OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY

Contemporary Generative AI and Agentic Workflows: Retrieval-Augmented Generation, Fine-Tuning, and Skills

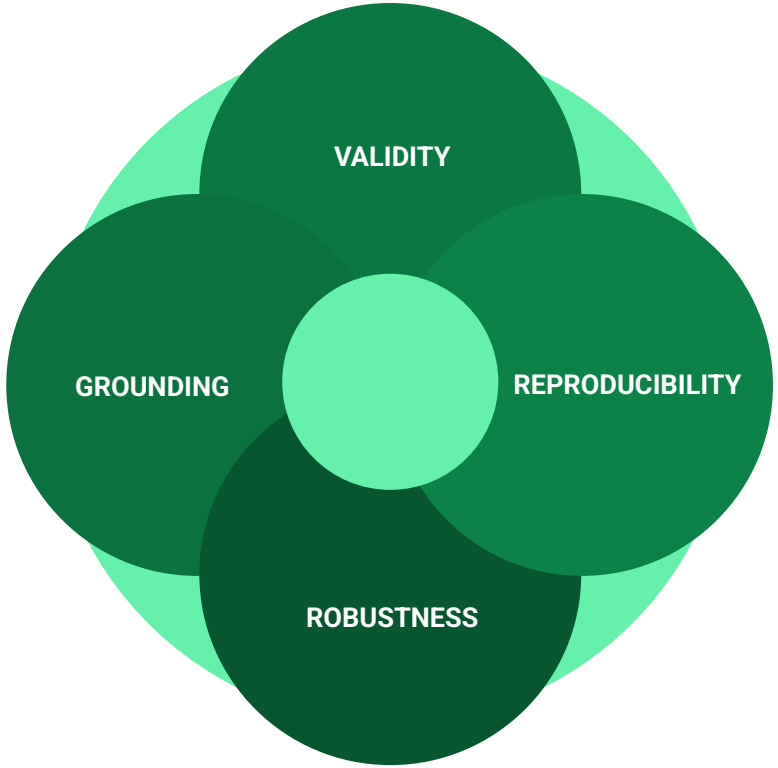
Vanessa Lama, Jazlyn Ilamni, Tony Ramirez

Analytics and AI Methods at Scale (AAIMS), NCCS, ORNL

U.S. DEPARTMENT of ENERGY

ORNL IS MANAGED BY UT-BATTELLE LLC
FOR THE US DEPARTMENT OF ENERGY

Why are we here today?

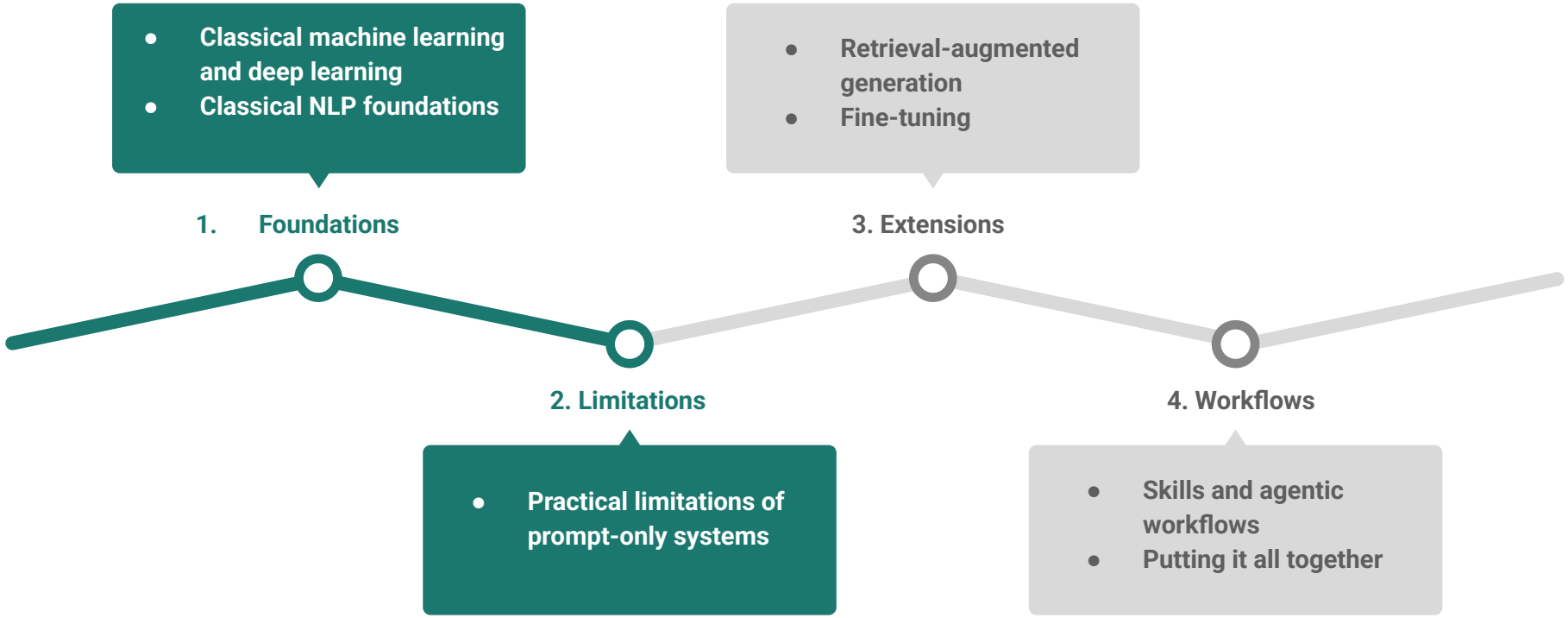


As GenAI is now moving from demos to real workflows,

the questions is no longer “what is an LLM?” but “how do we use one well?

Roadmap

- ❑ *where do these methods come from?*
- ❑ *what are the core limitations?*
- ❑ *what solutions extend LLM capability?*
- ❑ *how they become practical systems?*



Artificial Intelligence VS Machine Learning VS Deep Learning

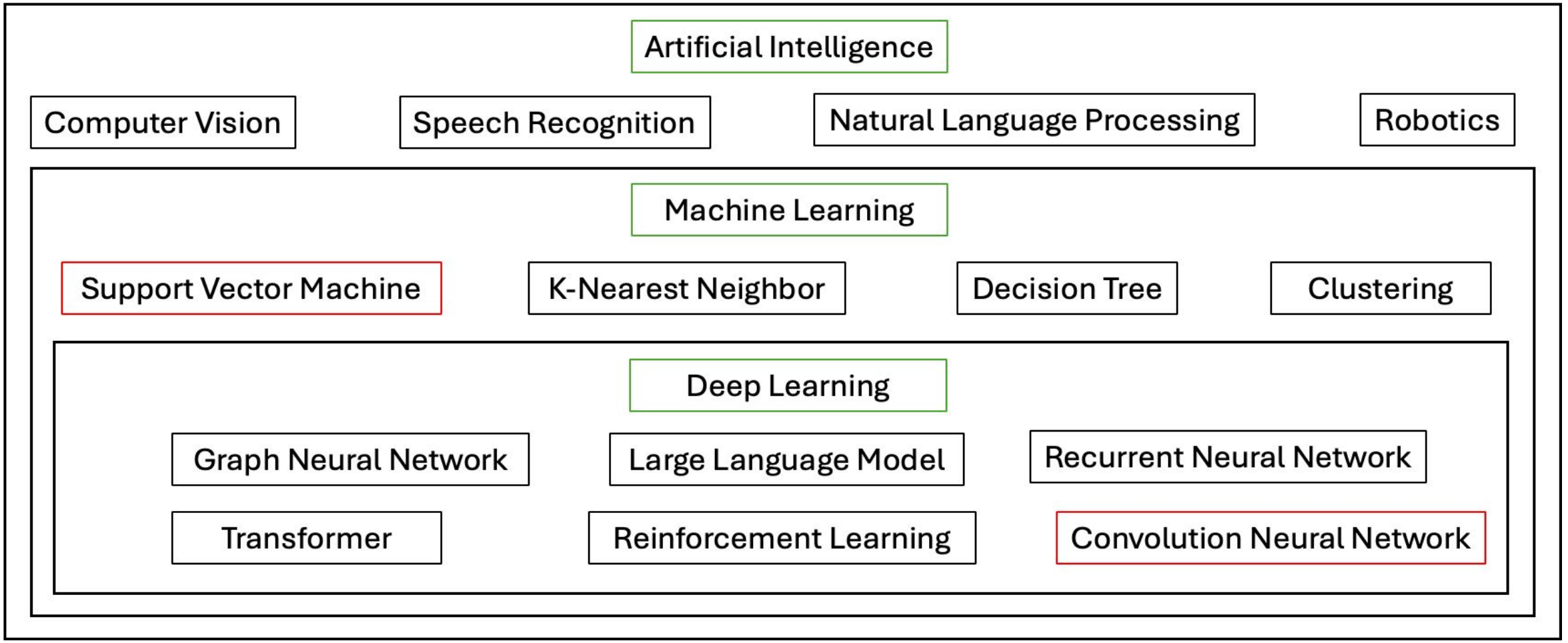
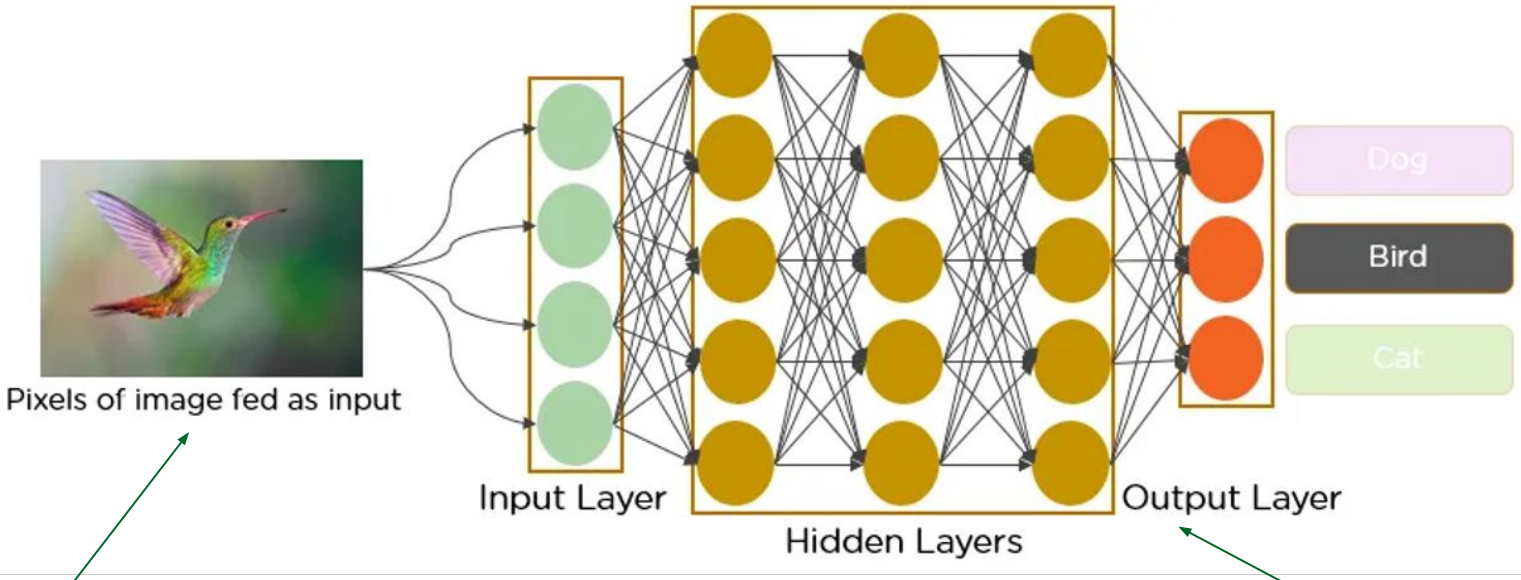
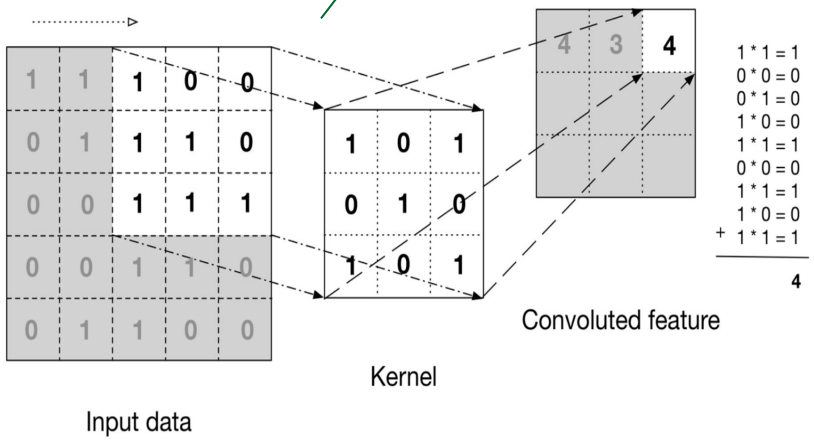


Image Classification



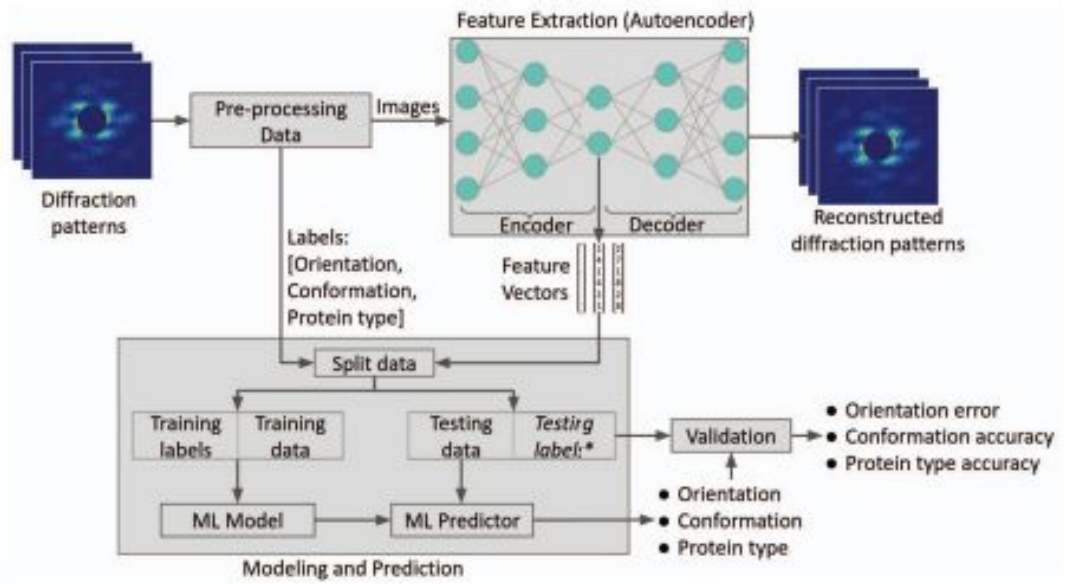


Feature Extraction

```
class ConvNet(nn.Module):
    def __init__(self):
        super(ConvNet, self).__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)
```

Model Architecture Design

What Science did we achieve?

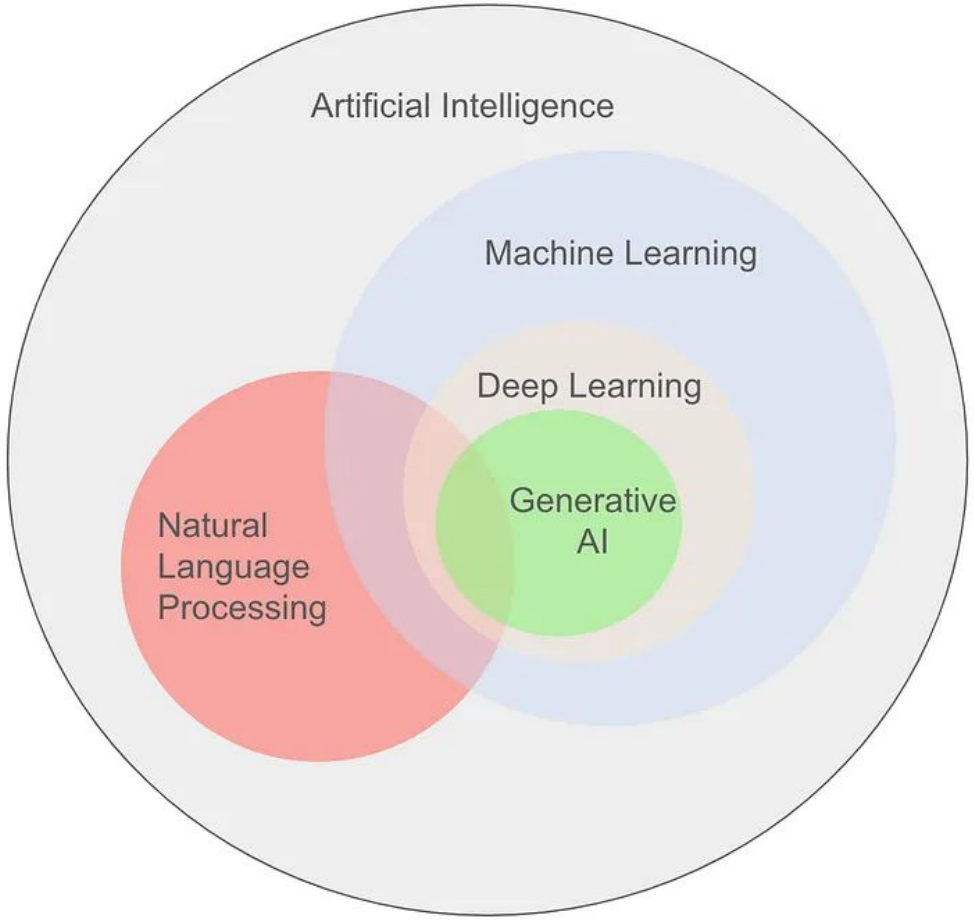


Identifying critical protein structure properties for times like covid-19!

"Identifying Structural Properties of Proteins from X-ray Free Electron Laser Diffraction Patterns" - https://ieeexplore.ieee.org/abstract/document/9973459

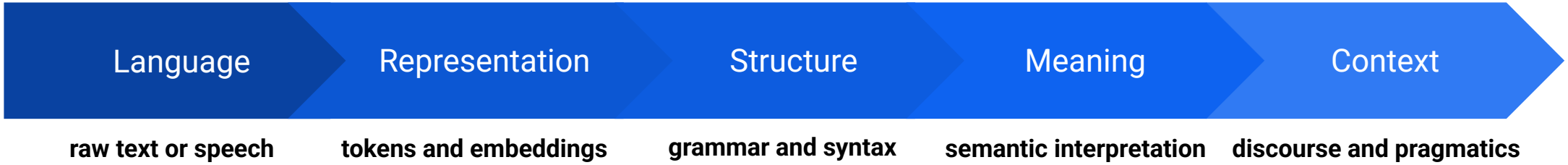
The emergence of Generative AI

Generative AI emerged from the convergence of advances in AI, machine learning, deep learning, and natural language processing.



https://medium.com/analytics-vidhya/what-is-generative-ai-9c34cfa0fd6b

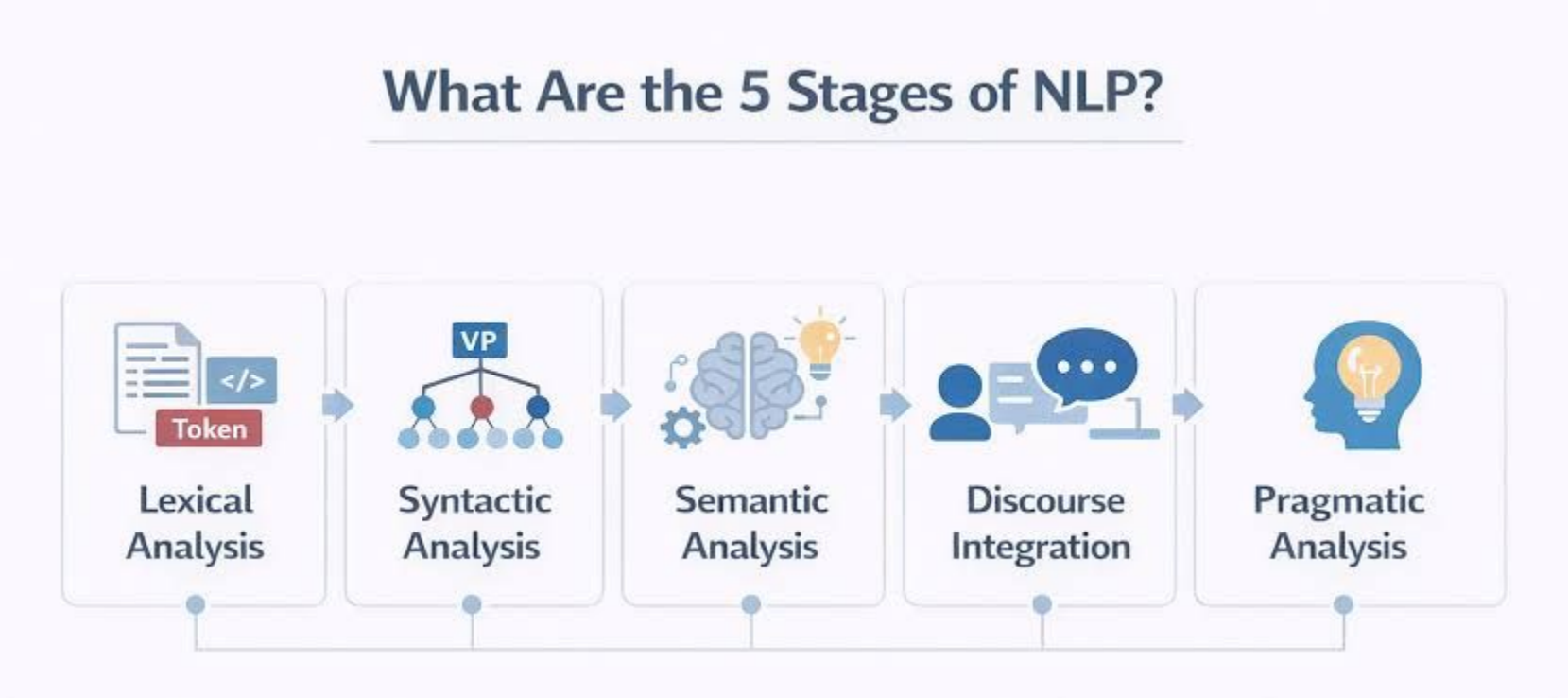
Core Natural Language Processing (NLP) Concepts



Language is transformed into computational representations that allow systems to analyze structure, infer meaning, and interpret context.

Background concepts adapted from Jurafsky and Martin, Speech and Language Processing.

What is Natural Language Processing (NLP)?



https://www.upgrad.com/blog/what-are-the-5-stages-of-nlp/

What is Natural Language Processing (NLP)?

Lexical Analysis	"Cats are running." → "Cats", "are", "running"
Syntactic Analysis	Identifies "The dog chased the cat" as Subject → Verb → Object
Semantic Analysis	Understands that "bank" could mean a financial institution or the side of a river depending on the sentence.
Discourse Integration	"Sarah bought a new laptop. She loves it." → Recognizes that "She" refers to Sarah and "it" refers to the laptop.
Pragmatic Analysis	"Can you open the window?" is understood as a polite request, not a question about ability.

https://www.upgrad.com/blog/what-are-the-5-stages-of-nlp/

Attention Is All You Need!

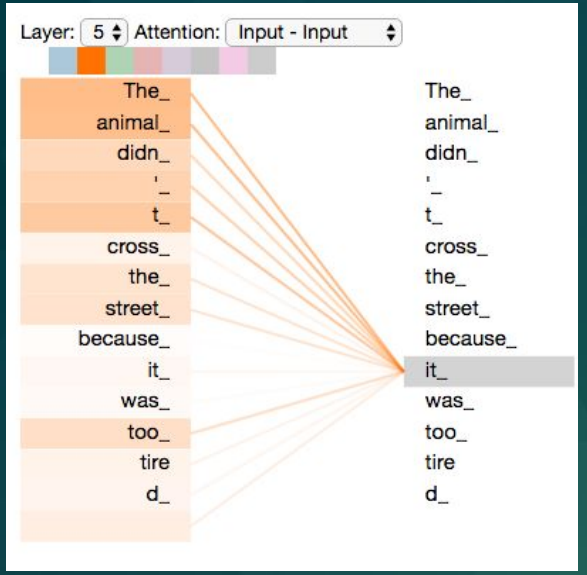
Token: a unit of text processed by the model

Embedding: vector representation of a token

Self-attention: a mechanism that lets each token weigh other relevant tokens

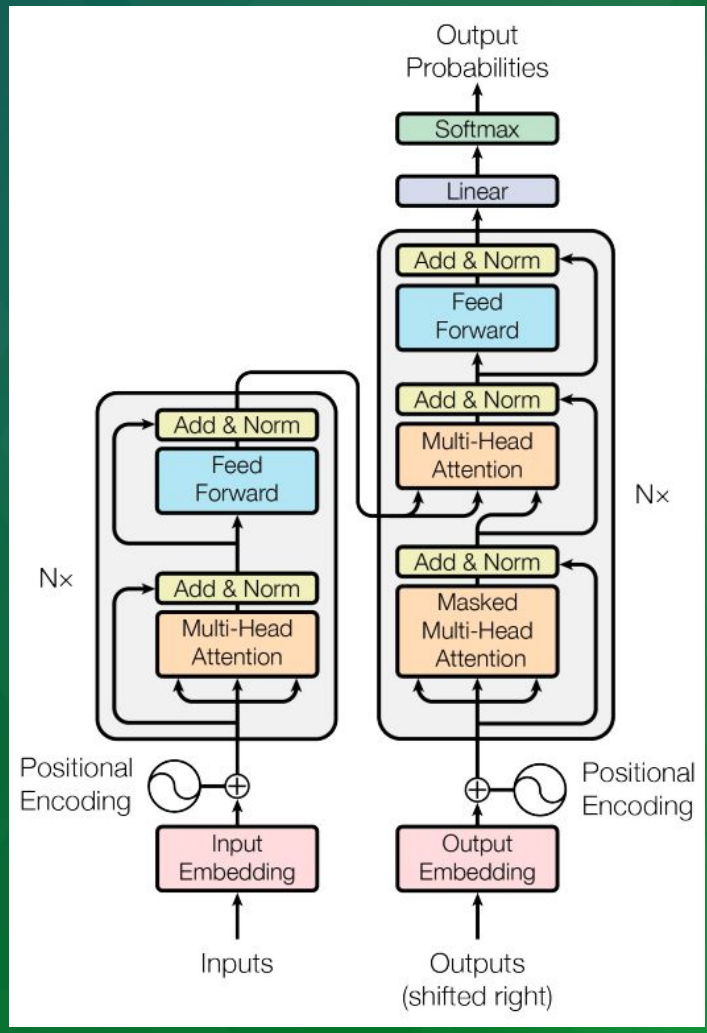
Transformer: a stacked architecture built around attention

Positional encoding: information about the position of each token in the sequence to the input embeddings



Self-attention in *"The animal didn't cross the street because it was too tired"*

When the model processes one word, attention lets it look at the other words and measure which ones matter most.



The transformer architecture organizes language processing through stacked attention-based layers, replacing strictly sequential recurrence with a mechanism that learns context-sensitive representations at scale.

Transformer architecture from Vaswani et al. (2017); self-attention illustration from Alammar (2018).

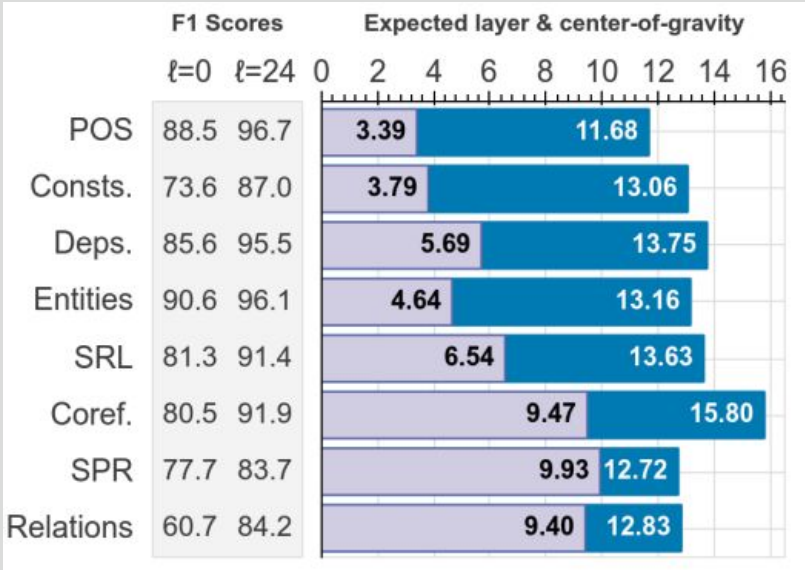
How Large Language Models (LLMs) Internalize Linguistic Structure

01	Lexical Analysis	tokenization and subword representations provide the basic units of input
02	Syntactic Analysis	attention patterns and layer structure help capture grammatical relationships
03	Semantic Analysis	contextual representations help words take meaning from surrounding text
04	Discourse Integration	attention across longer context helps connect references and ideas across sentences
05	Pragmatic Analysis	instruction tuning and dialogue context help shape intent-sensitive responses

ChatGPT

Results from Tenney et al. (2019).

"BERT Rediscovers the Classical NLP Pipeline" - Tenney et al. (2019).



POS = part-of-speech tagging
Consts. = constituent structure
Deps. = dependency relations
Entities = named entity recognition
SRL = semantic role labeling
Coref. = coreference resolution
SPR = semantic proto-roles
Relations = semantic relation classification

F1 scores show how much better each task can be recovered from the model after full contextual processing

High-level interpretation of the figure:

- *lower-level syntactic tasks like POS and constituency appear earlier*
- *higher-level semantic and discourse tasks like SRL and coreference appear later*
- *this resembles the progression of the classical NLP pipeline*

Limitations of Prompt-Only Systems

What LLMs do Well

✔ Fluent generation
coherent, natural text production

✔ Summarization
compression and reformulation of information

✔ Flexible language use
adaptation across tasks and prompts

✔ Generalization
broad capabilities from large-scale pretraining

What prompt-only use does not solve

✕ Grounding
fluent output may still be unsupported or hallucinatory

✕ Current/private knowledge
the model may not have access to local, private, or updated information

✕ Consistency
output quality may vary across prompts or reformulations

✕ Repeatability
prompting alone may not yield robust workflow behavior across repeated use

Conceptual limitations informed by Lewis et al. (2020), Ji et al. (2022), Bommasani et al. (2021), and Razavi et al. (2025).

Why Practical AI Systems Need More Than Prompting

Knowledge

- ➔ Current information
- ➔ Private or local documents
- ➔ Domain-specific context

Behaviour

- ➔ Consistent style and format
- ➔ Task-specific response patterns
- ➔ More controlled outputs

Process

- ➔ Repeatable multi-step workflows
- ➔ Tool use and external actions
- ➔ Structured task execution

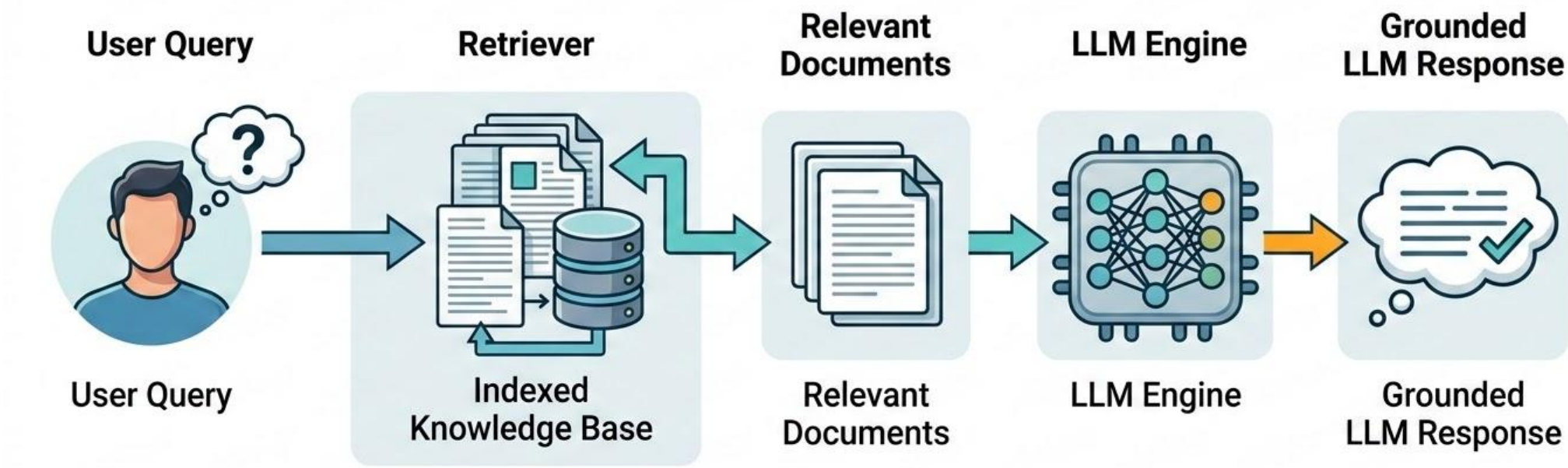
Retrieval Augmented Generation (RAG)
adds external knowledge at runtime

Fine-Tuning
shapes behavior through additional training

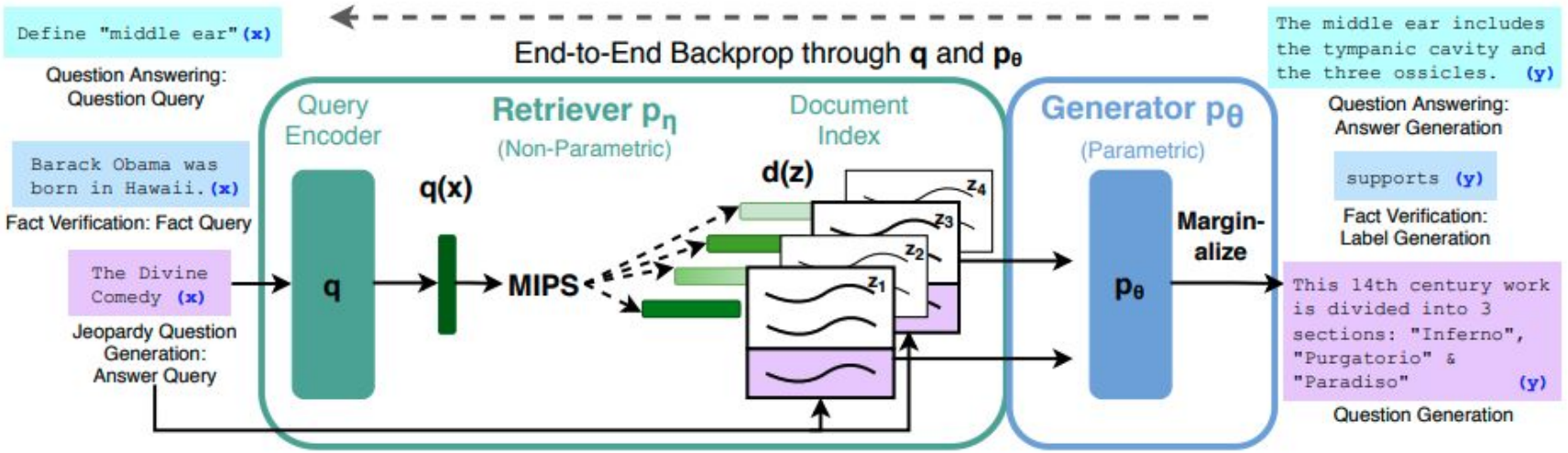
Skills
support structured, repeatable workflows

Retrieval Augmented Generation (RAG)

User Query → Retriever → Relevant Documents → Grounded LLM Response



Retrieval Augmented Generation (RAG)



***RAG** augments language generation with retrieved external evidence.*

Core probabilistic formulation:

$$p(y \mid x) = \sum_{z \in \text{top-}k} p_\eta(z \mid x)\, p_\theta(y \mid x, z)$$

x: input query
z: retrieved document or passage
y: generated output
p_\eta(z \mid x): retriever relevance distribution
p_\theta(y \mid x, z): generator conditioned on query and retrieved context

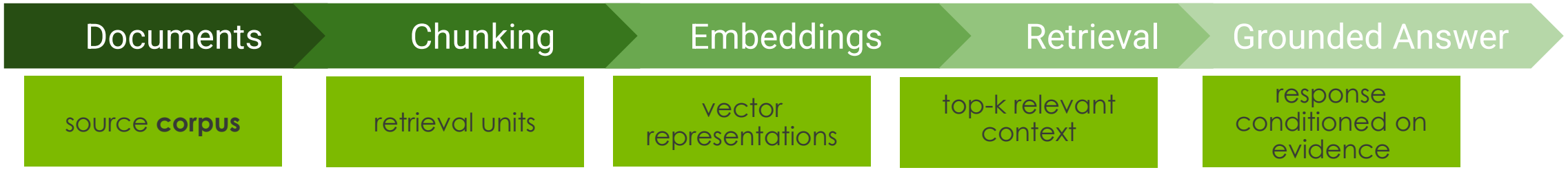
Interpretation:

RAG treats retrieved evidence as a latent variable and generates by marginalizing over top-ranked passages.

Figure and formulation from Lewis et al. (2020).

RAG Pipeline in This Tutorial

This tutorial operationalizes RAG as a document-processing and retrieval pipeline.



Venv Set Up

Mac

Create the venv:

- python -m venv .venv

Activate the venv:

- Mac:
  - source .venv/bin/activate

Windows

Create the venv:

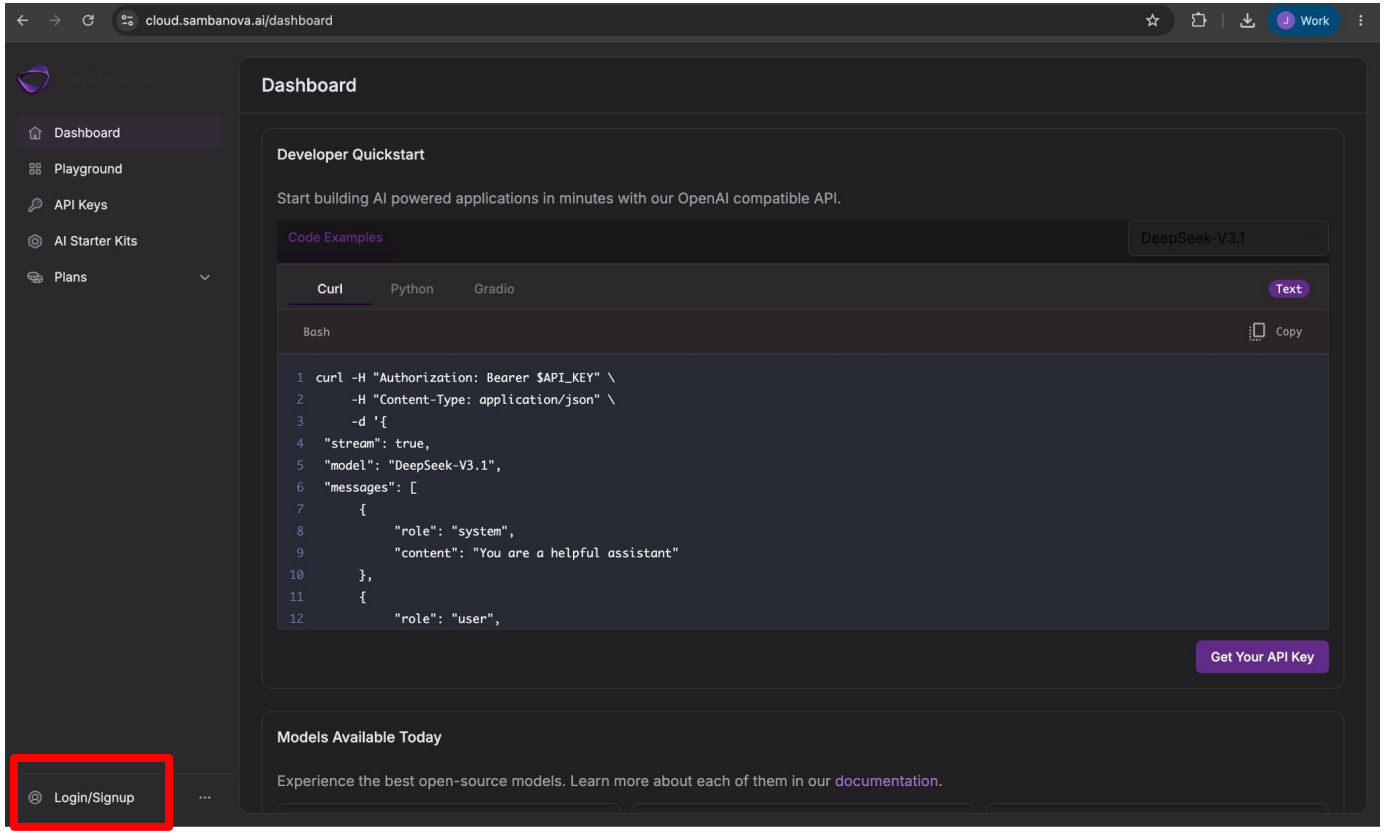
- python -m venv .venv

Activate the venv:

- Windows (CMD)
  - .venv\Scripts\activate.bat
- Windows (Powershell)
  - .venv\Scripts\Activate.ps1

SambaNova Set Up

- Navigate to the SambaNova Dashboard: https://cloud.sambanova.ai/dashboard
  - May need to clear cache if it does not load
- In the bottom left corner, click “Login/SignUp”
- Sign Up using XCAMS email address
- Verify your email address and log in

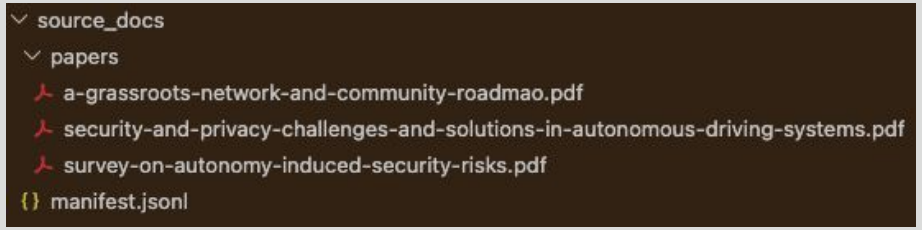


Document Chunking in Practice

Documents → **Chunking**

source **corpus**

- *"A Grassroots Network and Community Roadmap for Interconnected Autonomous Science Laboratories for Accelerated Discovery"*
- *"Security and Privacy Challenges and Solutions in Autonomous Driving Systems: A Comprehensive Review"*
- *"A Survey on Autonomy-Induced Security Risks in Large Model-Based Agents"*



- *python3 main.py --step chunk*
- *output:*

retrieval units

```python
def split_into_units(text: str) -> list[str]:
    paragraphs = [p.strip() for p in re.split(r"\n\s*\n+", text) if p.strip()]
    units: list[str] = []

    for paragraph in paragraphs:
        paragraph = re.sub(r"[ \t]+", " ", paragraph).strip()
        if len(paragraph) <= 800:
            units.append(paragraph)
            continue

        sentences = re.split(r"(?<=[.!?])\s+(?=[A-Z0-9\"'])", paragraph)
```

Initial paragraph of normalized text; Long paragraphs are further divided into sentence-level units

```python
def split_into_chunks(text: str, chunk_size: int, chunk_overlap: int) -> list[str]:
```

Core chunking function for retrieval-ready text segmentation

```python
while end < len(bounded_units):
    unit = bounded_units[end]
    sep = 2 if current_len > 0 else 0
    candidate_len = current_len + sep + len(unit)
    if current_len > 0 and candidate_len > chunk_size:
        break
    current_len = candidate_len
    end += 1
```

Units are packed into chunks up to the target size

```python
overlap_chars = 0
new_start = end
while new_start > prev_start:
    unit = bounded_units[new_start - 1]
    sep = 2 if new_start < end else 0
    next_overlap = overlap_chars + sep + len(unit)
    if next_overlap > chunk_overlap and new_start < end:
        break
    overlap_chars = next_overlap
    new_start -= 1

start = new_start if new_start > prev_start else end
```

Adjacent chunks retain overlapping context for retrieval continuity

Embeddings and Retrieval in Practice

Embeddings

vector representations

embedder.encode(...) Chunk text is mapped into dense vector representations

vectors.tolist()
Embeddings are converted into a storable numeric form

collection.upsert(... embeddings=embeddings)
Chunk vectors are indexed for later similarity search

```python
def flush_batch() -> None:
    nonlocal total
    if not ids:
        return
    batch_count = len(ids)
    vectors = embedder.encode(docs, convert_to_numpy=True)
    embeddings = vectors.tolist()
    collection.upsert(
        ids=ids,
        documents=docs,
        metadatas=metas,
        embeddings=embeddings,
    )
```

Embedding function:

$$e_i = f_\phi(c_i)$$

where:

- *c_i is a chunk*
- *f_\phi is the embedding model*
- *e_i is the dense vector representation*

- *python3 main.py --step embed*
- *output:*

Retrieval

top-k relevant context

```python
def retrieve_context(query_text: str, embedder, collection, top_k: int):
    try: …
    except Exception:  # pragma: no cover…

    query_vector = embedder.encode([query_text], convert_to_numpy=True).tolist()[0]
    try:
        result = collection.query(
            query_embeddings=[query_vector],
            n_results=top_k,
            include=["documents", "metadatas", "distances"],
        )
    except InvalidArgumentError as exc:
        raise RuntimeError(
            f"Embedding dimension mismatch for collection '{collection.name}'. "
            f"The current query model produced dimension {len(query_vector)}. "
            "Choose the matching --chroma-collection / "
            "--embedding-model pair or regenerate embeddings."
        ) from exc

    docs = result.get("documents", [[]])[0]
    metas = result.get("metadatas", [[]])[0]
    dists = result.get("distances", [[]])[0]
    return docs, metas, dists
```

query_vector = ...
The user query is embedded in the same vector space as the chunks

collection.query(...)
Nearest-neighbor search retrieves the top-k most relevant chunks

documents, metadatas, distances
The system returns both evidence text and similarity information

Embedding function:

$$p_\eta(z \mid x) \propto \exp\left(\text{sim}(q, d(z))\right)$$

where,

- *x: input query in natural language*
- *z: candidate retrieved document or chunk*
- *pη(z | x): retriever relevance distribution over documents given the query*
- *η: retriever parameters*
- *q: dense vector representation of the query*
- *d(z): dense vector representation of document or chunk z*
- *sim(q, d(z)): similarity score between the query and document embeddings*
- *exp(.): exponentiation that converts similarity scores into positive retrieval weights*

- *python3 main.py --step query --query "What security risks are discussed?"*

Grounded Answer Generation

Grounded Answer

response conditioned on evidence

```python
def build_grounded_user_message(
    query_text: str,
    source_guide: str,
    context_block: str,
) -> str:
    return (
        f"User question:\n{query_text}\n\n"
        "Answer the question using the retrieved context below. "
        "Cite supporting evidence inline with the chunk label and source.\n\n"
        f"Source guide:\n{source_guide}\n\n"
        f"Retrieved context:\n{context_block}"
    )
```

build_grounded_user_message(...)
Assembles a retrieval-grounded prompt that conditions generation on both the query and retrieved evidence.

```python
def request_grounded_answer(
    client,
    model_name: str,
    messages: list[dict[str, str]],
) -> str:
    response = client.chat.completions.create(
        model=model_name,
        messages=messages,
        temperature=0.1,
        top_p=0.1,
    )
    return response.choices[0].message.content or ""
```

request_grounded_answer(...)
Invokes the generative model on the grounded prompt to produce the final response.

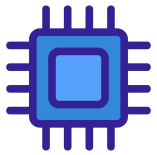
- *python3 main.py --step query --query "What security risks are discussed?"*

```
[3/3] Starting retrieval and grounded chat pipeline
[3/3] -> Models used in this step: embedding=google/embeddinggemma-300m, chat=gpt-oss-120b
[3/3] -> Requested collection: document_chunks__google_embeddinggemma_300m
[3/3] -> Loading query embedder
Loading weights: 100%|██████████| 314/314 [00:00<00:00, 5974.01it/s]
[3/3] -> Connecting to Chroma collection
[3/3] -> Active collection: document_chunks__google_embeddinggemma_300m
Chroma chat is ready. Type 'exit' or 'quit' to leave.
Collection: document_chunks__google_embeddinggemma_300m
Agent model: gpt-oss-120b
[3/3] -> Retrieving top 5 chunks for query
[3/3] -> Building grounded prompt from retrieved context
[3/3] -> Requesting grounded answer from SambaNova

Assistant: The paper identifies a broad set of security risks that arise as large-language-model (LLM)-based agents become more autonomous:

* **Prompt-related attacks** – the underlying LLM can be fooled by adversarial prompts and can hallucinate incorrect information, while standalone LLMs are vulnerable to prompt-injection attacks [Chunk 1][Chunk 2].

* **Memory attacks** – an agent's memory stores can be poisoned, manipulated, or exfiltrated, allowing an attacker to corrupt the agent's knowledge or steal data [Chunk 1].

* **Tool-use and execution pathways** – interfaces that let the agent invoke external tools or APIs create execution paths for unsafe behaviours, including privilege-escalation and misuse of external services [Chunk 1][Chunk 4].

* **Planning and action risks** – planning modules may generate brittle or misaligned action sequences, and agents can issue irreversible or unsafe commands in real environments [Chunk 1][Chunk 4].

* **Interaction with untrusted environments** – agents that consume unverified web content or other untrusted user inputs face additional hazards that break traditional safety assumptions [Chunk 1].

* **Multi-agent coordination** – when multiple agents cooperate, emergent dynamics can amplify failures or adversarial behaviour [Chunk 4].

* **Data-privacy and leakage** – agents may expose training-data-derived information, leak memorized sensitive data during generation, and increase exposure risk when they use real-time APIs or tool invocation [Chunk 4].

These risks are organized in the survey across six security dimensions (autonomy level, learning dynamics, goal formation, external impact, resource access, and alignment predictability) and are illustrated in the risk matrix (Table 4) that grades each threat from low to high severity [Chunk 2][Chunk 4].
```


https://github.com/vanessalama09/agentic-ai-training

Fine-Tuning

Pre-trained Model	Task / Domain Data	Adapted Model
general language knowledge learned during large-scale pretraining	examples that specify desired behavior, style, or task format	a fine-tuned model specialized for more consistent downstream performance



Conceptual overview informed by Wei et al. (2021) and Hu et al. (2021).

RAG-Assisted Data Creation

Seed Topics → Retrieval → Context-Grounded Example Generation → JSONL Training Data

Seed topics define the desired training distribution

Retrieval supplies domain-relevant evidence from the corpus

The generator produces chat-format examples conditioned on that evidence

The resulting JSONL dataset is used for fine-tuning

finetuning > data_creation > {} seed_topics.jsonl

```
1  {"topic":"autonomy-induced security risks","qu
2  {"topic":"autonomous driving security","questi
3  {"topic":"agent safeguards","question_style":"
4  {"topic":"privacy in autonomous systems","ques
5  {"topic":"comparing autonomy domains","questio
6  {"topic":"autonomous science labs","question_s
```

```python
def build_generation_prompt(seed: dict, source_guide: str, context_block: str) -> str:
    topic = seed.get("topic", "autonomous systems security")
    question_style = seed.get("question_style", "beginner explanation")
    retrieval_query = seed.get("retrieval_query", topic)

    return (
        f"Topic: {topic}\n"
        f"Question style: {question_style}\n"
        f"Retrieval query: {retrieval_query}\n\n"
        "Create one user question and one assistant answer suitable for a "
        "fine-tuning dataset.\n\n"
        f"Source guide:\n{source_guide}\n\n"
        f"Retrieved context:\n{context_block}"
    )
```

- *python3 main.py --step query --query "What safety risks arise in autonomous systems?" --show-sources*

```
Retrieved context:

[1] distance=0.806405
source=papers/survey-on-autonomy-induced-security-risks.pdf chunk_index=51
Their fail- ure modes evolve from local and stateless glitches into persistent, recursive,
ns the attack surface and compounds the failure trajectory.  Defenses must therefore trans

[2] distance=0.820535
source=papers/survey-on-autonomy-induced-security-risks.pdf chunk_index=20
This dual-reward mechanism allows the agent to arbitrate between autonomous action and huma
collectively support a model of en- dogenous security , where safety and alignment emerge

[3] distance=0.825379
source=papers/survey-on-autonomy-induced-security-risks.pdf chunk_index=24
By embedding reflective reasoning, risk-sensitive plan- ning, and value-aligned regulation
tonomously, but also of an- ticipating failure, adapting responsibly, and aligning their be

[4] distance=0.829842
source=papers/security-and-privacy-challenges-and-solutions-in-autonomous-driving-systems.
1.  While foundational to achieving autonomous capabilities, this escalating complexity si
 vehicles evolve from isolated mechanical  devices into highly connected nodes within a br

[5] distance=0.841397
source=papers/survey-on-autonomy-induced-security-risks.pdf chunk_index=88
Future agent architectures must encode such constraints as first-class elements, enforced
```

- *python3 finetuning/data_creation/generate_examples.py --examples-per-seed 5*

finetuning > {} generated_train.jsonl

```
1  ;."}, {"role": "user", "content": "What are autonomy-induced security risks in mod
2  ;."}, {"role": "user", "content": "Can you give a simple definition of what autono
3  ;."}, {"role": "user", "content": "What are autonomy-induced security risks in mod
4  ;."}, {"role": "user", "content": "Can you explain in simple terms what autonomy-i
5  ;."}, {"role": "user", "content": "Can you explain what autonomy-induced security
6  ;."}, {"role": "user", "content": "What are the main security and privacy risks th
7  ;."}, {"role": "user", "content": "What are the major security and privacy risks t
8  ;."}, {"role": "user", "content": "Can you give me a high-level overview of the ma
9  ;."}, {"role": "user", "content": "Can you give me a high-level overview of the ma
10 ;."}, {"role": "user", "content": "Can you give me an overview of the main securit
11 ;."}, {"role": "user", "content": "What practical mitigations can we apply to redu
12 ;."}, {"role": "user", "content": "I'm designing an autonomous agent that uses a l
13 ;."}, {"role": "user", "content": "I'm designing an autonomous LLM-based agent tha
14 ;."}, {"role": "user", "content": "I'm building an autonomous LLM-based agent that
15 ;."}, {"role": "user", "content": "What practical steps can we take to reduce the
16 ;."}, {"role": "user", "content": "Can you explain the main privacy concerns that
17 ;."}, {"role": "user", "content": "Can you explain the key privacy challenges that
18 ;."}, {"role": "user", "content": "Can you explain the main privacy concerns that
```

Generated examples are aggregated into a JSONL corpus for downstream model adaptation.

Fine-Tuning in Practice

Training Data → Fine-Tuning → Adapted Model

- *python3 finetuning/train_model.py*

```python
def format_chat_example(messages: list[dict[str, str]]) -> str:
    parts = []
    for message in messages:
        role = message.get("role", "user").strip().lower()
        content = message.get("content", "").strip()
        if not content:
            continue
        parts.append(f"{role}: {content}")
    return "\n".join(parts)
```

format_chat_example(...)
Chat-format messages are serialized into a training sequence

```python
tokenized = tokenizer(
    text,
    truncation=True,
    max_length=max_length,
    padding="max_length",
    return_tensors="pt",
)
```

tokenizer(...)
Examples are tokenized and truncated to a fixed sequence length

```python
self.examples.append(
    {
        "input_ids": tokenized["input_ids"].squeeze(0),
        "attention_mask": tokenized["attention_mask"].squeeze(0),
    }
)
```

Tokenized examples are stored as tensors for model training

```python
tokenizer = AutoTokenizer.from_pretrained(args.model_name)
model = AutoModelForCausalLM.from_pretrained(args.model_name)
dataset = ChatFineTuneDataset(train_file, tokenizer, args.max_
collator = DataCollatorForLanguageModeling(tokenizer=tokenize
```

A pre-trained causal language model is loaded as the fine-tuning starting point

```python
training_args = TrainingArguments(
    output_dir=str(output_dir),
    overwrite_output_dir=True,
    num_train_epochs=args.epochs,
    per_device_train_batch_size=args.batch_size,
    learning_rate=args.learning_rate,
    save_strategy="epoch",
    logging_steps=1,
    report_to="none",
    remove_unused_columns=False,
)
```

Optimization hyperparameters define the supervised training configuration

```python
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=dataset,
    data_collator=collator,
)

trainer.train()
trainer.save_model(str(output_dir))
tokenizer.save_pretrained(str(output_dir))
print(f"[ft-train] Saved fine-tuned model to {output_dir}")
```

The Hugging Face training loop coordinates model, data, and optimization

The model parameters are updated on the generated training dataset

The adapted model is saved for downstream use

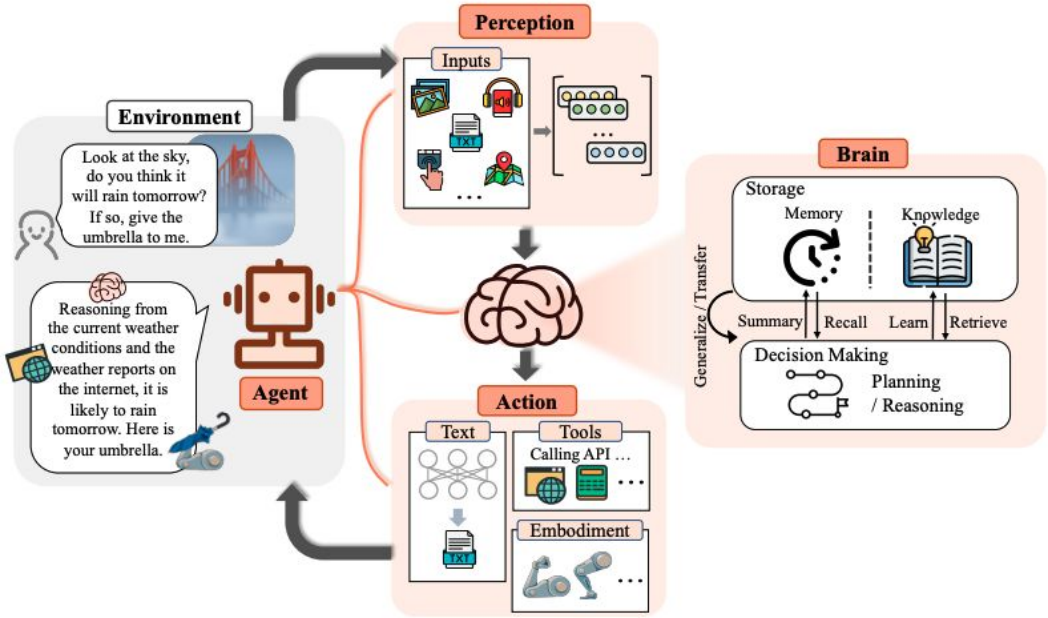
From Single Responses to Agentic Workflows

➔ ***What is an Agent? How is that different from a model, a skill, or a workflow?***

LLM
the reasoning and generation engine

Agent
the system that uses the model to pursue a task over multiple steps

Skill
a recipe/instruction; the reusable procedure or guidance the agent applies during execution



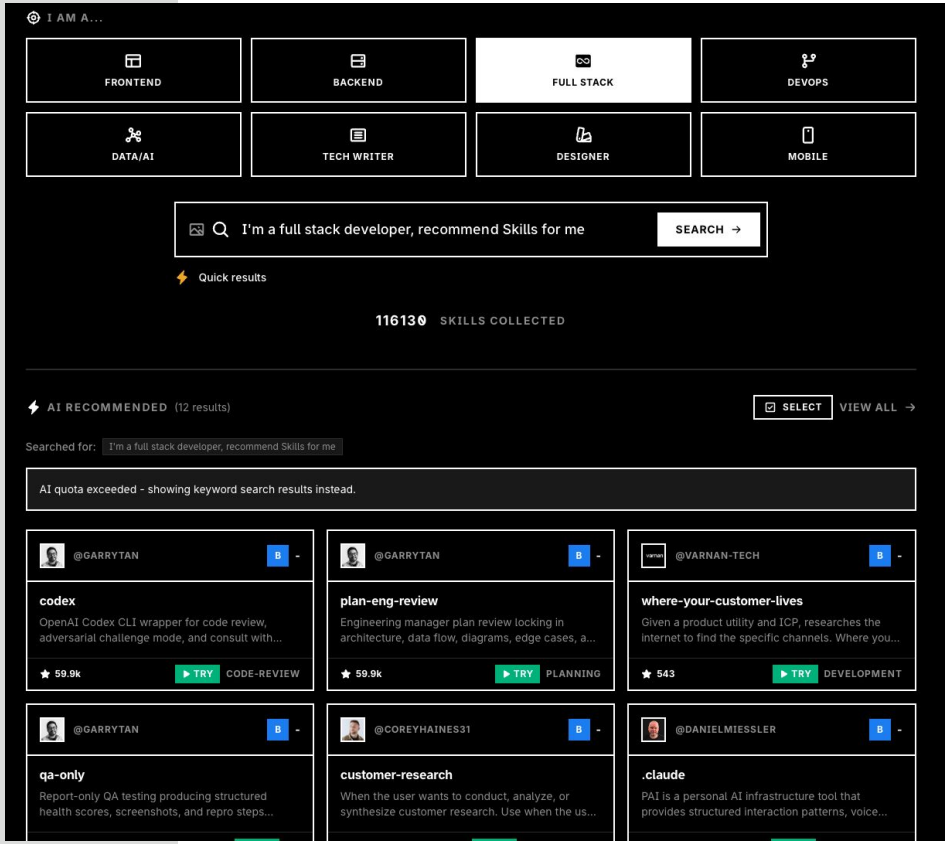
Unlike one-shot prompting, agents operate through iterative reasoning and action within an environment.

Single Response	Skill-Based Workflow
prompt -> model -> answer	prompt -> task decomposition -> retrieval/tools/memory -> intermediate steps -> final output

Figure from Xi et al. (2023).

From Single Responses to Agentic Workflows

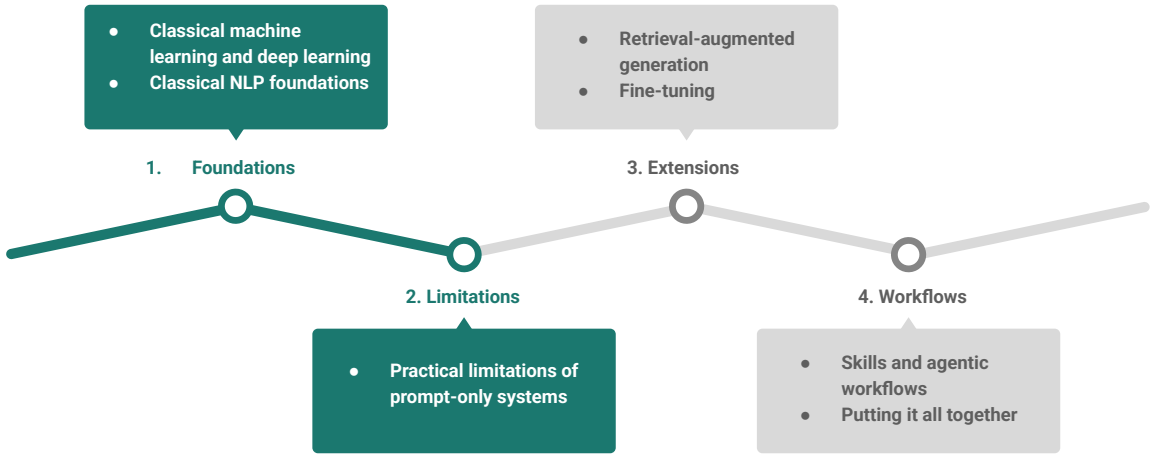
Single Response	Skill-Based Workflow
prompt -> model -> answer	prompt -> task decomposition -> retrieval/tools/memory -> intermediate steps -> final output



https://www.skillhub.club/

In conclusion

- *where do these methods come from?*
- *what are the core limitations?*
- *what solutions extend LLM capability?*
- *how they become practical systems?*

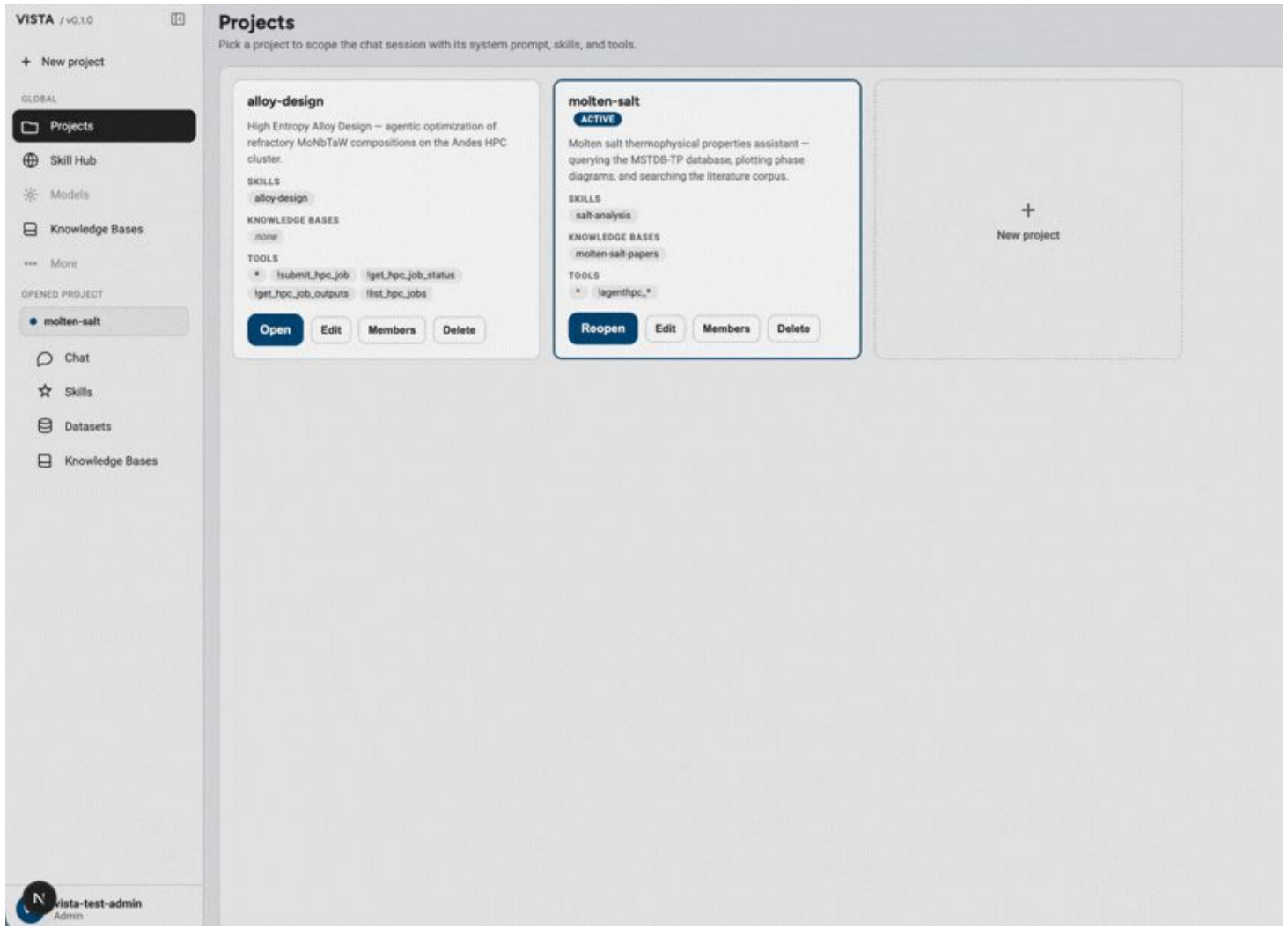


- *Foundations*
  modern LLMs internalize many classical NLP functions through learned representations
- *Limitations*
  prompt-only systems remain limited in grounding, consistency, and repeatability
- *Extensions*
  RAG, fine-tuning, and data creation workflows address different practical gaps
- *Agentic Workflows*
  skills and agents move AI systems from isolated responses to structured multi-step execution

References

- Vaswani, A., et al. (2017). Attention Is All You Need. arXiv:1706.03762. https://arxiv.org/abs/1706.03762
- Alammar, J. (2018). The Illustrated Transformer. https://jalammar.github.io/illustrated-transformer/
- Tenney, I., Das, D., & Pavlick, E. (2019). BERT Rediscovers the Classical NLP Pipeline. arXiv:1905.05950. https://arxiv.org/abs/1905.05950
- Lewis, P., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401. https://arxiv.org/abs/2005.11401
- Ji, Z., et al. (2022). Survey of Hallucination in Natural Language Generation. arXiv:2202.03629. https://arxiv.org/abs/2202.03629
- Bommasani, R., et al. (2021). On the Opportunities and Risks of Foundation Models. arXiv:2108.07258. https://arxiv.org/abs/2108.07258
- Razavi, M., et al. (2025). Benchmarking Prompt Sensitivity in Large Language Models. arXiv:2502.06065. https://arxiv.org/abs/2502.06065
- Wei, J., et al. (2021). Finetuned Language Models Are Zero-Shot Learners. arXiv:2109.01652. https://arxiv.org/abs/2109.01652
- Hu, E. J., et al. (2021). LoRA: Low-Rank Adaptation of Large Language Models. arXiv:2106.09685. https://arxiv.org/abs/2106.09685
- Xi, Z., et al. (2023). The Rise and Potential of Large Language Model Based Agents: A Survey. arXiv:2309.07864. https://arxiv.org/abs/2309.07864

Current Agentic Needs



Fully agentic tools with human-in-the-loop design is on demand!

https://genesis-vista.github.io/

Any Questions/Concerns/Comments?

Vanessa Lama, Jazlyn Ilamni, Tony Ramirez

EMAIL: lamav@ornl.gov
Analytics and AI Methods at Scale (AAIMS), NCCS, ORNL

U.S. DEPARTMENT of ENERGY

ORNL IS MANAGED BY UT-BATTELLE LLC
FOR THE US DEPARTMENT OF ENERGY