Systems for Machine Learning

Arvind Krishnamurthy

Lecture Outline

- What is systems for ML?
- Course information and logistics
- Introduction to transformer models and attention

Machine Learning Applications

Image recognition

Speech recognition

Self-driving cars

Weather predictions

and so on

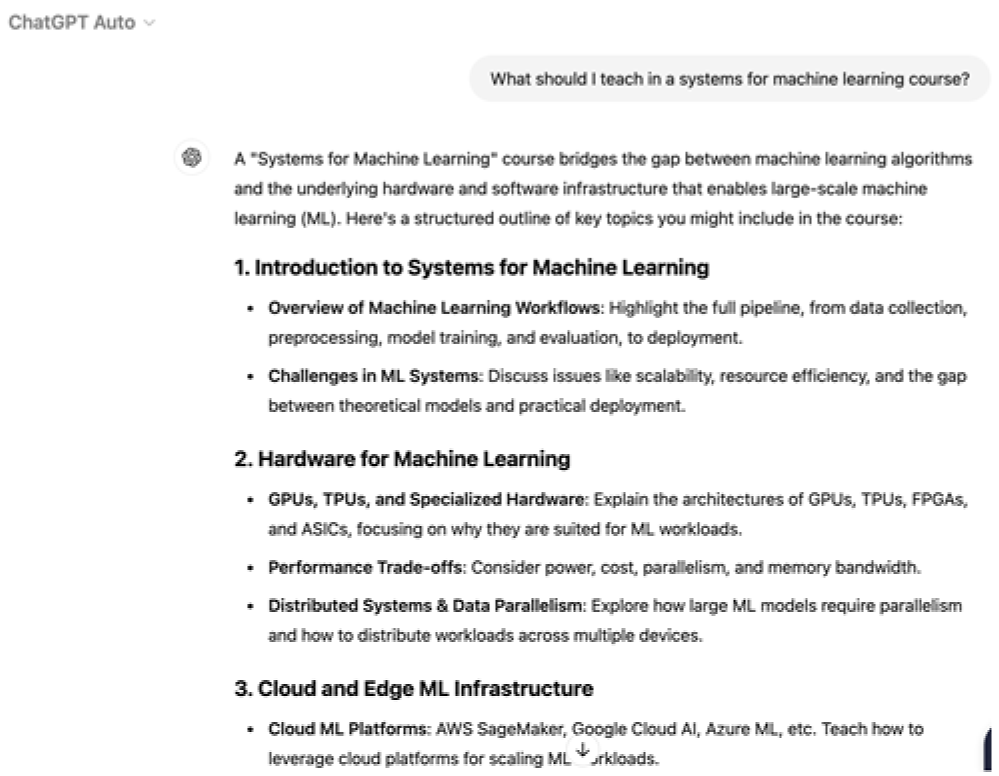
Higgs challenge

Advent of Generative AI

Content generation

“A photo of a cat wearing a diwali dress” - my daughter’s interaction with Bard

Large language model



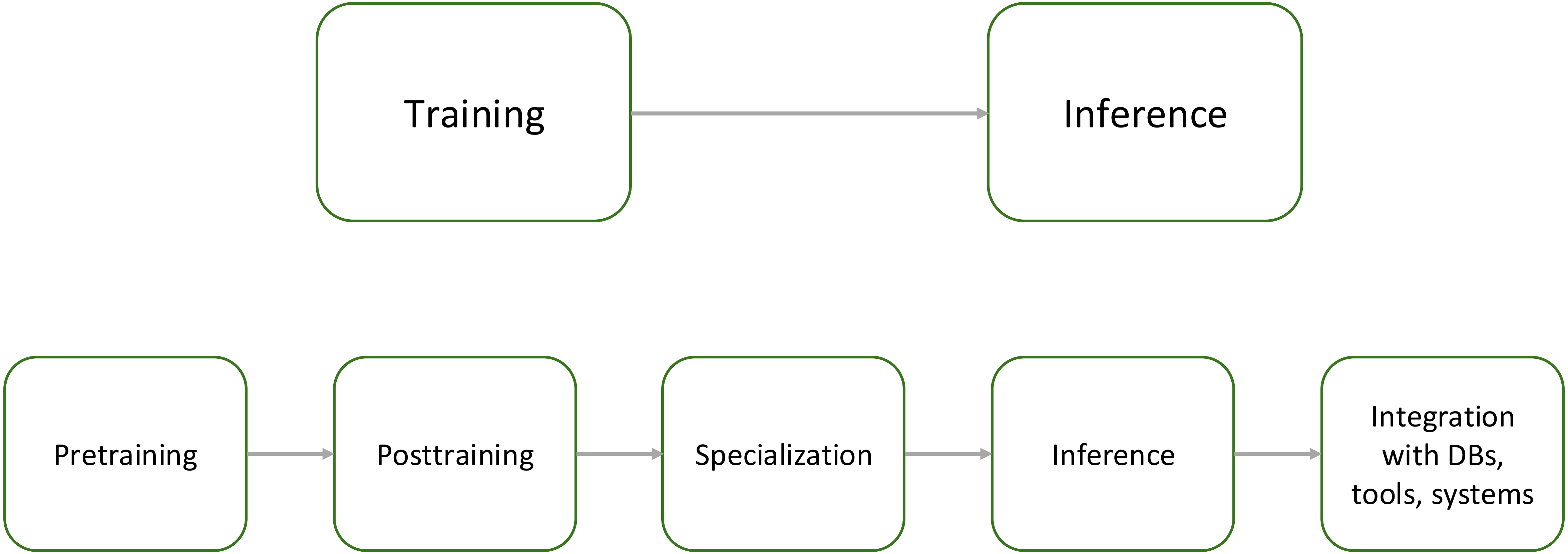
Systems for ML

- How do we enable development and delivery of ML models as efficiently as possible?
- Can be viewed through two lenses:
  - Life-cycle of ML model development and deployment
  - Systems stack at every stage of the process

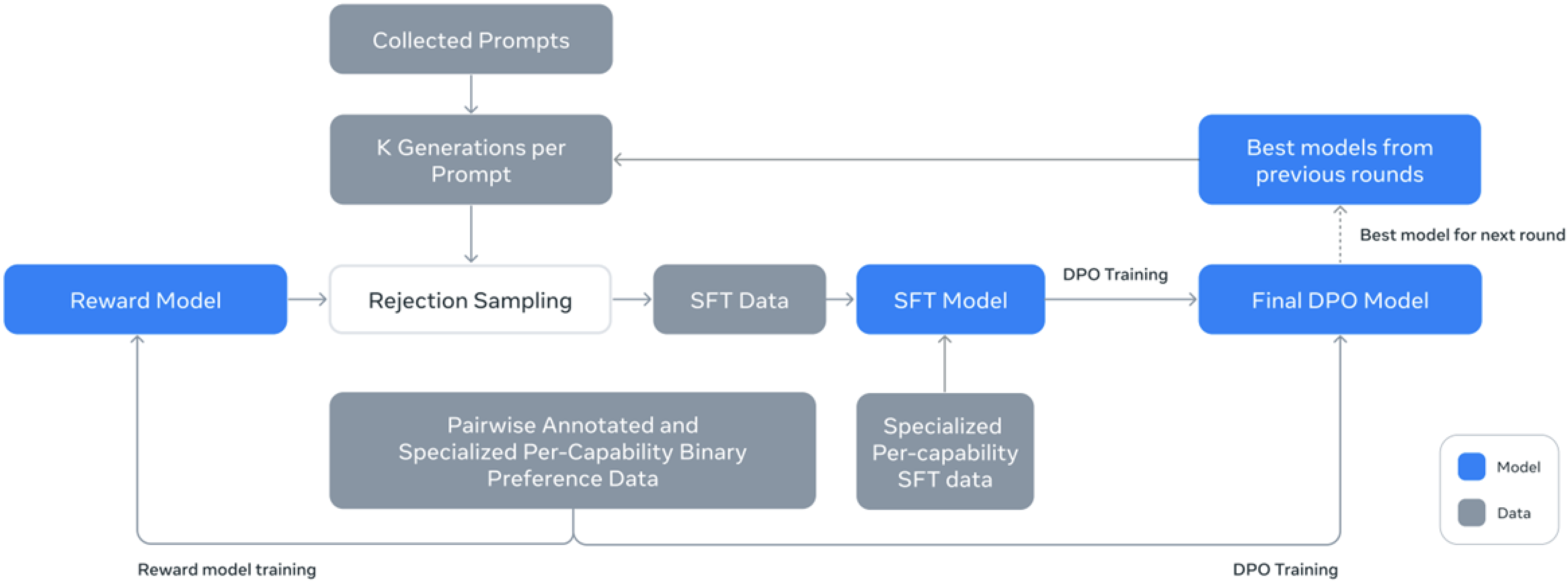
Systems for ML

- How do we enable development and delivery of ML models as efficiently as possible?
- Can be viewed through two lenses:
  - Life-cycle of ML model development and deployment
  - Systems stack at every stage of the process

Nuanced view of ML life cycle



Post-training in Llama 3



Cross-stack view of ML Systems

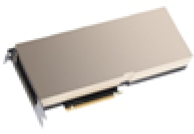
- ML is expensive! Estimated investment on x.AI's cluster is 4B, estimated cost of Llama 3 training is 300M to 700M
- ML systems focus is on improving efficiency & programmability
- Need performance improvements at every layer of the stack

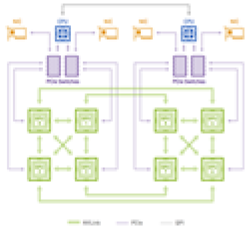
Distributed Systems & Apps

Networked Accelerators

Node-level Systems

Compilers | **Libraries**





Hardware

- GPUs and TPUs have interesting hardware models; understanding the hardware model is crucial to optimizing upper layers

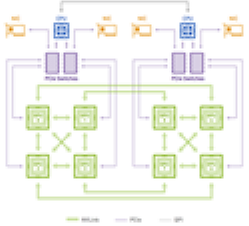
- Custom interconnects at the “node” level and the “cluster” level allow for fast communications

Distributed Systems & Apps

Networked Accelerators

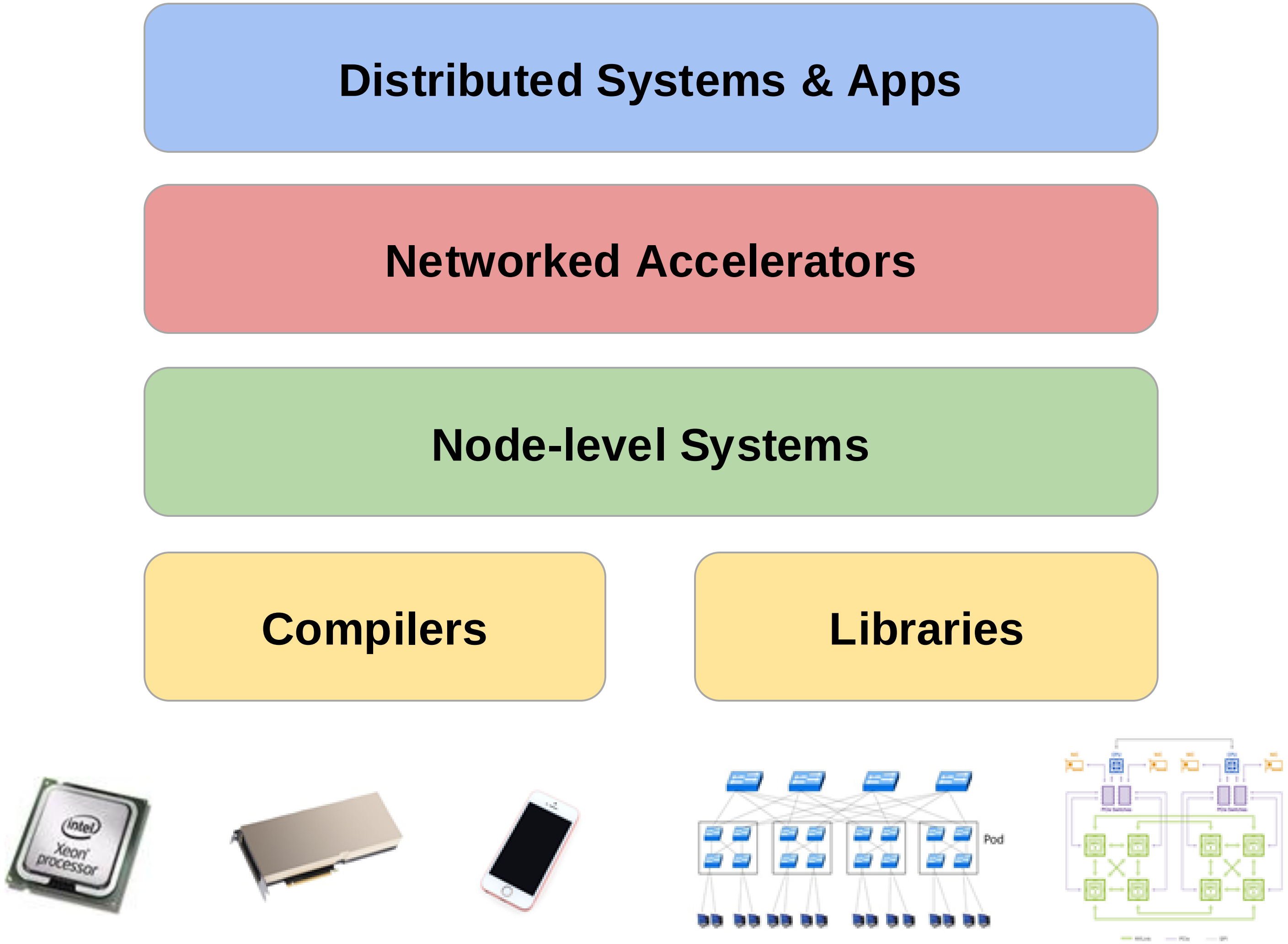
Node-level Systems

Compilers | **Libraries**



Compilers and Libraries

- Target the performance of low-level kernels that are crucial to overall system performance
- Focus on LLMs means that custom libraries can have broad impact
- Compilers have significantly evolved to generate optimized code for specific tensors, fusing operations, and generating auto-differentiations.



Node-level Systems

- Batching & sharding of models, quantization
- Reuse of computed state & managing use of memory
- Orchestration of multiple accelerators and the CPU

Distributed Systems & Apps

Networked Accelerators

Node-level Systems

Compilers | **Libraries**

Networked & Distributed Systems

- Fast collectives
- Training: orchestration of different kinds of parallelism
- Inference: load balancing and cluster management of nodes

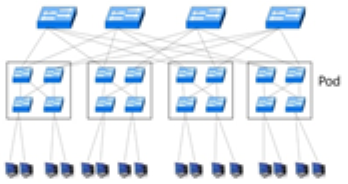
Distributed Systems & Apps

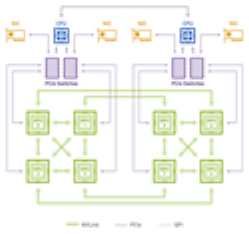
Networked Accelerators

Node-level Systems

Compilers | **Libraries**







Networked & Distributed Systems

ML ecosystem is extremely vibrant; new architectures, new models, new apps, …

Many questions:

- How to optimize when models are heterogeneous and composed?
- How to disaggregate and optimize for e2e use cases?
- How to integrate AI/ML with generic distributed computing and database systems?
- How to support AI/ML workloads with data management capabilities?

This Course

Centered around transformers and LLMs, but learnings carry over

Broadly touching on the two lenses of viewing ML

- Systems stack for ML
- Life cycle of ML

Course logistics:

- Three assignments: key-value caching in LLM, GPU performance modeling, and advanced multi-stage inference
- Quarter-long course project
- Lectures and readings will evolve during the course! Also, many guest lectures

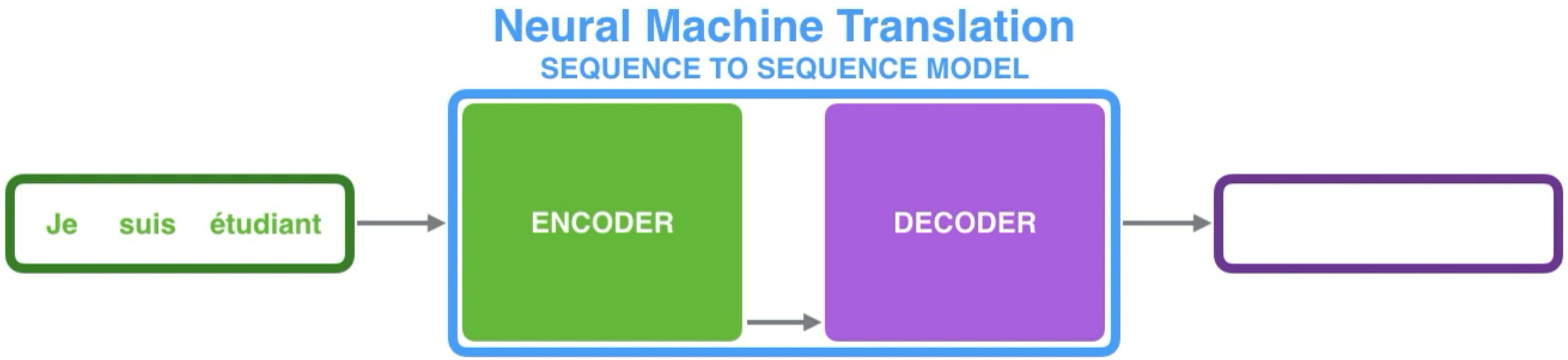
Transformers & Attention

- Evolution from Seq2Seq models
- Attention mechanism
- Transformer model
- Performance implications of Transformer models

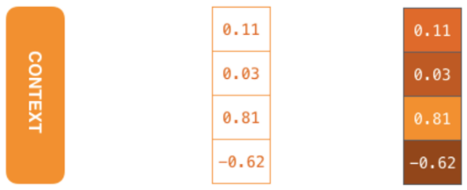
Seq2Seq Models for Machine Translation

Generalization of word prediction based on previous k words ⇒ translation of a sequence

Encoder-decode models with a fixed context communication mechanism (encoder bottleneck)



Context Vector

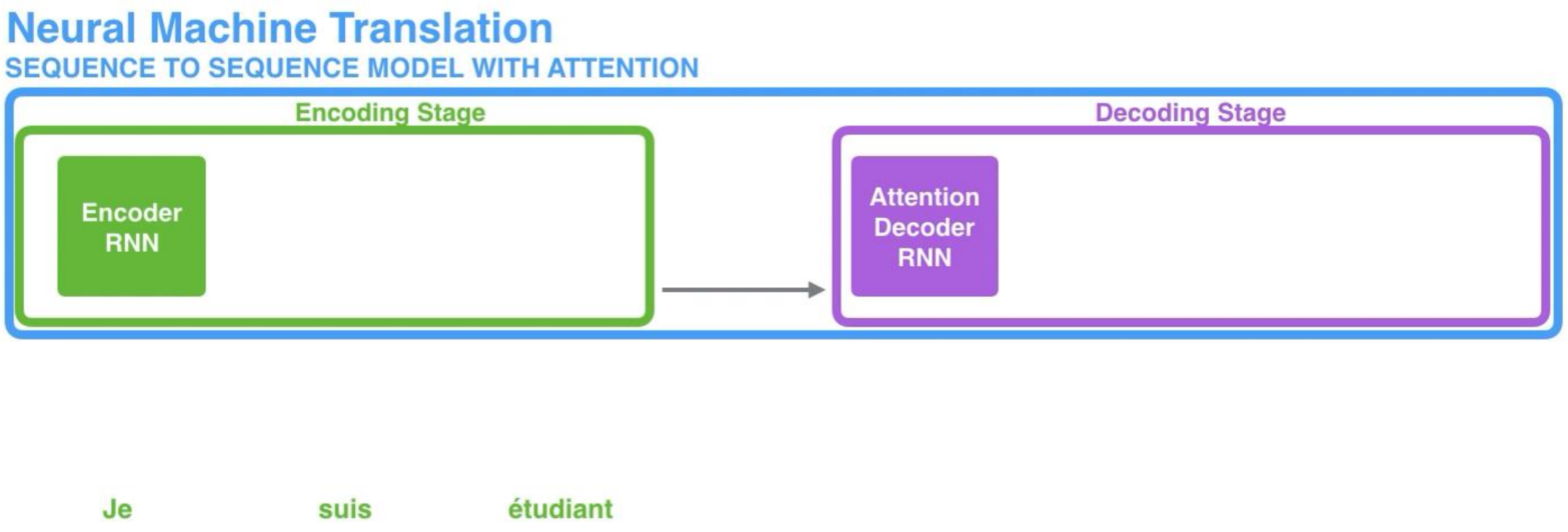


Animations from Jay Alammar

Attention Mechanism for Seq2Seq

What if the decoder can perform a "soft search" on all of the encoder state?

Find the relevance of the encoder "hidden states" for the current work being generated by decoder



Animations from Jay Alammar

Attention Mechanism (contd.)

Generate context vector for each step of decoding based on all encoder hidden states

Animations from Jay Alammar

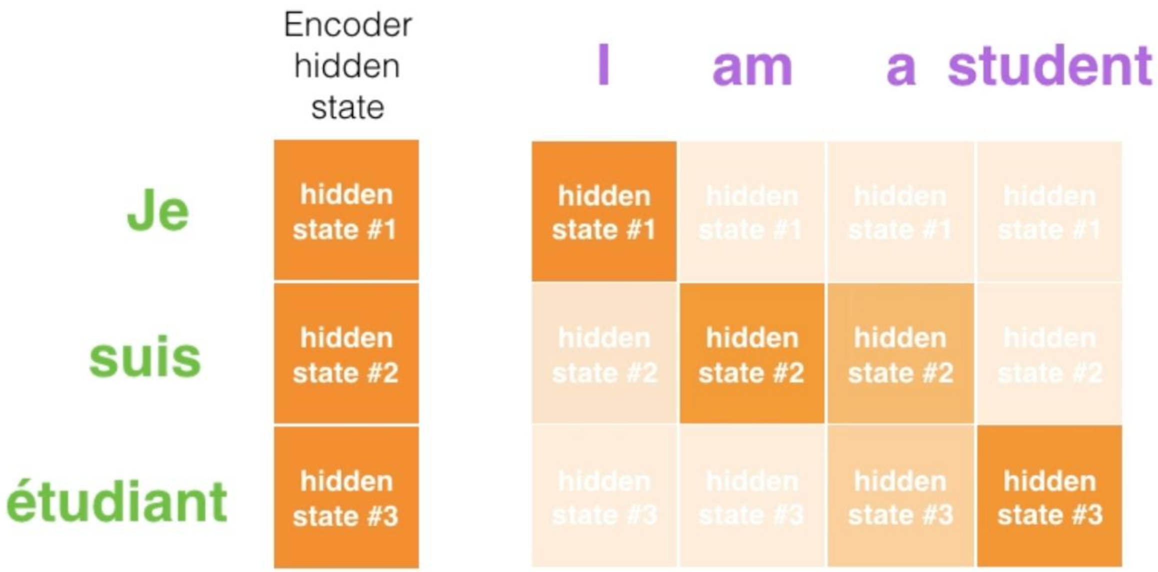
Attention Mechanism (contd.)

Generate context vector for each step of decoding based on all encoder hidden states

Animations from Jay Alammar

Attention Mechanism (contd.)

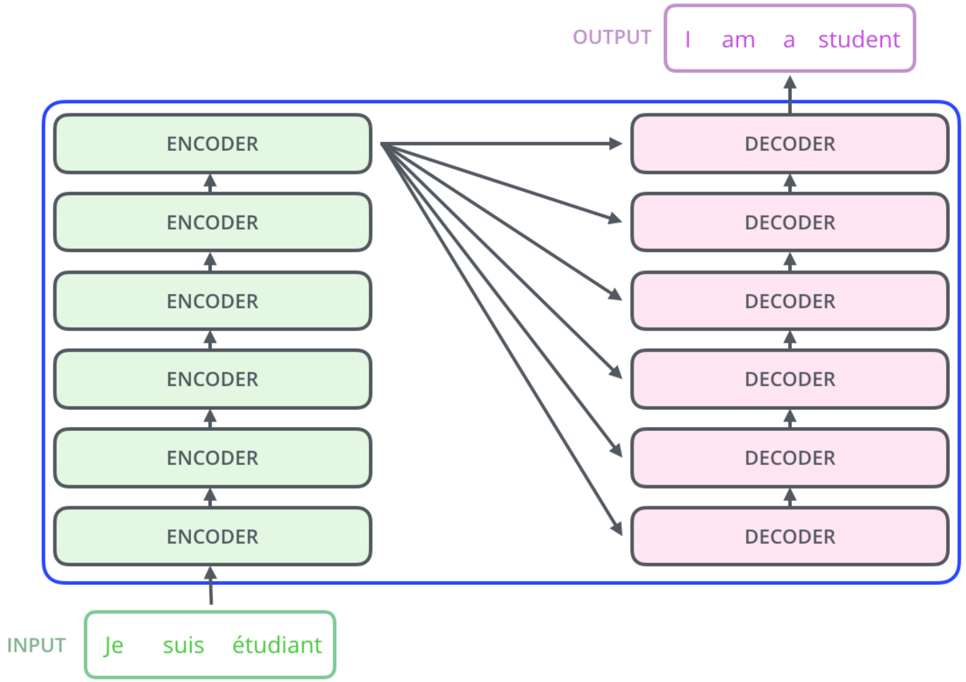
Example translation and how attention associates relevant words

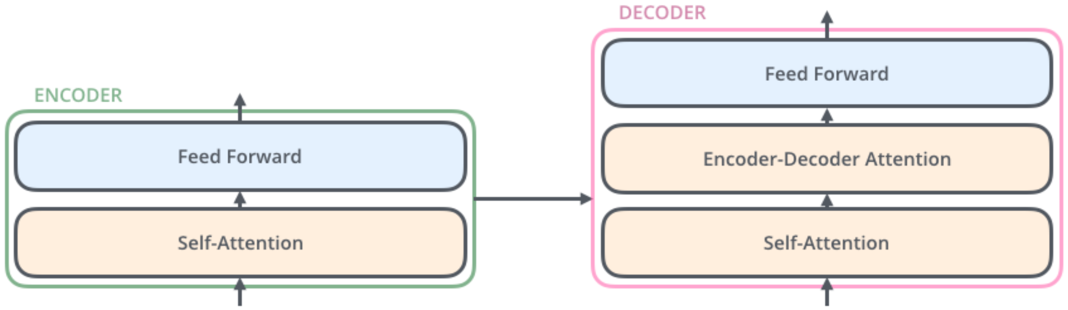


Animations from Jay Alammar

Transformers & Attention

Generalization as well as a more focused use of attention



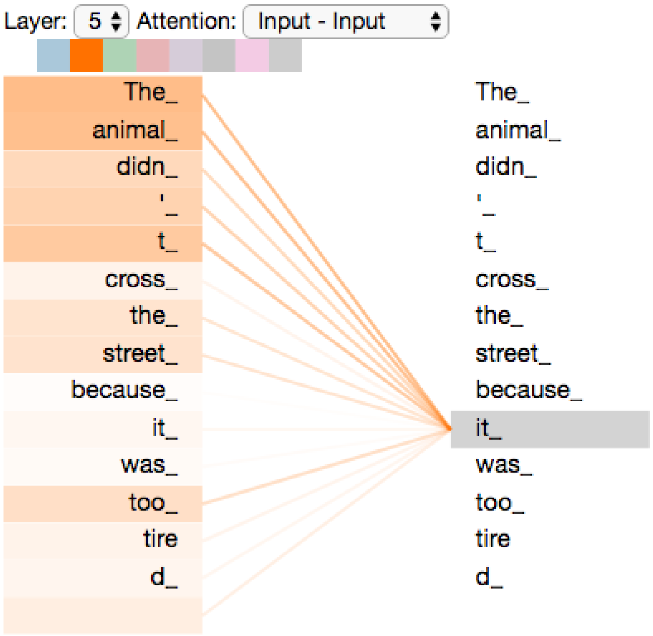


Animations from Jay Alammar

Self Attention Mechanism

View this as the “communication phase” of a transformer model; tokens interact with each other

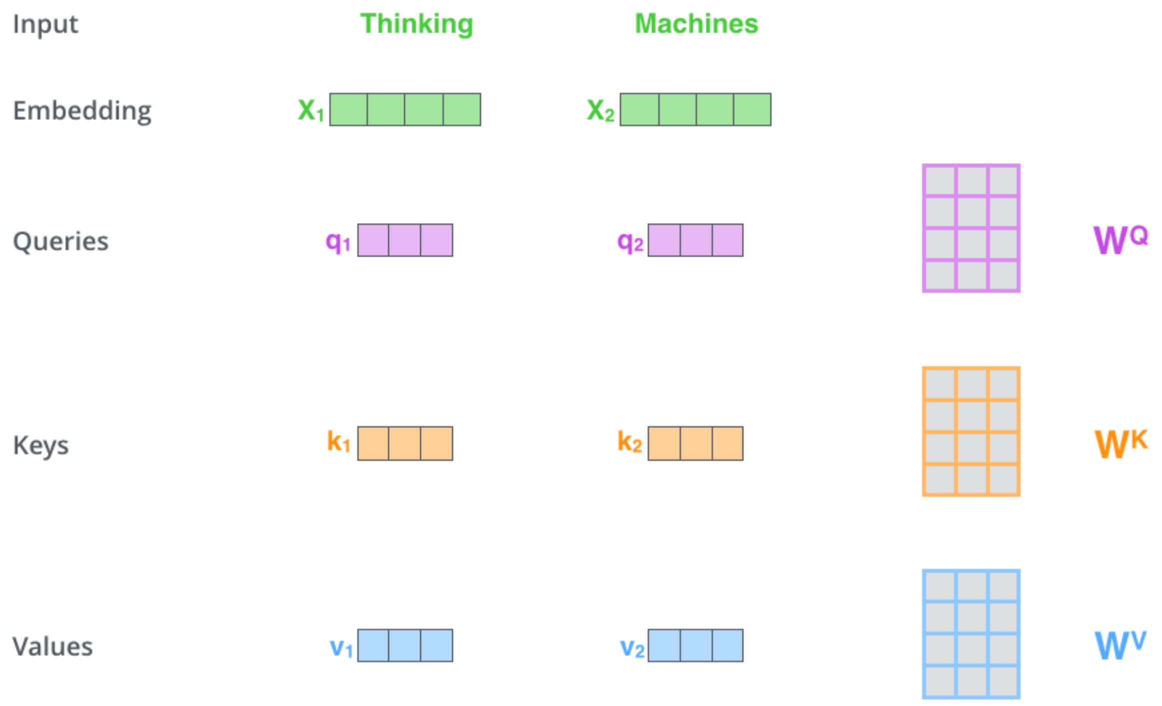
”The animal didn't cross the street because it was too tired”



Animations from Jay Alammar

Self Attention in Detail

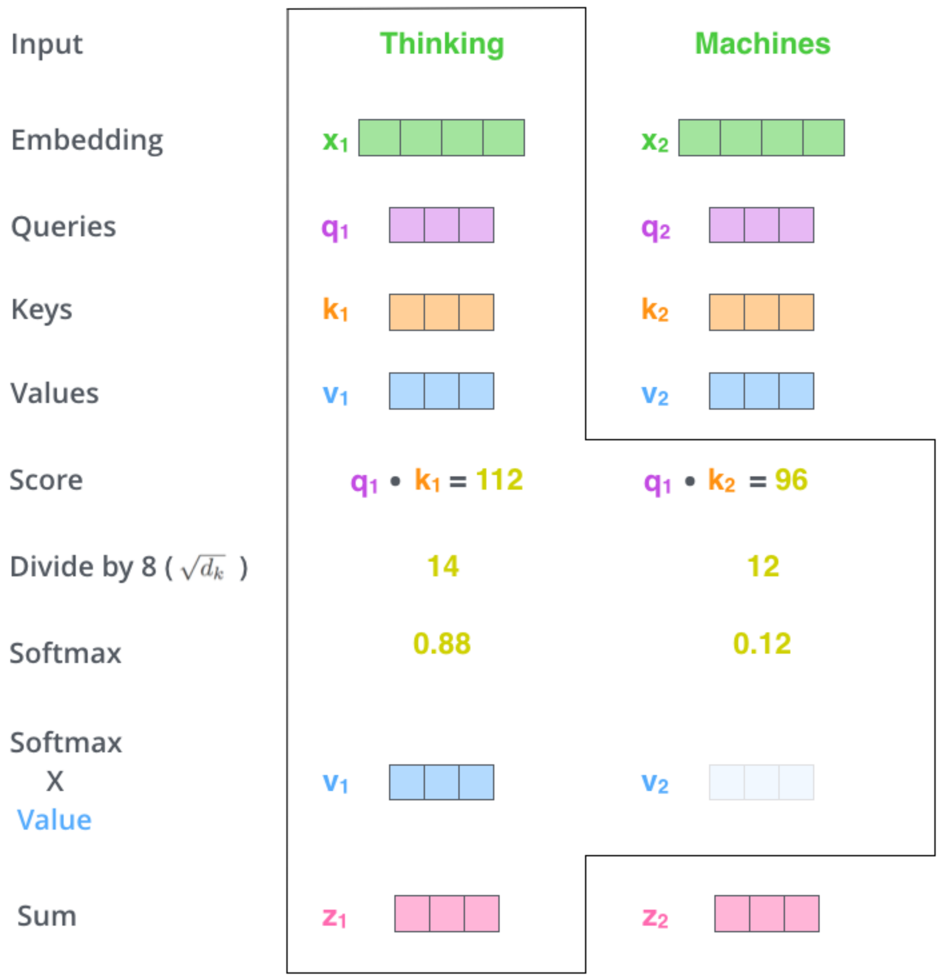
View this as the "communication phase" of a transformer model; tokens interact with each other



- Query, key, and value vectors are obtained from the embeddings
- Represent different abstractions of the token
  - Query: what the token is interested in finding out from other tokens
  - Key: what semantics the token is offering to other tokens
  - Value: what is the value associated with the semantics

Animations from Jay Alammar

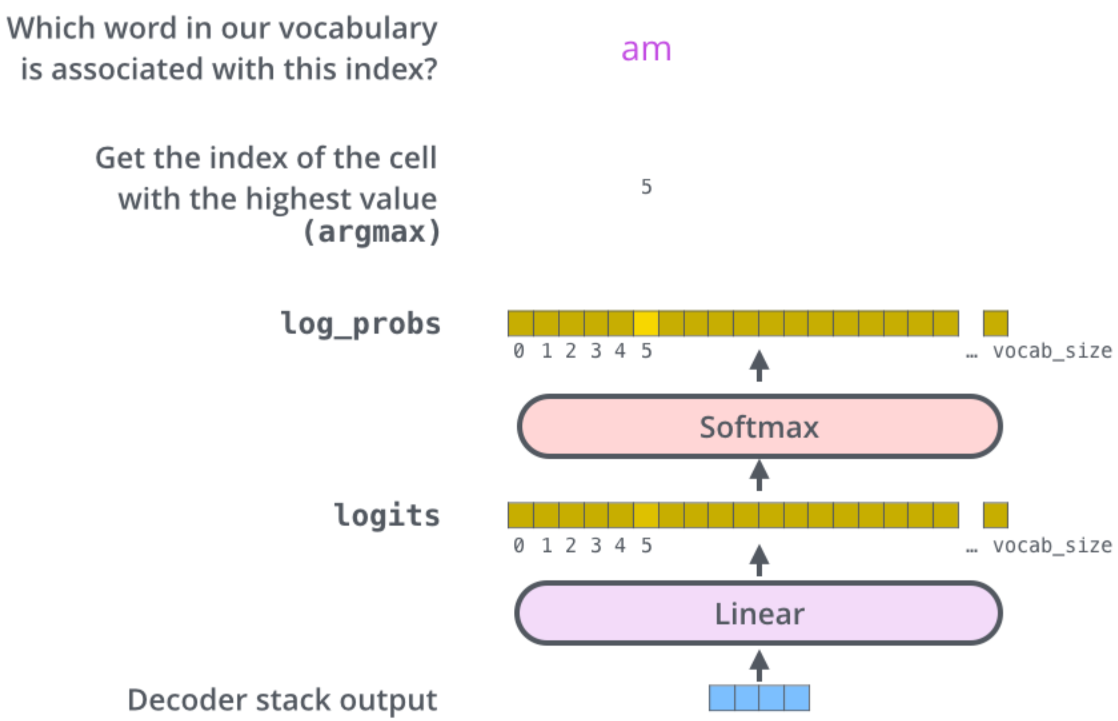
Self Attention in Detail



- Query * Key represents the strength of communication between tokens
- Softmaxed weighted sum of value represents the overall meaning associated with a token
- This resulting "z" value is fed through the feed-forward network (a multi-layer perceptron with no cross interactions)
  - FFN is the "compute" phase
  - Typically there are "multiple heads", each producing a "z" value, and they are all concatenated before the FFN phase

Animations from Jay Alammar

Final Layer



- Compute logits for every possible token
- Transform them into probabilities
- Identify a token based on the probabilities

Animations from Jay Alammar

Some other details

- Popular LLMs are primarily decoder-only models that perform predictions of the next token
- Can serve as multi-model as long as there is a way to tokenize the input
- “Prefill” stage is performing the decoder computation on the input sequence
- “Decode” stage is the prediction process that terminates when the end-of-string token is generated