Introduction

- AI Engineering:
  - Focuses less on modeling and training, and more on model adaption and evaluation
  - Works with models that are bigger and consume more compute resources, and incur higher latency than traditional ML models
- Token:
  - Basic unit of a language model (can be a word, character, or part of a word)
  - Allow models to break words into meaningful components
  - There are fewer unique tokens than unique words
- Masked language model: Trained to predict a token anywhere in a sequence, using the context from both before and after the missing token → fill in the blank
- Autoregressive language model:
  - Trained to predict the next token in a sequence, using only the preceding tokens
  - Can continually generate one token after another
- Generative models can generate open-ended outputs
- Classical ML models are closed-ended since they only can outputs that are among predefined values
- Completions are predictions based on probabilities, and not guaranteed to be correct
- Self-supervision:
  - Model infers labels from the input data
  - Language modeling is self-supervised because each input sequence provides both the labels (tokens to be predicted) and the contexts the model can use to predict these labels → labels are inferred from the input data
- Foundation model: Can be built upon for different needs
- Multimodal model: Models that can work with more than one data modality
- Agents: AI that can plan and use tools
- Three layers of AI stack:
  - Application development: Provide a model with good prompts and necessary context
  - Model development: Tooling for developing models, including frameworks for modeling, training, finetuning, and inference optimization
  - Infrastructure: Tooling for model infrastructure, including tooling for model serving, managing data and compute, and monitoring
- Inference optimization: Make models faster and cheaper
- Prompt engineering: Get AI models to express the desirable behaviors from the input alone, without changing the model weights

Foundation Models

- Sampling:
  - Process of constructing output
  - How does a model chooses an output from all possible options
  - Makes an AI's output probabilistic
  - To generate the next token, the model first computes the probability distribution over all tokens in the vocabulary
  - Greedy sampling: Always pick the most likely outcome → creates boring outputs since the most common words are responded
  - Instead of picking the most likely token, the model can sample the next tokens ac-

cording to the probability distribution

- Sampling strategies:
  - Temperature: A higher temperature reduces the probabilities of common tokens and increases the probability of rarer tokens
  - Top-k: Pick top-k logits to perform softmax over these → smaler k makes the text more predictable but less interesting
  - Top-p:
    * Allows for a more dynamic selection of values to be sampled from than top-k
    * Model sums the probabilities of the most likely next values in descending order and stops when the sum reaches p → only the values within the cumulative probability are considered
    * Focuses only on the set of most relevant values for each context and allows outputs to be more contextually appropriate
    * Typically range from 0.95 to 0.99
- Logit vector:
  - Corresponds to one possible value → one token in the model's vocabulary
  - Logit vector size is the size of the vocabulary
  - Do not represent probabilities
  - To convert logits to probabilities, a softmax layer is often used
- Sparsity allows for more efficient data storage and computation (large percentage of zero-value parameters)
- Number of tokens in a model's dataset isn't the same as its number of training tokens → the number of training tokens measures the tokens that the model is trained on
- FLOP: Unit for a model's compute requirement → measures the number of floating point operations performed for a certain task
- Utilization: Measures how much of the maximum compute capacity one can use
- Post-Training:
  - Issues of a pre-trained model:
    * Self-supervision optimizes the model for text completion, not conversations
    * Outputs can be biased or wrong
  - Solution:
    * Supervised finetuning (SFT): finetune the pre-trained model on high-quality instruction data to optimize models for conversations instead of completion
    * Preference finetuning: Finetune the model to output responses that aligh with human preferences
  - Preference finetuning is typically done with reinforcement learning or reinforcement learning from human feedback (RLHF)
  - Pre-training optimizes token-level quality
  - Post-training optimizes the model to generate responses that users prefer
- RLHF:
  - Train a reward model that scores the foundation model's output
  - Optimize the foundation model to generate responses for which the reward model will give maximal scores
- Reward model:
  - Outputs a score for how good the response is
  - Generate multiple responses for each prompt and the resulting data is comparison data with the format (prompt, winning_response, losing_response)

- Robustness: A model is robust if it doesn't dramatically change its outputs with small variations in the input
- Inconsistency: Generating very different responses for the same or slightly different prompts
- Hallucination:
  - Response that isn't grounded in facts
  - Cannot differentiate between the data it's given and the data it generates
  - Snowball hallucinations: Model continues ti justify the initial wrong assumption
  - Also caused by the mismatch between the model's internal knowledge and the labeler's internal knowledge