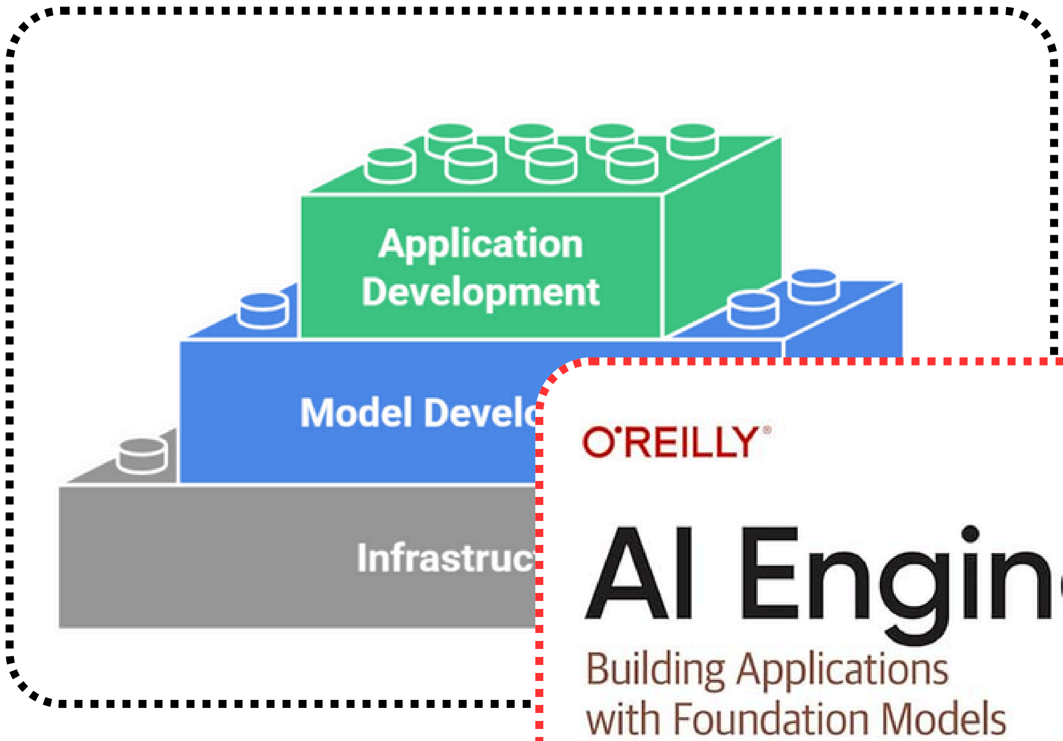
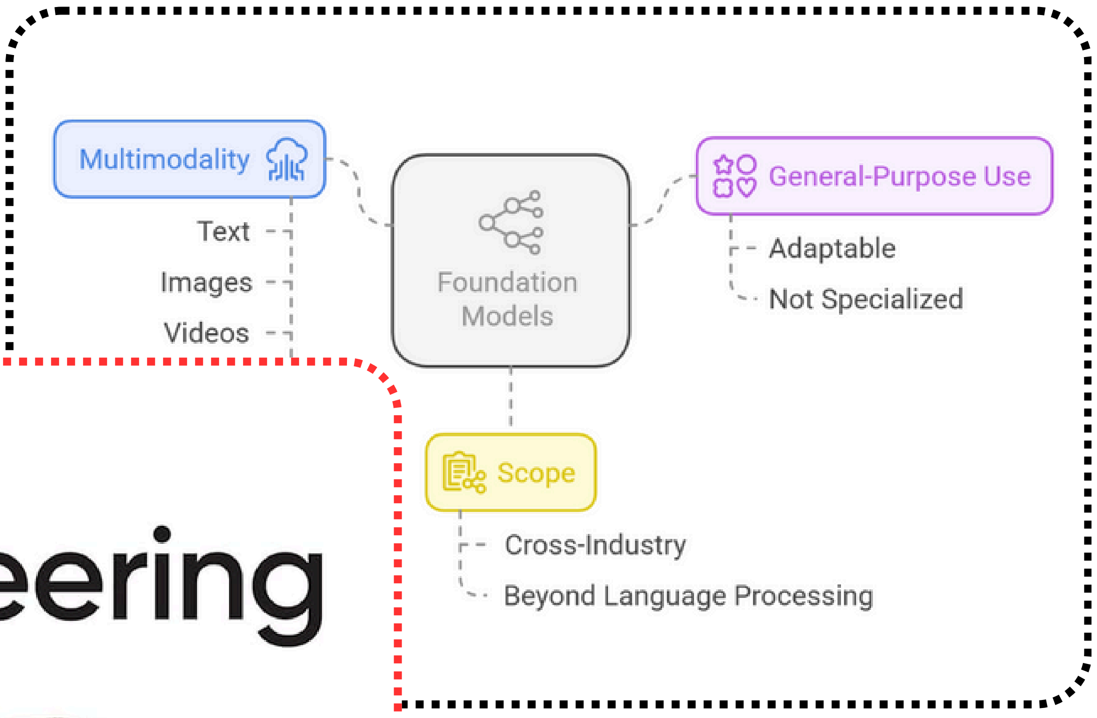
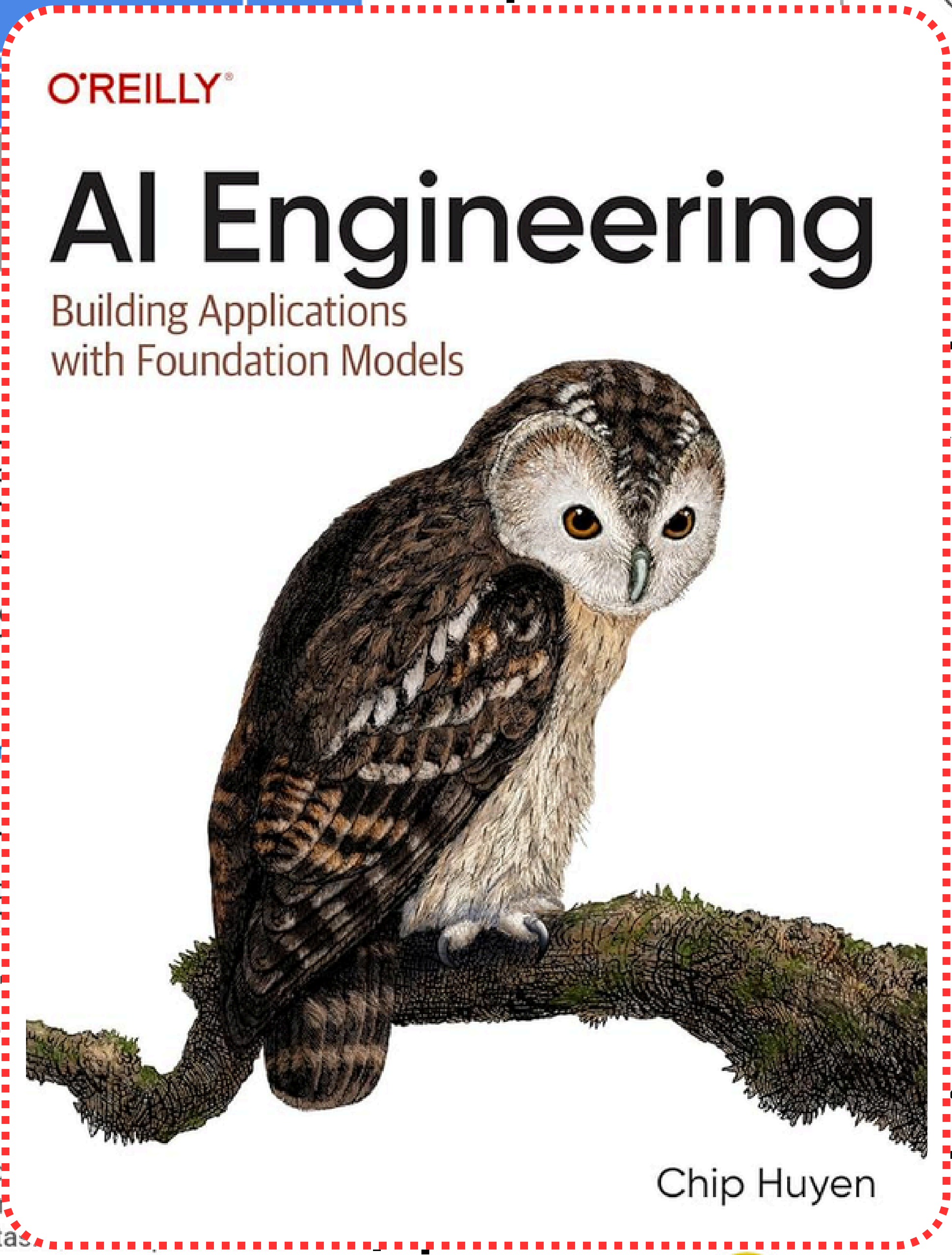
Excerpts from Chapter 1

Introduction to Building AI Applications with Foundation Models



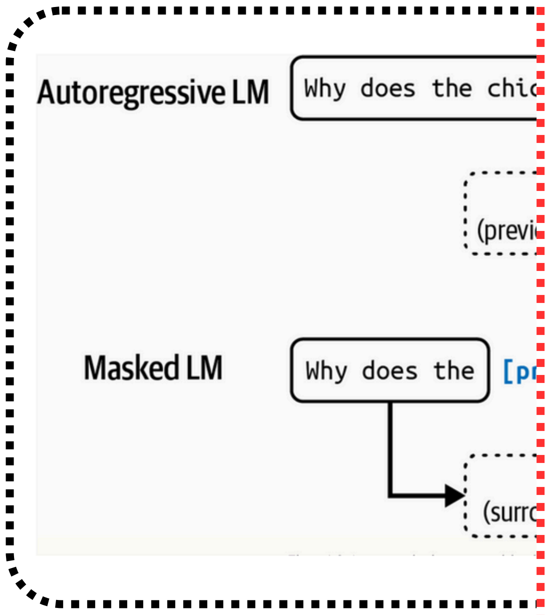


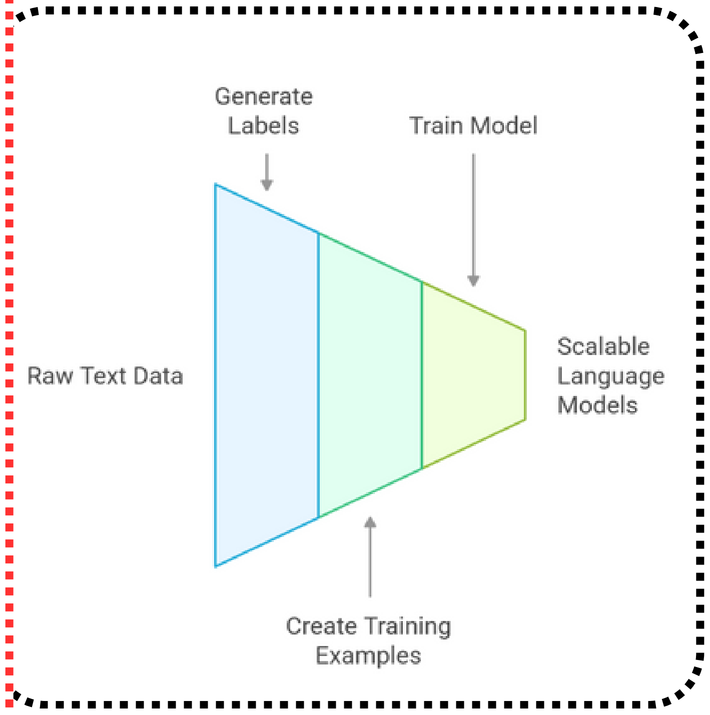

O'REILLY®

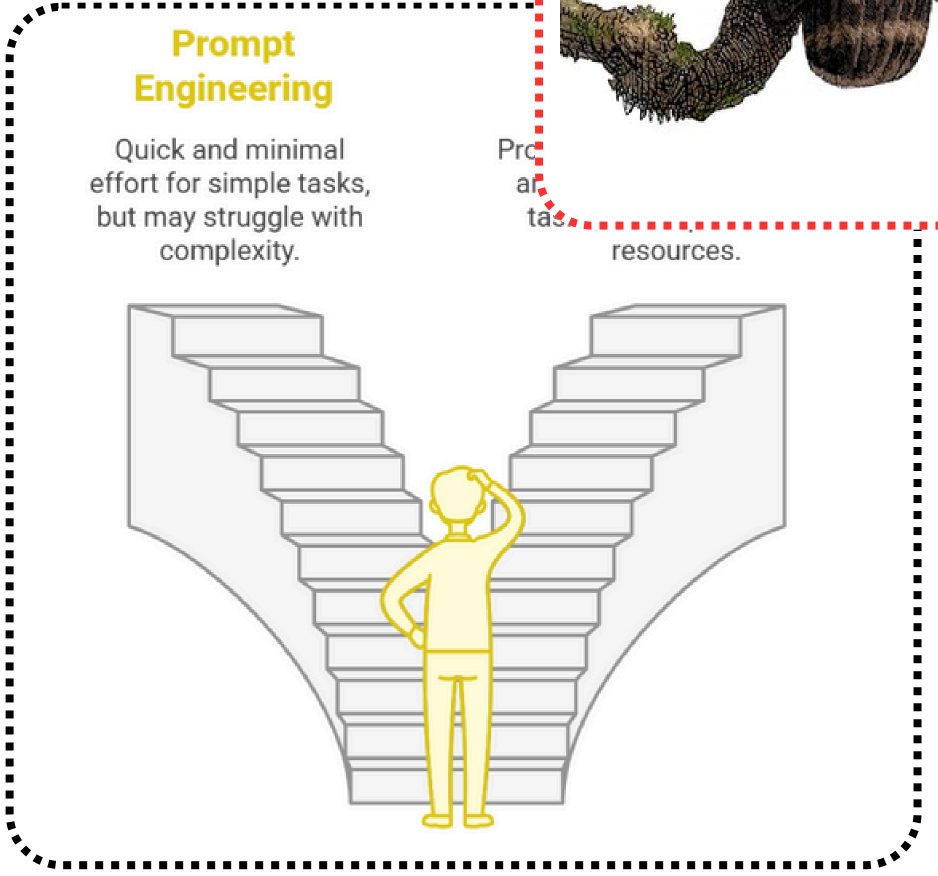
AI Engineering

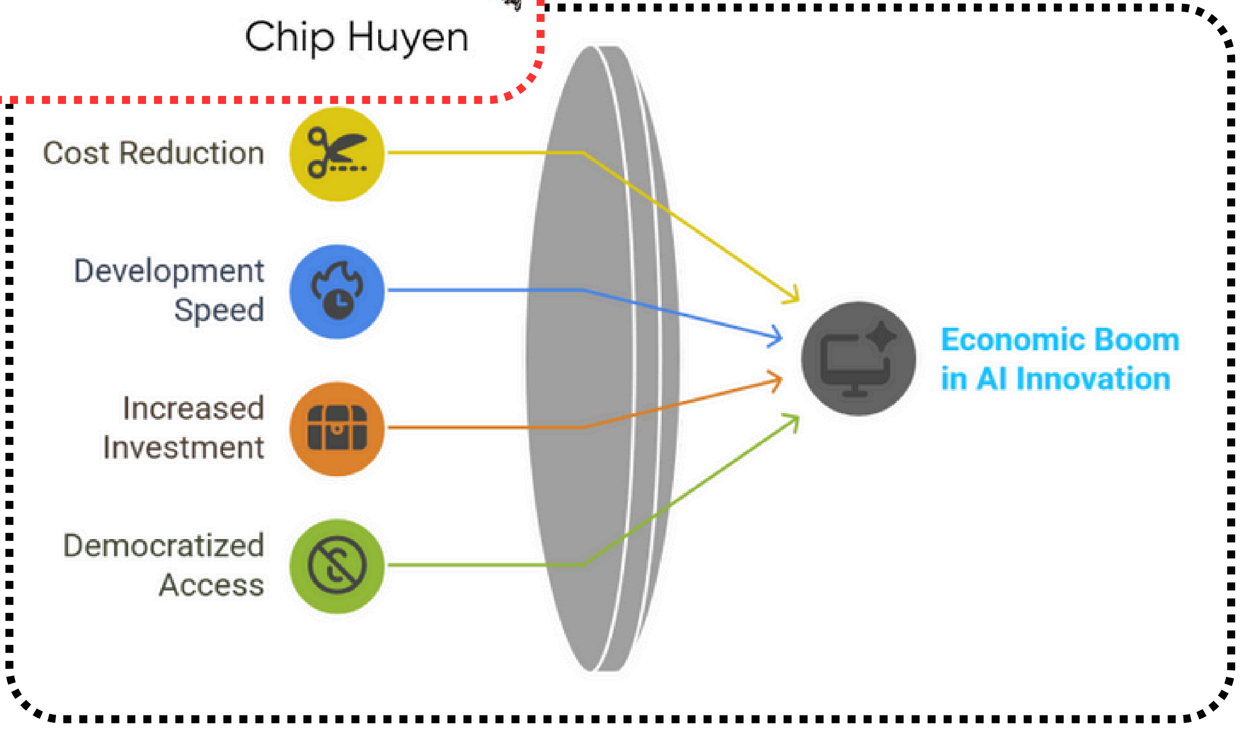
Building Applications with Foundation Models

Chip Huyen






Prompt Engineering
Quick and minimal effort for simple tasks, but may struggle with complexity.
Pro
a
tas
resources.



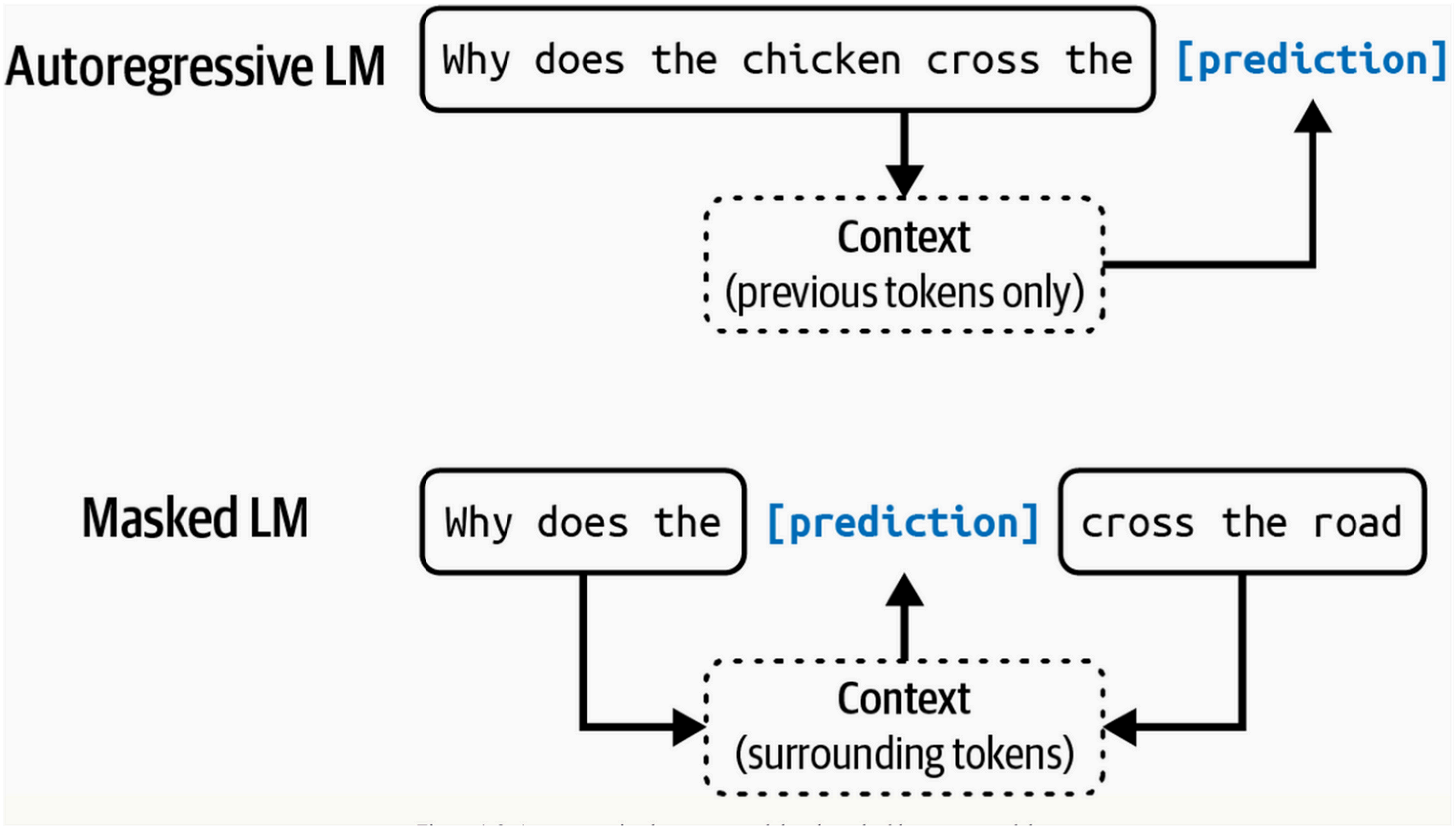
What are the two main types of language models mentioned, and how do they differ in their predictive approach?

Language models are computational systems designed to understand and generate human-like text, solving problems like text generation, translation, and classification by learning patterns in language data. The two main types mentioned are:

- **Masked Language Models (MLMs)**: These models predict missing words or tokens in a sentence by analyzing context from both before and after the gap. For example, in "My favorite __ is blue," an MLM might predict "color." They're useful for tasks like sentiment analysis or code debugging because they consider the full context. A prominent example is BERT.

- **Autoregressive Language Models**: These predict the next word in a sequence based only on the words that come before it. For instance, given "My favorite color is," the model might predict "blue." They excel at generating continuous text, making them popular for applications like chatbots or story generation.

The key difference is that MLMs use bidirectional context (before and after), while autoregressive models rely solely on prior context, shaping how they solve language-related problems.



What is self-supervision, and why is it significant for scaling language models?

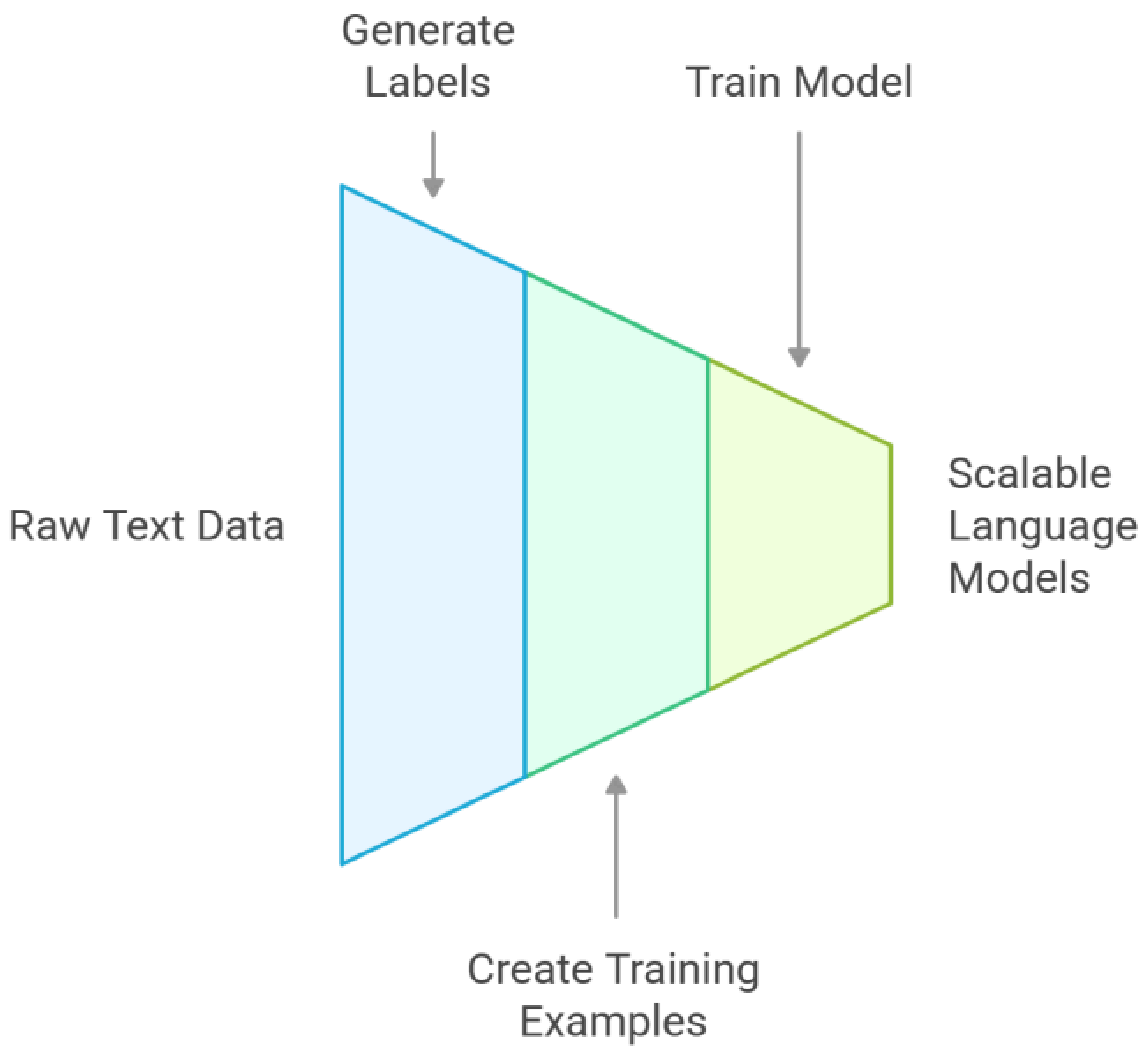
Self-supervision is a training method where a language model generates its own labels from the input data, rather than relying on human-labeled examples. For instance, in a sentence, the model might predict the next word based on the previous ones, using the text itself as both question and answer. This is significant for scaling language models because:

- It eliminates the need for costly, manually labeled datasets, allowing models to train on vast amounts of freely available text (e.g., books, websites).

- It enables massive scaling by turning a single sentence into multiple training examples, efficiently teaching the model to understand and generate language.

- This approach solves the problem of data scarcity in traditional supervised learning, making it feasible to build larger, more capable models that can tackle diverse language tasks.

Scaling Language Models through Self Supervision

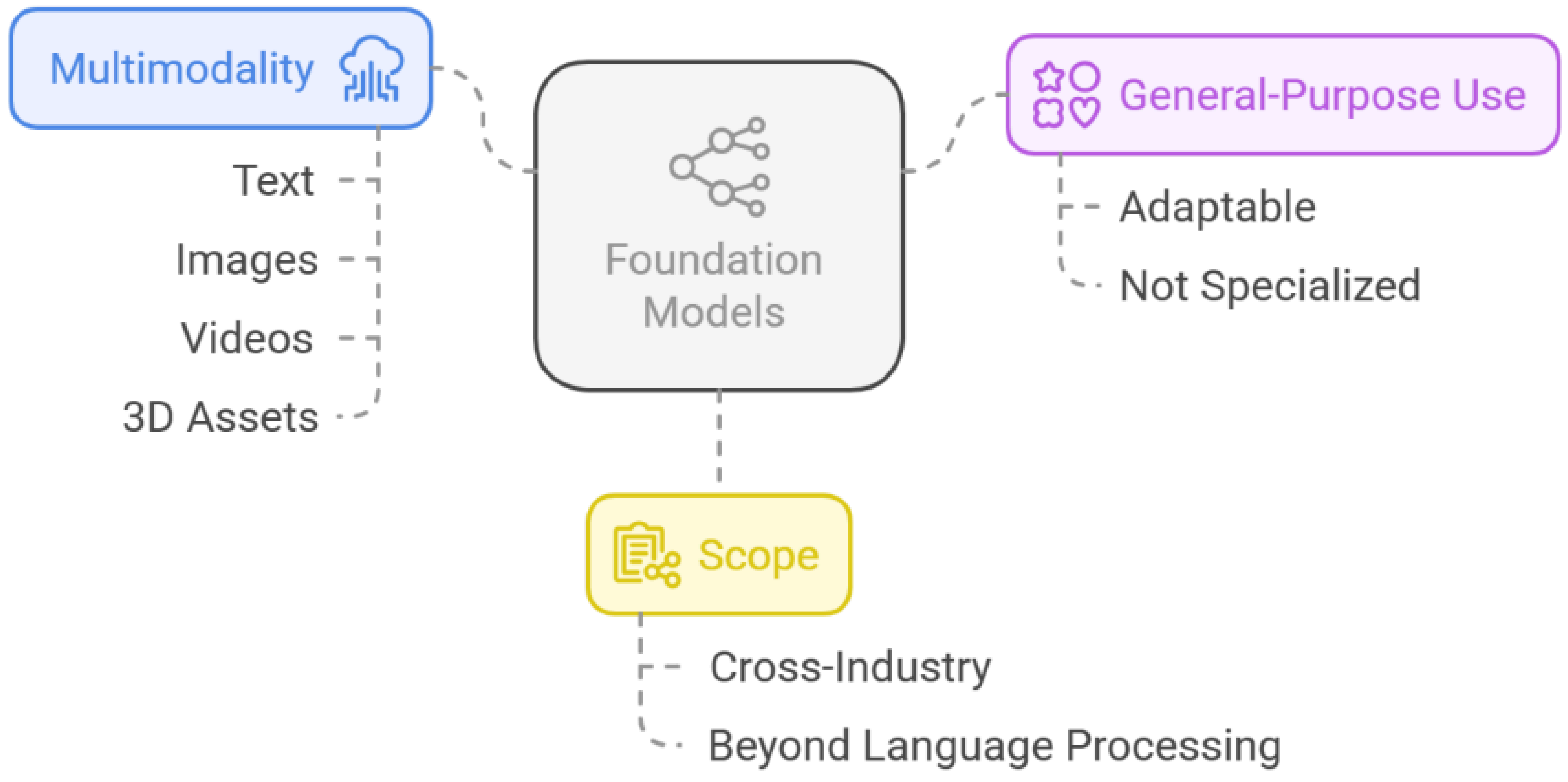


How do foundation models differ from traditional large language models (LLMs)?

Foundation models are a broader category of AI models that go beyond traditional large language models (LLMs), which focus exclusively on text. The differences include:

- **Multimodality**: Foundation models handle multiple data types—like text, images, videos, or 3D assets—while traditional LLMs are text-only. For example, models like GPT-4V can process both text and images.
- **General-Purpose Use**: Foundation models are designed as versatile bases for many applications, adaptable out of the box, whereas traditional LLMs are often more specialized.
- **Scope**: The "foundation" label highlights their role as a starting point for diverse tasks across industries, not just language processing.

This makes foundation models more flexible, solving a wider range of problems, from image captioning to multimodal content generation, compared to the language-specific focus of traditional LLMs.



What are the differences between prompt engineering and finetuning as model adaptation techniques?

Both prompt engineering and finetuning adapt AI models to specific tasks, but they differ in how they work:

- **Prompt Engineering**: This involves crafting specific instructions or examples in the input (the "prompt") to guide the model’s output without altering its internal parameters (weights). It’s quick, requires minimal data, and is ideal for simple adaptations, but it may struggle with highly specialized tasks.

- **Finetuning**: This updates the model’s weights by retraining it on a task-specific dataset. It demands more data and computing power but yields better performance for complex or niche tasks, like tailoring a model for legal document analysis.

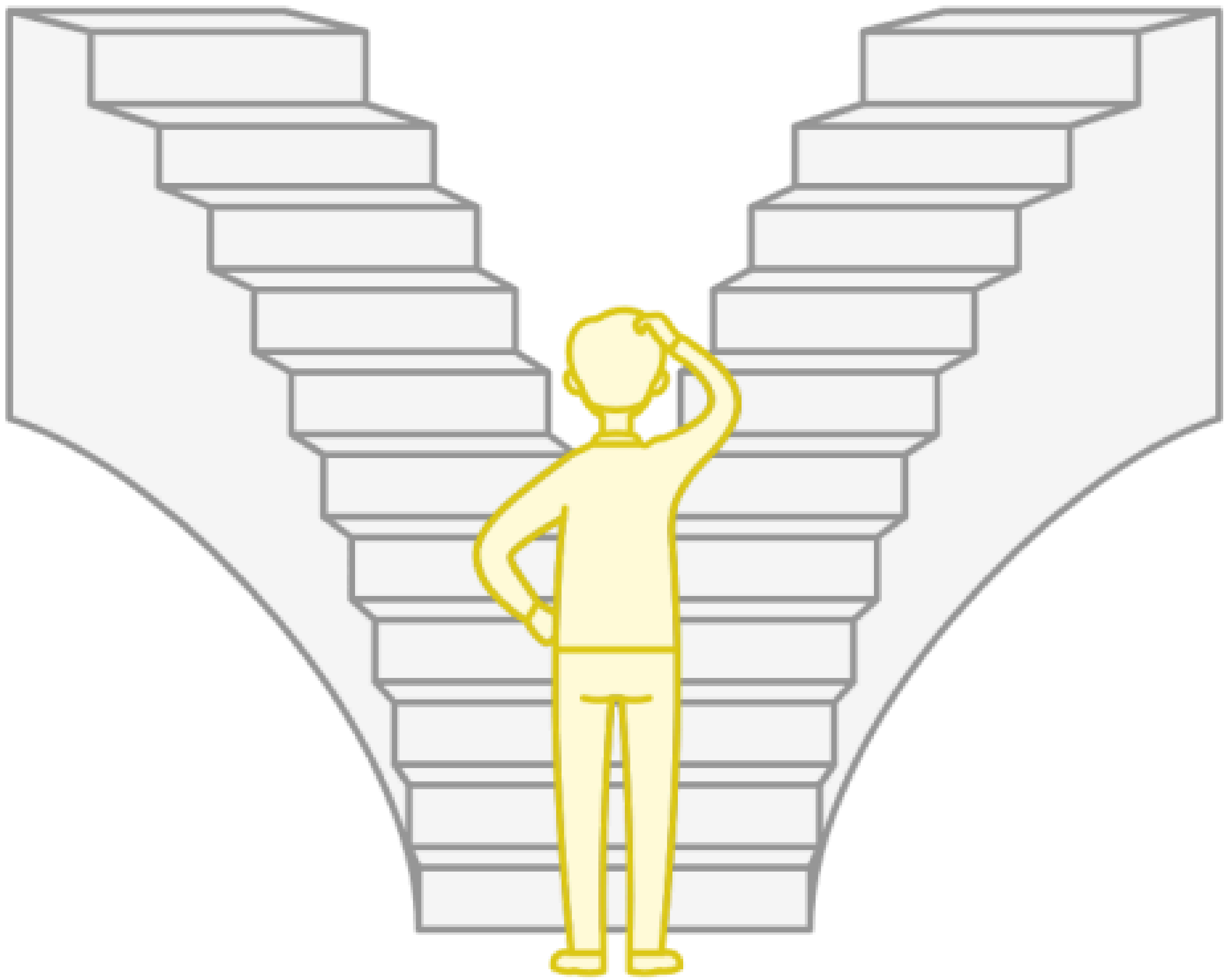
Prompt engineering solves immediate adaptability needs with less effort, while finetuning addresses deeper customization, enhancing a model's problem-solving precision.

Prompt Engineering

Quick and minimal effort for simple tasks, but may struggle with complexity.

Finetuning

Provides deep customization and precision for complex tasks, but requires more resources.



What are the three layers of the AI engineering stack, and what are their primary responsibilities?

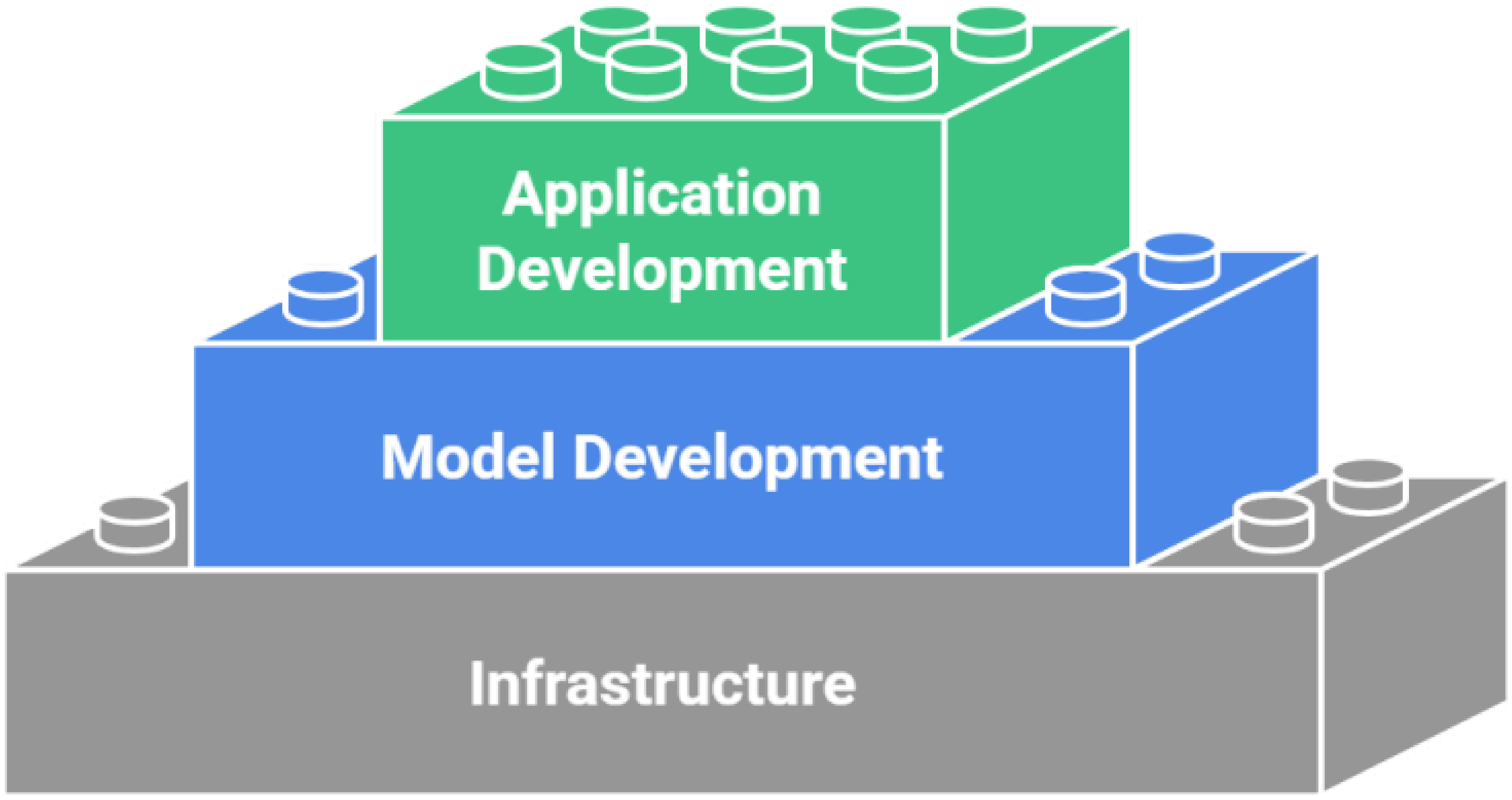
The AI engineering stack has three layers, each with distinct roles:

- **Application Development**: Focuses on building user-facing AI solutions. Responsibilities include designing prompts, constructing context, evaluating outputs, and creating interfaces (e.g., a chatbot UI).

- **Model Development**: Handles the creation and refinement of AI models. Tasks include training, finetuning, dataset preparation, and optimizing inference speed.

- **Infrastructure**: Supports the system’s backbone. It manages compute resources, data storage, model serving (e.g., via APIs), and monitoring performance.

Together, these layers solve the problem of deploying scalable, functional AI systems

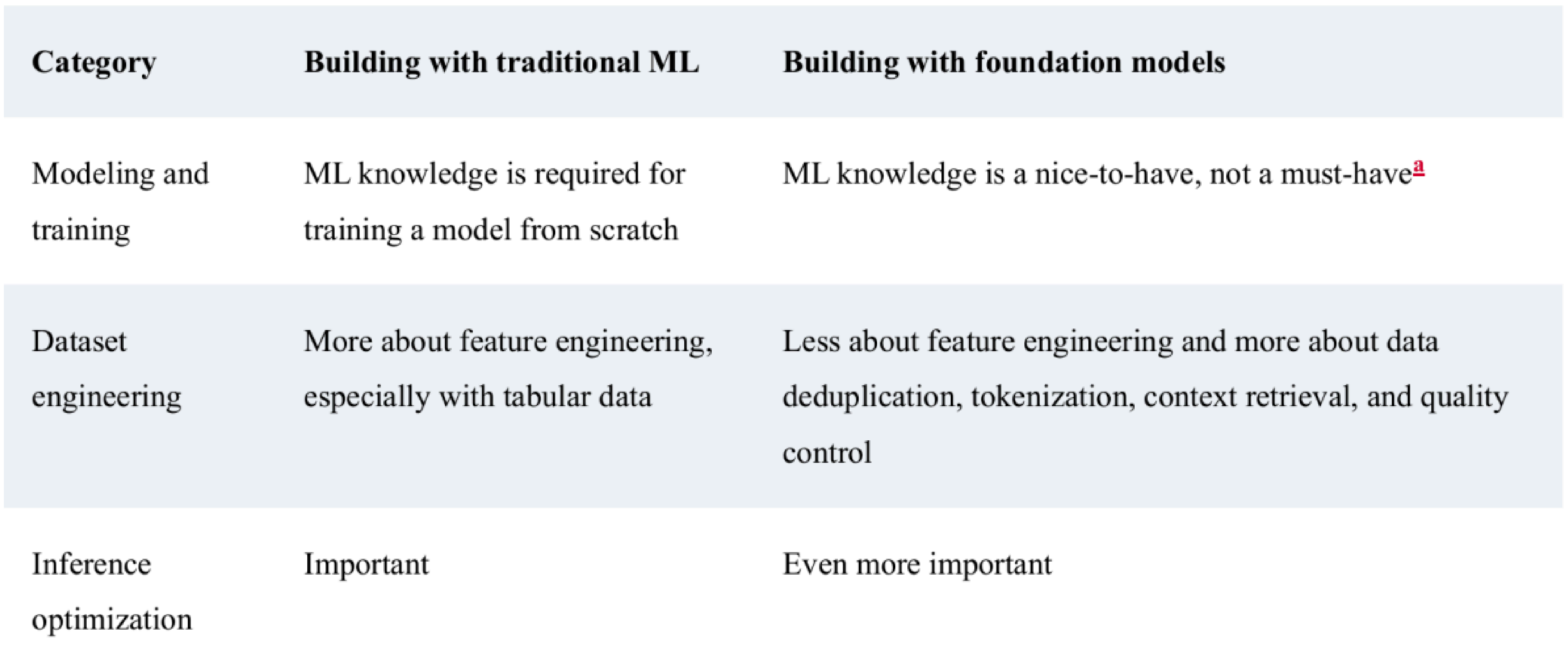


How does AI engineering differ from traditional ML engineering?

Traditional ML refers to all ML before foundation models. AI engineering stands apart from traditional ML engineering in:

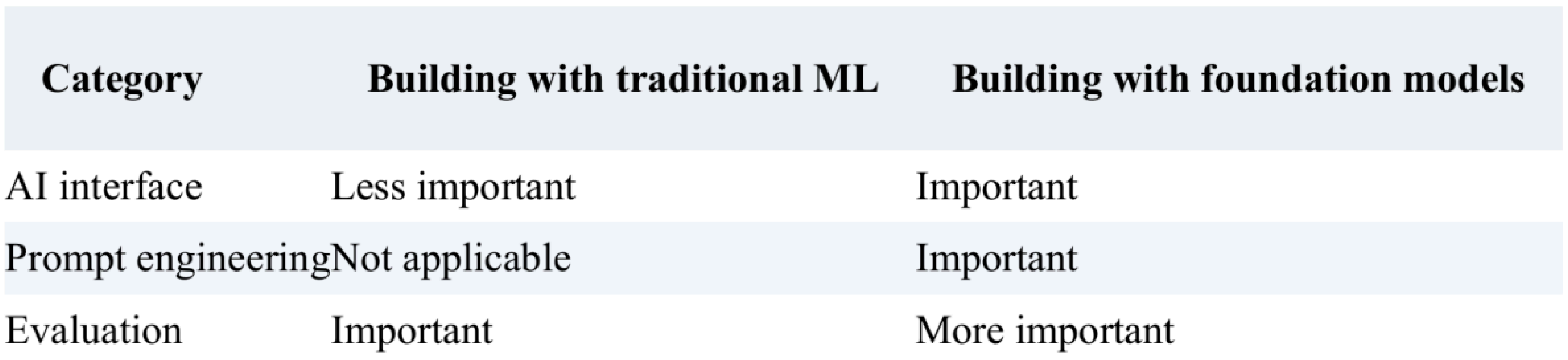
- **Model Approach**: AI engineering uses pre-trained models (e.g., via APIs), while traditional ML builds models from scratch.
- **Scale and Compute**: AI engineering handles larger models, demanding more computational optimization.
- **Evaluation**: AI engineering faces trickier evaluation due to open-ended tasks, unlike traditional ML's simpler metrics.

This shift solves the problem of rapidly deploying advanced AI without starting from zero.



Category	Building with traditional ML	Building with foundation models
Modeling and training	ML knowledge is required for training a model from scratch	ML knowledge is a nice-to-have, not a must-have[a]
Dataset engineering	More about feature engineering, especially with tabular data	Less about feature engineering and more about data deduplication, tokenization, context retrieval, and quality control
Inference optimization	Important	Even more important

The importance of different categories in app development for AI engineering and ML engineering is shown below:



Category	Building with traditional ML	Building with foundation models
AI interface	Less important	Important
Prompt engineering	Not applicable	Important
Evaluation	Important	More important

What are the immediate economic impacts of foundation models on building AI applications?

Foundation models have transformed the economics of AI application development by:

- **Cutting Costs**: APIs provide access to powerful models cheaply, reducing the need for expensive in-house training.

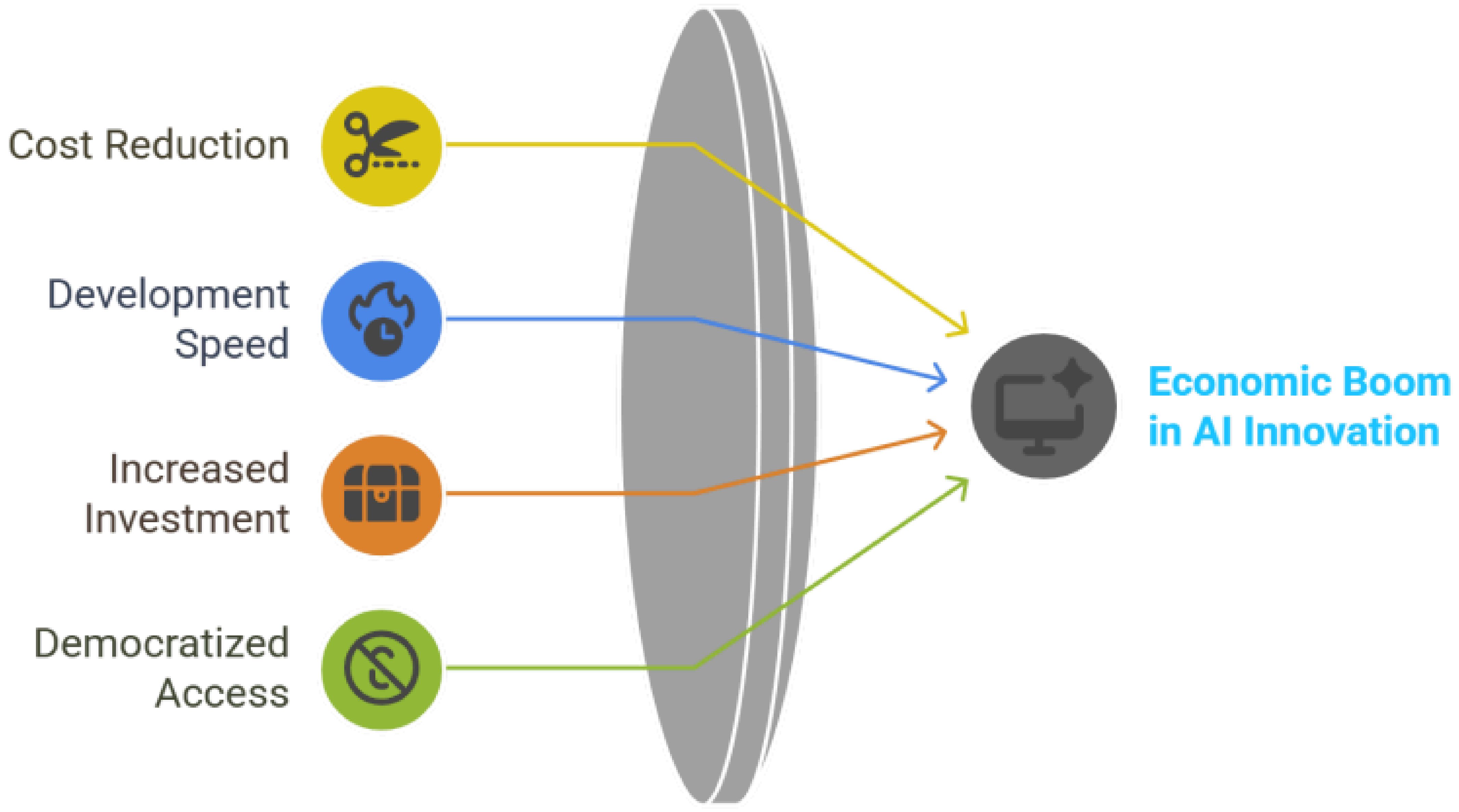
- **Speeding Development**: Adapting pre-trained models is faster than building anew, accelerating market entry.

- **Driving Investment**: Their potential has spurred funding and adoption across industries.

- **Democratizing Access**: Non-experts can create applications, lowering barriers and solving the problem of exclusivity in AI development.

This has fueled an economic boom in

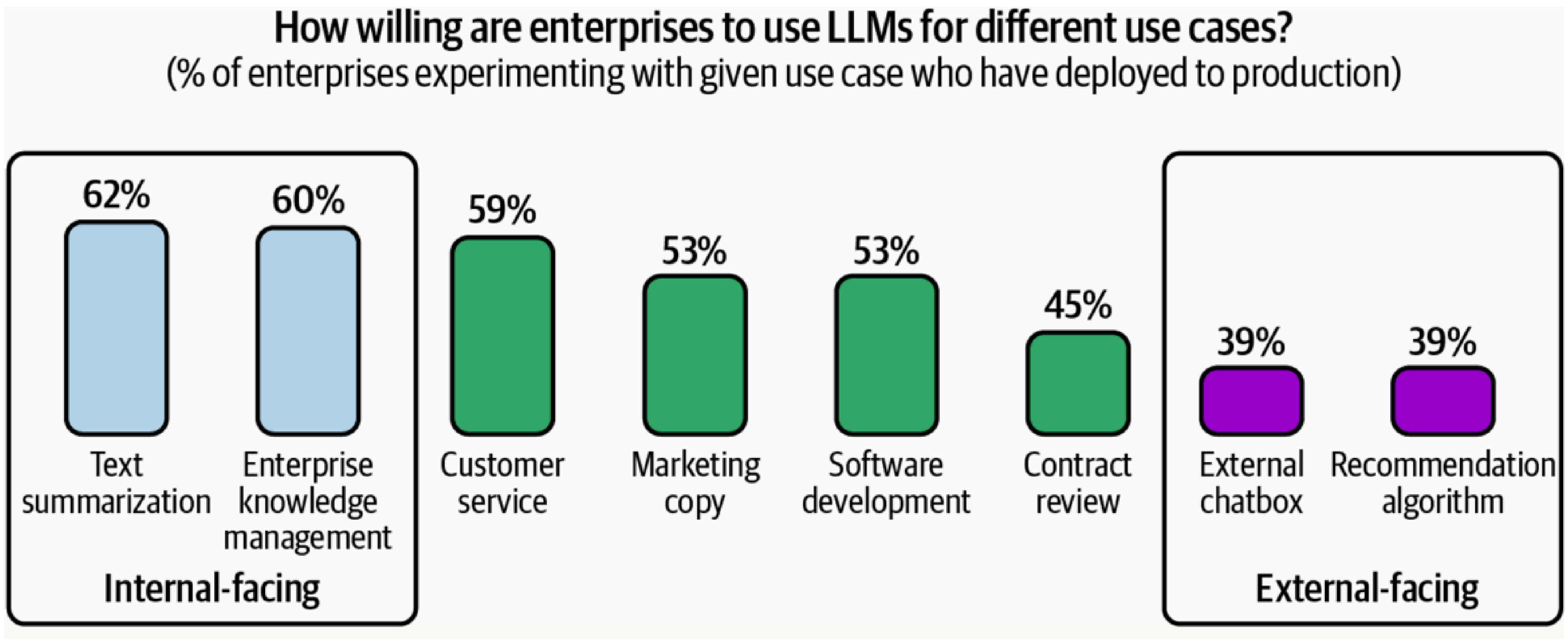
AI innovation



What type of LLM applications are typically deployed by enterprises?

The enterprise world generally prefers applications with lower risks. For example, a **2024 a16z Growth report** showed that companies are faster to deploy internal-facing applications (internal knowledge management) than external-facing applications (customer support chatbots), as shown in Figure below.

Internal applications help companies develop their AI engineering expertise while minimizing the risks associated with data privacy, compliance, and potential catastrophic failures. Similarly, while foundation models are open-ended and can be used for any task, many applications built on top of them are still close ended, such as classification.



Share this post with others!

Repost

Data Science Manager with **15 years experience** in **AI** || Certified Generative AI Specialist || Experienced **AI Leader** & Coach for ML, DL, and NLP || PhD @ **IIT-Bombay**

Follow