fine tuning LLM are often sufficient to produce well-rounded solutions for most of these tasks!

Productionizing Prompt Engineering Solutions

"Auto" prompt optimization

- What: build prompt abstractions and automatically refining prompts with *gradient-guided* or *LLM-assisted* frameworks
- Why: manually tuning prompts for black-box LLM is laborious and more like an art then science, and continuous optimization is challenging.

Meta Prompts

- What: organizing and packaging prompt template and relevant (model) parameters.
- Why: versioning the definition and configuration for prompt engineering in a unified fashion for integration and compatibility purposes.

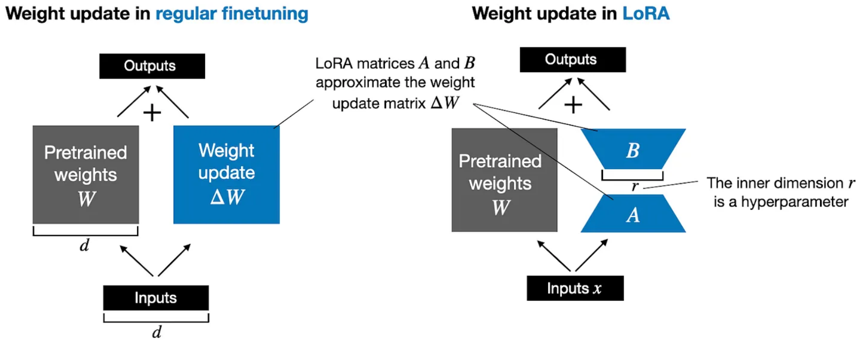
Prompt serialization

- What: seamless reading / writing prompts and metadata to and from file in production env.
- Why: ensure data consistency across applications, projects, environments, and CI/CD pipelines in a structured fashion.

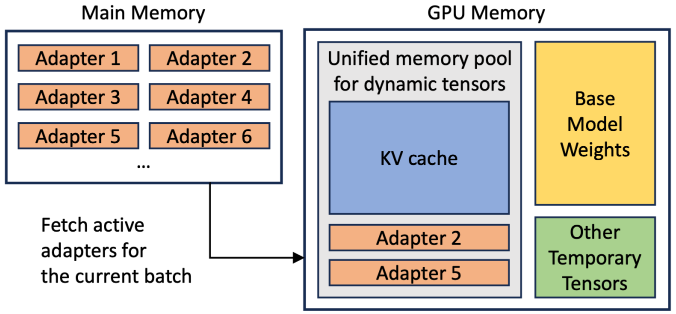
Wen, Yuxin, et al. "Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery."

Building and Serving LoRA LLM

Training



Serving



Why LoRA

- Flexible, stable, and effective parameter-efficient LLM tuning
- Versatile serving
  - Can serve multi-LoRA on single GPU
  - Can concurrently serving LoRA adapters for single and multiple requests

Practical considerations

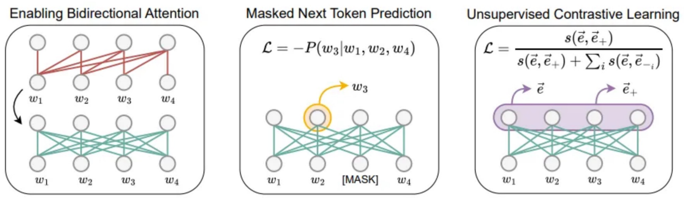
- Be aware of the compute-memory tradeoff with quantization
- Avoid “catastrophic forgetting” (overfitting for fine-tuning) with learning rate scheduler and controlling the tuning epochs / steps
- Picking the right modules to target (the more the better for linear/proj. layers?)
- Balancing LoRA parameters r and alpha

Hu, Edward J., et al. "Lora: Low-rank adaptation of large language models.”
Sheng, Ying, et al. "S-lora: Serving thousands of concurrent lora adapters."

Case study 1: LLM as cold-start candidate generator

Scenario: how to retrieve real-time candidates for a member (with profile data) who started a session but has very few historical interaction records?

"LLM2Vec"



- Remark: 1). Text-to-embeddings for the profile and session contexts to facilitate online KNN; 2). BERT models often don't possess the context length, open-world knowledge, and instruction-follow ability of the more recent LLMs.

Prompt optimization

- What: identify the optimal prompt template (for both member and item) that optimizes recall performance before tuning the LLM with in-house data.
- Remark: construct the best prompt configurations before tuning the model for maximum efficiency.

Fine-tuning LLM

- What: collect in-house (user profile, context) to item engagement data to hydrate the NL training data, and tune the foundation model with contrastive loss in the Siamese setup.
- Remark: inject domain patterns and knowledge into the foundation LLM for optimal retrieval performance.

Wang, Liang, et al. "Improving text embeddings with large language models."

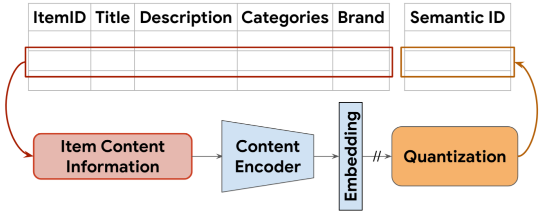
Step 1 – Enhancing enhancing existing Recsys data and model

Case study 2: Sematic ID

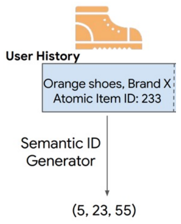
Scenario: how to more effectively encode large corpus of items in a semantically meaningful way so they can be integrated into LLM and downstream models ?

"Semantic ID"

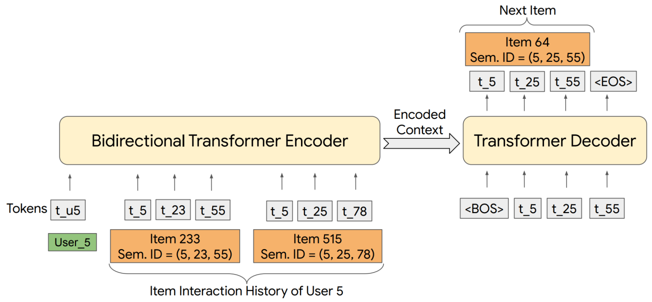
- What:



- Remark: quantizing item representations with multi-level codebook:



Generative retrieval

- 

- Remark: treating semantic IDs as **tokens** and apply NL techniques (e.g. seq-to-seq modeling) such that the generated tokens can be mapped to items.

Semantic ID in ranking model

- What: replace the atomic ID in ranking model for improved memory efficiency and generalization.
- Remark: semantic ID is a more memory-efficient and generalizable representation of large user / item corpus. (WHY?)

Sun, Weiwei, et al. "Learning to tokenize for generative retrieval."
Rajput, Shashank, et al. "Recommender systems with generative retrieval."

Case study 3: unifying semantic search and contextual recommendation

Scenario: given their growing similarity in the problem space and the capacity of LLM, can we re-define query & context to facilitate unified solution for semantic search and contextual recommendation?

"NL context"

- What: a broader definition that includes explicit textual query, textualized scenario context (inc. user / item contexts).
- Remark: the goal is to cross-pollinating the semantic modeling of explicit (structured) query and personalized modeling of implicit (unstructured) contexts via LLM.

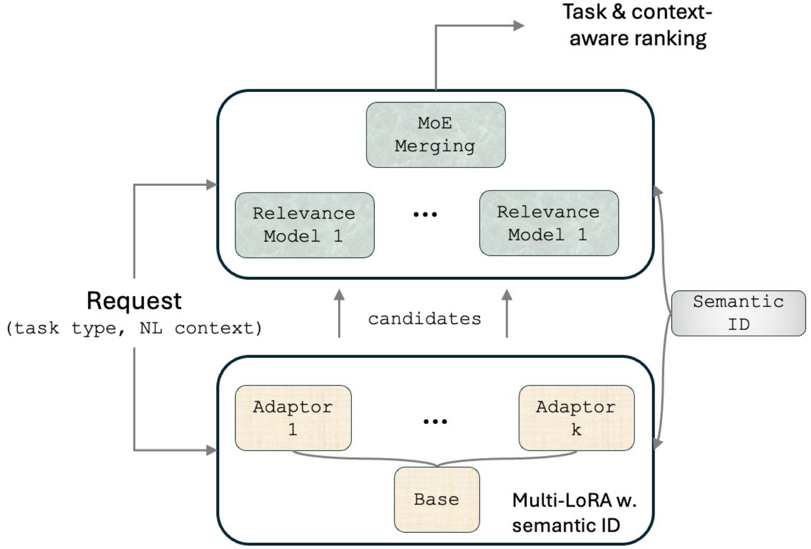
Semantic ID backbone

- Remark : the unified semantic representation is applicable to all tasks under NL contexts.

Multi-LoRA support

- Remark : enable a unified fine-tuning and serving stack to capture the specificity of each NL context in retrieval phases.

Unified ranking stack



Note: in practice, the retrieval phase is always multi-source with such as term-based retrieval still playing critical roles. Now, with all the good items in the plate, how do we serve a "visual feast" to the users?

Recsys Display In a Nutshell

- Presenting raw recommendation is not enough:
  - Mismatch between the representation of the suggestion versus users' information need
  - Need techniques for automatic generation of satisfactory explanation & reasoning & insights that are intelligible (UNDERSTANDABLE) for users interacting with the system
- But, understanding is rarely the end goal
  - Need to operationalize the effectiveness of explanation & reasoning & insights in terms of a specific notion of usefulness or display goal (e.g. improved particular decision support, reduce the cost of a specific type of error …)
  - Explanation vs. transparency vs. justification
    - Explanation don't have to be transparent to the underlying algorithm
    - A justification explains why a decision is a good one, without explaining how it was made
- Re-purposing raw contents (e.g. title rewrite) or generating new contents (e.g. homepage images in e-commerce) are also optimizing specific notions of usefulness or display goal.
- Finally, keep in mind that the "real estate" is limited especially on mobile Apps!

Recsys Display In a Nutshell

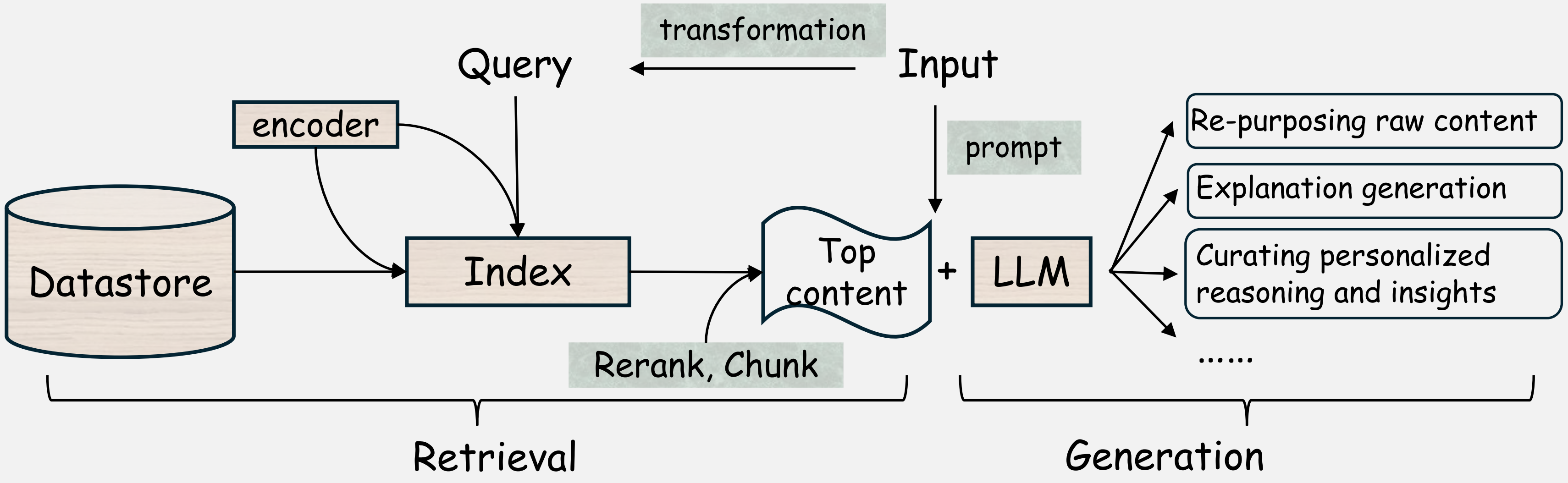
Display Goal	Definition	Comment
Transparency	Explain how the system works	Establish visibility to the system status
Trust	Increase user confidence	Mitigating the effect of poor recommendation
Scrutability	Allow user to tell when system is wrong	Establish user control
Effectiveness	Help user make good decisions	Depend on the algorithm, also useful for introducing new domains and help understanding full range of options
Efficiency	Help user make faster decisions	Usability principle: understand which suggestion is the best, how quickly a task can be performed
Persuasiveness	Convince user to try	Attempt to influence user
Satisfaction	Increase ease of use	Aid the satisfaction with the reco process and recommended suggestions without adding cognitive efforts
Stakeholder goals	Coherence with system welfare	

Recsys Display In a Nutshell

- Three generic levels of explanation & reasoning in social and E-commerce Recsys:
  1. Individual-user level
     - Using raw data the platform has on the user (including history)
  2. Contextualization level
     - Establishing relations to anything that's not in user / content raw data but affects user behavior, e.g. situational feature, preference space, other users (neighbors), …
  3. Self-actualization level
     - Moving beyond information-finding and promote discovery and exploration to fulfill personal / societal values and goals
  4. And of course, the hybrid style

- What information to use? How to obtain them? How to use the obtained information?
  - Retrieval-augmented generation (RAG) is a powerful technique for these challenges.

RAG Overview



Sources of Information

- Unstructured datastores and structured knowledge bases / graphs (most common);
- Real-time contexts;
- Various plugins for combining with domain knowledge and results
- ...

Three key questions

- What to retrieve?
- How to retrieve?
- How to use & serve retrieved contexts?

Practical Painpoints

- Information missing from retrieval;
- Useful information isn't consolidated into context;
- Having useful information in the context, but end up not specified / in wrong format / hallucinated in the response;
- Response is too generic / incomplete;
- ...

Case Study 1 – RAG for personalized explanation and reasoning

What to retrieve?

Individual user level (insights in relation to people background)

- Content-based explanation
  - Consider similarity between content attributes / properties based on user behaviors
  - Keywords, tags, topics...
- Case-based (influence) reasoning
  - Detailed contents are omitted and focus on considering cases for comparison
- Knowledge / Utility reasoning
  - Reasons over knowledgebase can overlap with the above styles for achieving certain utilities

Contextualization level (insights in relation to a context)

- Collaborative reasoning
  - Adding persuasion from neighbors (assuming there's already some interests)
- Action reasoning
  - Extrapolating other explanation styles into the action space
- Blind-spot explanation
  - Contextualization in relation to the overall space
- ...

Self-actualization level (insights in relation to personal values / goals, the reasoning can have more impact than the recommendation itself)

- Goal-directing explanation
  - Suppose we have user labels for goal / intent understanding
- User-controlled explanation
  - Writing actions controlled by user
- Broaden-the-horizon (educational) reasoning
- Discover-the-unexplored explanation

Case Study 1 – RAG for personalized explanation and reasoning

How to retrieve?

Embedding-based retrieval

- Versatile with abundant established solutions (including LLM2Vec)
- Can take advantage of the VectorStore advancements

Indexing and matching

- Methods like BM25 are still the key players for many types of queries

Content-based filtering

- Methods like TF-IDF and PMI-based retrieval are effective with reasonable performance and good interpretability for certain tasks

Collaborative filtering retrieval

- Good at capturing similarity patterns from interaction data for certain styles of explanation / reasoning

Generative retrieval

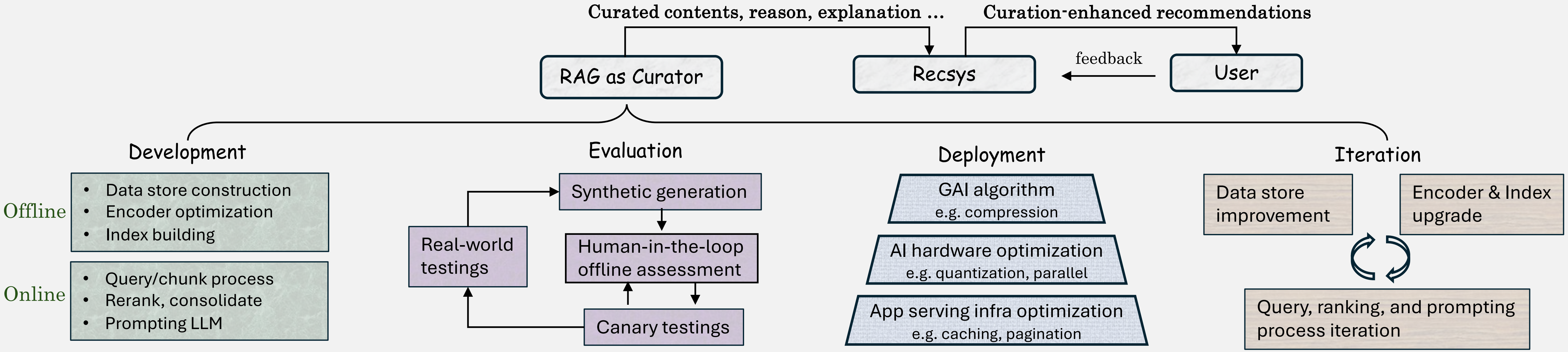
- The new retrieval paradigm, supplementing existing methods in long-context scenarios

And don't forget to invest in…

- Encoder, query transformation
- Chunking / aggregation strategy
- VectorStore operations
- Adaptive & recursive retrieval
- Retrieval from external sources
- … (agentic RAG flows)

Case Study 1 – RAG for personalized explanation and reasoning

— How to use retrieval depends on product design and how well the system is productionized.



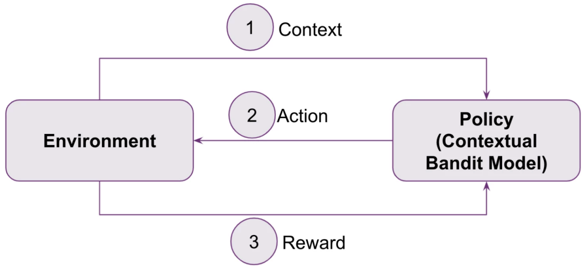
Remark: as the display component of an online system, we want the product design and RAG strategy to evolve and adapt quickly to users' needs. **But how do we know what might work??**

Case Study 2 – Display (creative) optimization with bandits

Scenario: how to effectively probe and adapt display strategies to individual user needs under various (evolving) contexts?

Defining the bandit problem

- What: (contextual) bandit is an algorithmic framework that learns optimal decisions by balancing **exploration & exploitation** (while considering contextual information for each decision).



Policy optimization

- Contextual information to use
  Richer and richer with the new definition of NL contexts.
- Policy structure
  Linear structure is best established; Neural nets are more suitable for the unstructured NL contexts.
- E/E strategy
  Epsilon greedy, Thompson sampling
- Policy learning and evaluation
  Off-policy learning & eval with logged data

Challenges in practice

- Define the right problem (RAG problem space is very large with a lot of configurations, the problem space for copy optimization is often simpler)
- Evolving problem space
- Delayed / indirect reward
- Non-stationary environment
- Runtime uncertainty
- Sensitivity of off-policy eval
- ...

With good items in the plate and a visual feast, now, how to arrange a satisfactory interaction experience for the users?

Interactive (conversational) Recsys before the LLM Era

User Actions

- Ask for recommendation
- Input preference
- Inquire information
- Answer system problem
- Provide feedback to recommendation
- Input critique
- ...

Intent Understanding
(with conventional MLSys)

Belief and state tracking
(read / write <-> DB)

What capabilities are lacking in this paradigm?

System Actions

- Request information
- Provide recommendation
- Answer question
- Provide explanation / reasoning
- Inform and acknowledgement

Generate response
(model-free template-based generation)

Interactive (conversational) Recsys before the LLM Era

Cognition

- Current paradigm: limited by conventional ML sys' capability to fulfill the intent recognition tasks and interpreting complex context.
- Fix: use LLM's open-world, generalization, and zero-shot capability for fine-grained and adaptive cognition tasks.

Memorization

- Current paradigm: unable to effectively process complex dialogue states, user history, and other unstructured / semi-structured data formats.
- Fix: introduce structure / format conversions and on-demand transformation with LLM, and VectorStore as short-term / long-term DB solutions.

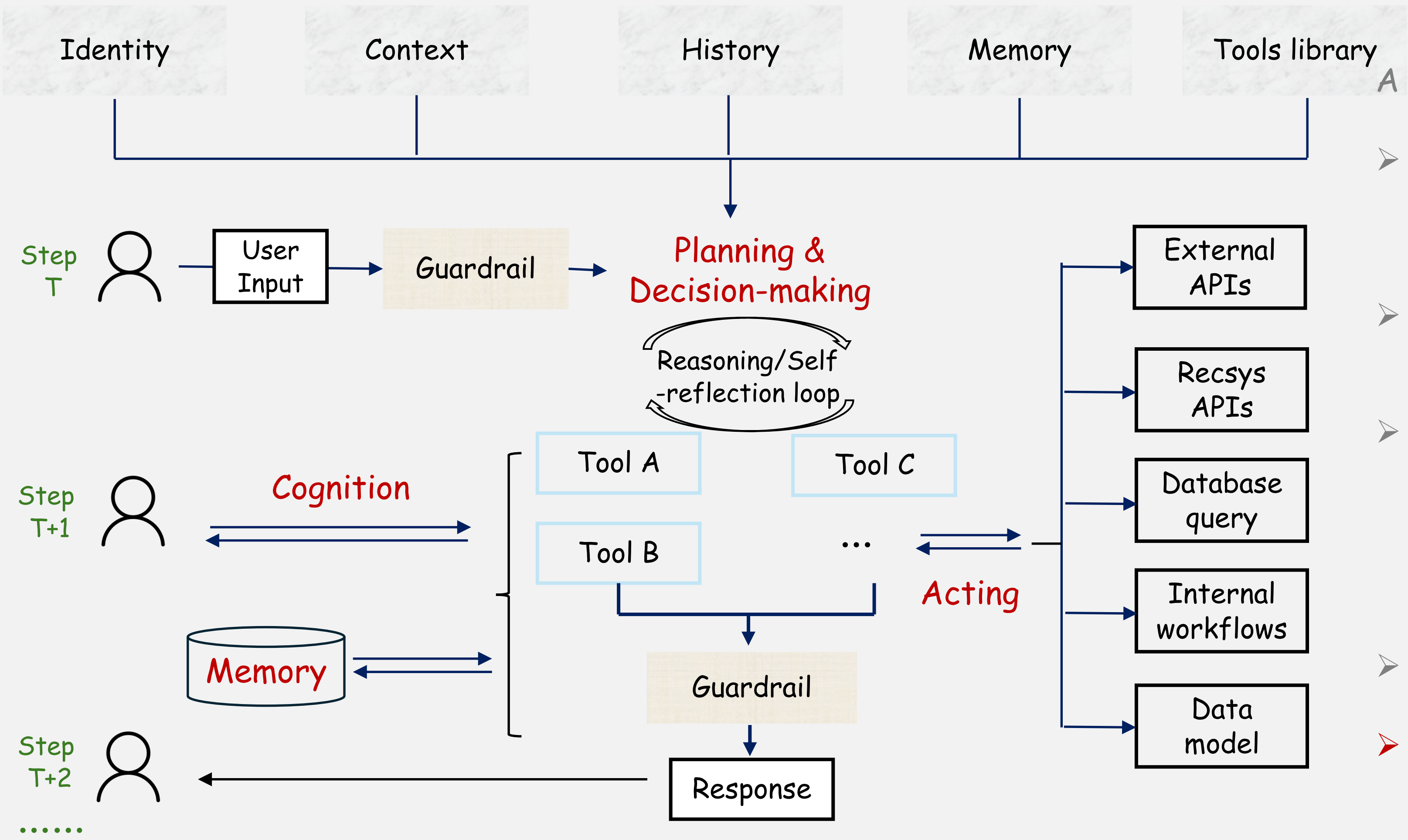
Decision making

- Current paradigm: using hard-coded routing logics so the system itself cannot plan and make decisions adaptively under all contexts and user actions.
- Fix: combining hard-coded logics with the preliminary reasoning capability of LLM for *generic* task planning and adaptive routing & decision-making.

Acting

- Current paradigm: restricted within the dialogue system, mostly reactive rather than proactive, and not interfacing with external tools and systems for read, write, and more complex operations.
- Fix: leverage LLM's API interfacing capability for tool access, utilization, and more complex system operations like system and user-initiated actions.

Integrating the New Capabilities



A monster control system:

- How many Agents (specialized entity that perform a specific complex task) are needed?
- How many Tools (atomic operation with well-defined inputs and outputs)?
- How to orchestrate?
  - Task -- decompose, execute, …
  - Agent -- creation, communication…
  - Resource -- memory, computation…
  - Workflow – chaining, sequencing, ….
- How to evaluate & optimize online / offline?
- Need a holistic framework to enhance capability, scalability, flexibility, resource efficiency, fault tolerance.

Multi-agent Framework

Design, manage, orchestrate, and coordinate multiple AI agents to work together on complex tasks.

Specialized sub-agents

- Each sub-agent is specialized in a given task (e.g. with dedicated large or small LLM as backbone)

Distributed

- Each agent operates independently as a microservice
- Enhances flexibility and scalability

Standard communication

- Agents communicate through standardized API interfaces and protocols
- Centralized control panel & message queues

Organized tool registry

- Enable unified creation, discovery, configuration, experimentation of tools.

Graph structured w. router

- Graph representation with nodes as agents and edges as communications, cycles enabled;
- Router (e.g. played by agent) as the controller for main state transitions.

Other important components

- Access control
- Conflict resolution
- Messaging system
- Memory / caching
- Observability
- Error handling …

Case Study 1 -- Register Recsys as Tools

Scenario: after building the unified semantic & contextual Recsys, in the agentic framework, it needs to be registered as a tool in order to be discovered, invoked, managed, and experimented.

Discovery

- What: the Recsys can be matched to the appropriate task by the Agent.
- How: provide and refresh well-documented Tool description and definitions to the short-term and long-term memories.

Invocation

- What: Agent can easily and correctly call the Recsys API and process the input & output.
- How: 1). adopt the standardized communication protocols, 2). building versatile Tools (e.g. that can take any NL context).

Management

- What: managing the lifecycle of Recsys including description change, upgrade, deprecation, etc.
- How: versioning the Tool and simultaneously handling referencing and rollout (inc. hierarchical tools).

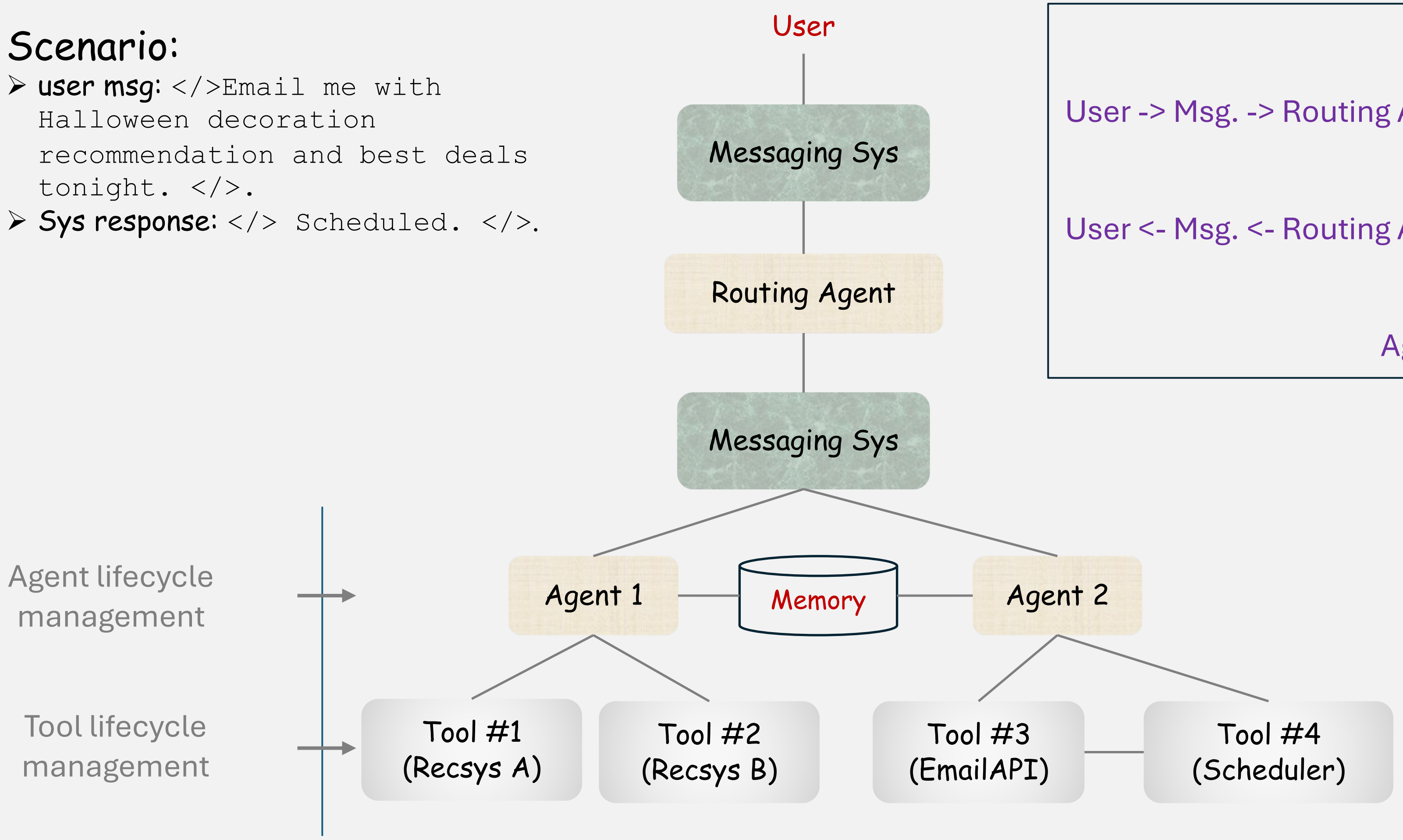
Experimentation

- What : A/B testing the impact of a Recsys upgrade in the agentic system.
- How: maintain both the control and treatment variants of the Tool in the discovery, invocation, and management cycles.

Case Study 2 – Agent Call Patterns

Scenario:

- user msg: `</>Email me with Halloween decoration recommendation and best deals tonight. </>.`
- Sys response: `</> Scheduled. </>.`



Roughly speaking…

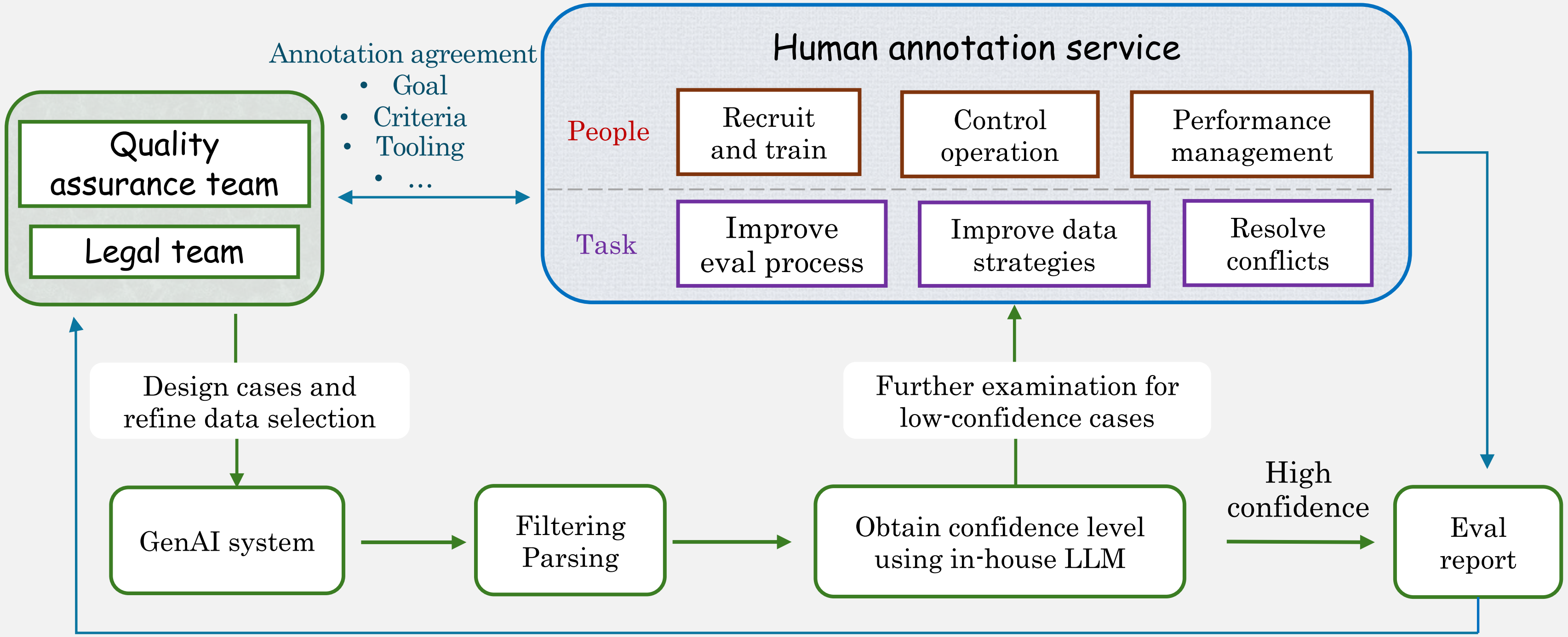
User -> Msg. -> Routing Agent -> Msg. -> Agent 1 -> SyncCall(Tool#1, Tool#2)
|
|
User <- Msg. <- Routing Agent <- (Msg. , Memory) <- Agent1
|
|
Agent 2 -> Memory -> AsyncCall(Tool#3, Tool#4)

Remark:

- in reality, we don't need to do such traversals for all requests by leveraging (in-memory) caching solutions
- other typical service optimizations include load balancing, parallel / async. processing, and message-queue-DB and even API designs (e.g. gRPC vs. REST)

Beyond evaluating LLM: Human-in-the-loop for GenAI systems

We have talked about:
- Why to evaluate: assessment (reward and risk), selection, guard railing …
- What to evaluate: generic NL tasks, domain-specific tasks, generation evaluation …
- How to evaluate: benchmark, scoring models, LLM-as-a-judge (jury of LLM) … Human-in-the-loop?



AI-Human Alignment in a Nutshell

What is alignment?
-- Telling model / system what is safe, what is helpful, what is harmful, what is useless ...... such that they can always behave in the intended ways.

Data-driven approach

- Debasing
- High-quality human feedback
- Augmentation
- Prompt improvement
- Filtering

Algorithmic approach

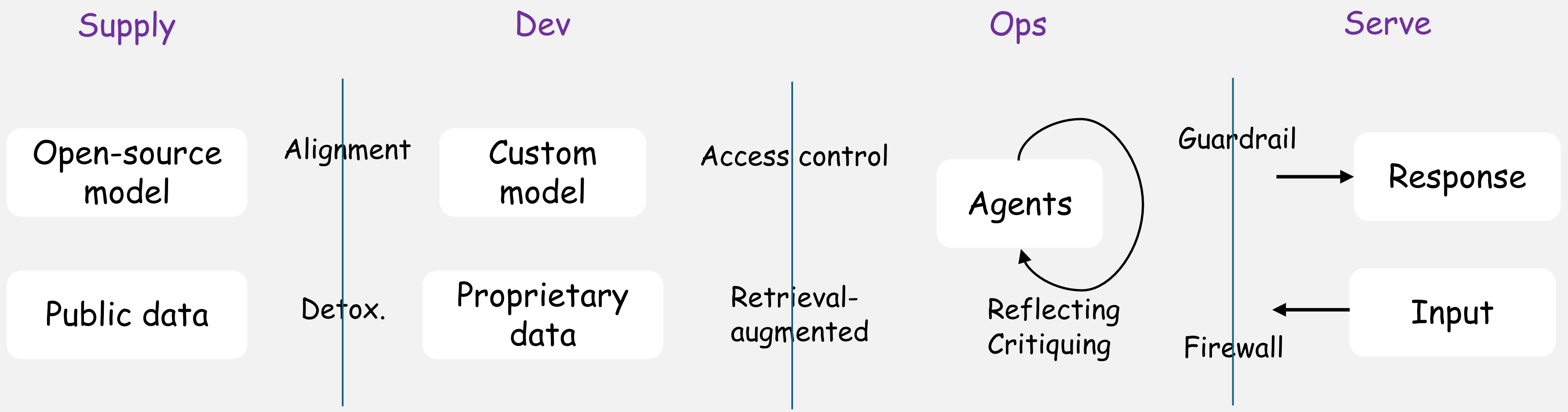
- Inference-time Scaling
- Calibration
- Improve learning objectives
- Controlled generation
- Debasing
- Prompt improvement

HCI design

- Filtering
- Add Interpretability & Reasoning
- Inference-time Scaling
- Feedback mechanism
- Controlled generation

Defend and Build Trusted GenRec System

Multi-layer defense (the ideal setup)



Challenges in reality:

- Many of these are resource-intensive operations (both offline and online)
- How to scale up and reduce redundant operations?
- How to effective handle the evolving threat landscape and governance …

Defend and Build Trusted GenRec System

Red and blue teaming (the ideal setup)

Red-team defense

- Reliability
  - Correctness / consistency / robustness
- Security
  - Jailbreak / injection / theft …
- Safety
  - Privacy / toxicity / bias / fairness / ethic …

Offering rigid solution

Agent

LLM | Plan
Context | Reason
Memory | Tool
……

Offering adaptive solution

Blue-team defense

- Model routing
  - By expertise / performance / trustworthiness / cost …
- Guardrail
  - Accuracy / latency / …
- Fine-tuning
  - Security / reliability / fairness / bias / ethic …

Challenges in reality:

- Similarly, many of these are resource-intensive operations (both offline and online)
- Lack of consensus on standardization for developing and testing many of the components
- Data scarcity issues (just like fraud detection)
- Aligning with the regulatory compliances …

Putting Things Together (finalized)

LLM Foundation

	Development	Serving
Prompt engineering	Meta-prompts	Templating
	Design and publish prompt	In-context learning
	Prompt eval, tuning, testing	Grounding
Tools	Tool / API registry	Chain orchestration
	Skill creation and testing	Reasoning
	Publishing and discovery	Planning
Content & memory	Content indexing	EBR (unstructured) query
	DB onboarding	DB & API query
	API onboarding	Conversation management
Models	LLM pre-training / tuning	Algorithmic acceleration
	Offline evaluation	AI system and architecture optimization
	Learning from feedback Continual learning	Online monitoring / alarming
	Non-GAI models	Supplement / edge function
Responsible AI	GAI response risk assessment	Red teaming
	Human-in-the-loop development	Access control and moderation service
	Data collection/annotation	Controlled generation

+

LLMOps

Delivery pipeline

- LLM selection
- Adaptation to user task
- Database management
- API / Tool registry
- External data integration
- GenAI evaluation
- Cost & latency optimization
- Observability
- Human-in-the-loop
- Trust and reliable AI
- Chain orchestration
- Defense
- Human-AI alignment

Feedback loop

Get ready to build your first Personalized Assistant!

Case Study: Multi-modal GenAI in Recsys

Landing multi-modal GenAI in Recsys covers all the elements discussed in this tutorial.

Categorization

- **Contrastive**:
  Learn multi-modal representations that are aligned in the embedding space (e.g. CLIP);
- **Generative**:
  Learn latent structure of multi-modal data generation process (e.g. VAE, Diffusion models).

Applications

- **Improving recommendation** (e.g. Multi-modal representation learning and retrieval)
- **Improving display** (e.g. image refinement, description / review generation)
- **Improving interactivity** (e.g. virtual try-on, in-context visualization)

Model Building

- **Pretraining** – contrastive or generative pretraining of multi-modal encoder-decoder model on large multi-dim corpus with fusing techniques.
- **Tuning** -- often instruction-tuned and fine-tuned (e.g. with LoRA and conditioning) for domain alignment & controlled generation.

Integration

- **As a tool**:
  e.g. taking (processed) textual input and output the generated images.
- **As a controller (Agent)**:
  e.g. take prompt and multi-modal data as input, and generate task decomposition and call sub-agents.

Harm and Risks

- **Misinformation** (curation vs. creation)
- **Manipulative danger** (false persuasion)
- **Safety concerns** (especially when images are involved)
- **Societal bias**
- **Legal issues** (e.g. infringement)
- **More challenges in evaluation** (lack of data, difficult to audit ...)

Defend and Trust

- **Recap:**
  - Multi-layer defense on the (Input, Response) layer, Ops layer, Dev layer, and Supply layer
  - Red teaming as the rigid solution
  - Blue teaming as the adaptive solution
  - ...

Some Interesting Open Problems

- Online/offline evaluation, observability and monitoring
  - Goal: developing a parallel process to measure and maintain the efficacy of the GenAI components in the system
  - Challenges: brittle metrics, measuring generated responses at scale, ensuring reproducibility, longitudinal analysis, coming up with appropriate monitoring and experimental designs for both user satisfaction and system efficiency
- Seamless multi-modality integration and serving
  - Goal: effectively collect, process, understand the relationship, and produce coherent outputs that 1). Incorporate patterns from various input types; 2). Enable natural and diverse human-computer interactions
  - Challenges: obtaining high-quality data, data integration complexity, synchronizing and alignment, need more advanced pre-training / tuning techniques, standardization and interpretability, serving scalability ...

➢ GenRec assistant with persona

➢ Synthetic data generation and integration

➢ Renovated HCI design elements and concepts

Some Critical Open Problems

- Effective in-house GAI serving stack
  - Goal: hitting the optimal tradeoff between performance, cost, and latency with trust/safety control
  - Challenges: managing fragmented tech stacks, catching up with new solution ideas, addressing data silos and safety concerns, scaling applications with resource constraints ...
- Acquiring high-quality human data at scale
  - Goal: providing the fuel for all stages in the GAI development cycle (unlike Recsys which can leverage a wide range of user feedback)
  - Challenges: cost of human annotation, procedural complexity of training and workforce management, task and criteria design ...

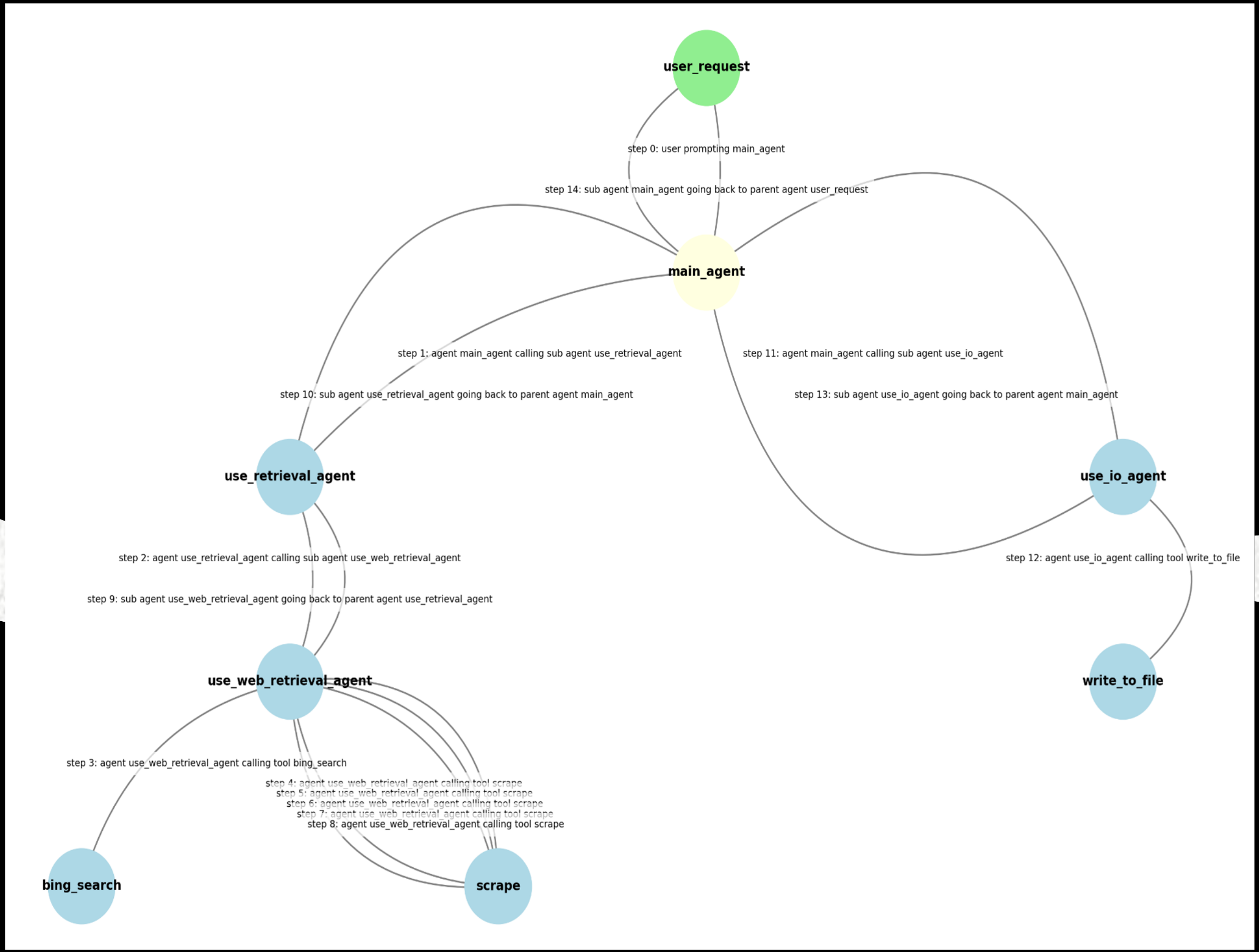
- ➢ Consistent quality, reliability, and high success rate (especially for Agentic systems)
- ➢ All the unresolved privacy-related issues
- ➢ Real LLM4Planning and life-long learning capability

Final Remarks

- Can’t emphasize enough on data
  - “Garbage in garbage out” is still a very real thing in the LLM era
- Viewpoints on “LLM for planning”
  - Depends on whether the question has already been answered in the prompt?
  - Remains largely an open field of study, be cautious in production …
- Challenges of CI/CD in this fast-evolving problem space
  - Similar to what deep learning practitioners have experienced before 2015, troubled by tool migration and maintenance issues?
- On the growing computational capacity
  - How much to count on “the bitter lesson”, “scaling law”, and “emerging capability” to set long-term goals and visions?
- Adopt a framework v.s. build your own
  - No free lunch
- How AI teams and initiatives could be more effectively organized in the future?
- Reflecting on some ongoing Agent initiatives (feat. @Danqing Zhang)

Product 1 -- LiteMultiAgent (@Danqing Zhang)

- LiteMultiAgent
  - https://github.com/PathOnAI/LiteMultiAgent -- a hierarchical multi-agent system
  - Highlights -- hierarchy of agents (where high-level agents use sub-agents as tools through function calling) such that the execution of sub-agents is *parallelized* by parallel function calling.
  - System components
    - `Tool Registry`: register custom functions, sub-agents as tools
    - `AgentFactory`: Creates agent instances, with different agent class and agent type
    - `AgentManager`: Manages agent interactions and hierarchies.



Registering a Tool

register sub agent as tool example

```
ToolRegistry.register(Tool(
    sub_agent.agent_name,
    sub_agent,
    sub_agent.agent_description,
    {
        "task": {
            "type": "string",
            "description": sub_agent.parameter_description,
            "required": True
        }
    }
))
```

Creating an Agent Hierarchy

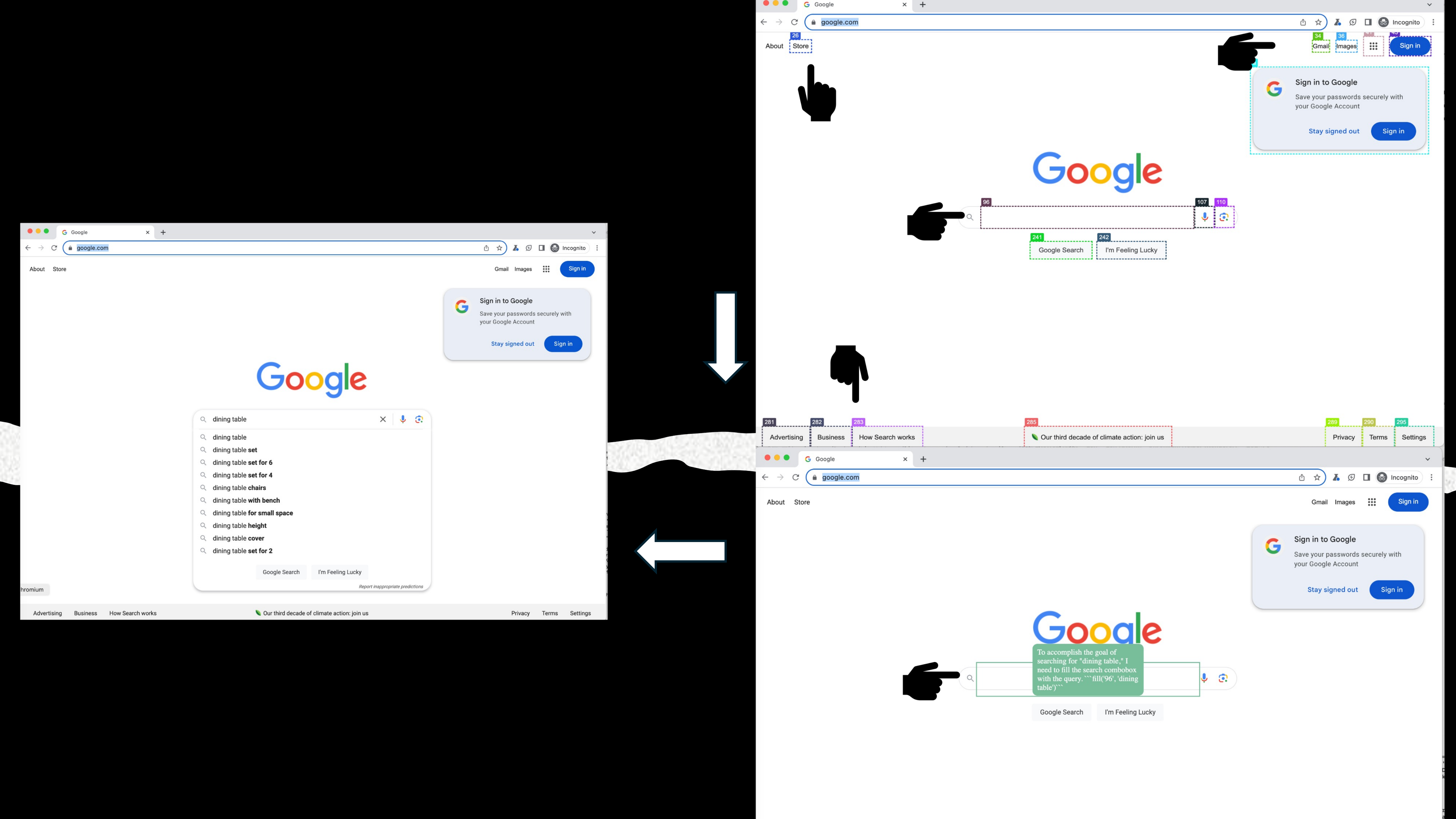
```
retrieval_agent_config = {
    "name": "retrieval_agent",
    "type": "composite",
    "agent_class": "FunctionCallingAgent",
    "meta_data":
        {
            "meta_task_id": meta_task_id,
            "task_id": task_id,
            "save_to": "csv",
            "log": "log",
            "model_name": "gpt-4o-mini",
            "tool_choice": "auto"
        },
    "tools": [],
    "sub_agents": [
        web_retrieval_agent_config,
        file_retrieval_agent_config,
        db_retrieval_agent_config
    ],
    "agent_description": "Use a smart research assistant to look up information using multiple sources including web
    "parameter_description": "The task description specifying the information source (web search, database, local fi
}
retrieval_agent = agent_manager.get_agent(retrieval_agent_config)
```

Setting Up an Agent

```
agent_manager = AgentManager()
web_retrieval_agent_config = {
    "name": "web_retrieval_agent",
    "type": "atomic",
    "agent_class": "HighLevelPlanningAgent",
    "meta_data":
        {
            "meta_task_id": "web_retrieval_subtask",
            "task_id": 4,
            "save_to": "csv",
            "log": "log",
            "model_name": "gpt-4o-mini",
            "tool_choice": "auto"
        },
    "tools": ["bing_search", "scrape"],  # Changed from "write_file" to "write_to_file"
    "agent_description": "Perform a search using API and return the searched results.",
    "parameter_description": "The task description describing what to read or write."
}
web_retrieval_agent = agent_manager.get_agent(web_retrieval_agent_config)
```

Product 2 -- LiteMultiAgent (@Danqing Zhang)

- LiteWebAgent
  - https://github.com/PathOnAI/LiteWebAgent
  - Highlights
    - decouple action generation and action grounding
      - action generation
      - action grounding
    - flexible framework to incorporate different types of agents with strong baseline
      - planning agent that replans based on action execution
      - context-aware high-level planning
      - prompting agents
    - first open-source framework that includes search agent for web browsing
      - implemented basic BFS/ DFS search agent
      - built solid framework and extended to the MCTS, LATS search agent for web browsing
    - user interface, demo effect and browser


Google
google.com
About
Store
Gmail
Images
Sign in
Incognito
Sign in to Google
Save your passwords securely with your Google Account
Stay signed out
dining table
dining table set
dining table set for 6
dining table set for 4
dining table chairs
dining table with bench
dining table for small space
dining table height
dining table cover
dining table set for 2
Google Search
I'm Feeling Lucky
Report inappropriate predictions
Advertising
Business
How Search works
Our third decade of climate action: join us
Privacy
Terms
Settings
To accomplish the goal of searching for "dining table," I need to fill the search combobox with the query. ```fill('96', 'dining table')```

THANK YOU, Enjoy CIKM'24 and Boise :)

Q&A

We are always just one email / LinkedIn DM away :)

Presenter contact: daxu5180@gmail.com

Interested in Danqing's startup projects? Contact: danqing.zhang.personal@gmail.com

Xu, Da, et al. "Survey for Landing Generative AI in Social and E-commerce Recsys--the Industry Perspectives. "
(this tutorial will be reflected in V2 of the survey paper releasing in Nov 2024)