

Wrap Up

Mausam

Grading

- 25% final exam
- 20% midterm
- 12.5% A1
- 12.5% A2
- 5% A2.1
- 15% A3
- 10% Quiz (best 5 out of 6)

Key Points of the Course

Key Points: NLP Overview

- Challenges
 - Ambiguity, Ambiguity, Ambiguity, Sparsity
- Words: Morphology
- Sentences: Syntax
 - POS tagging, NP chunking, Parsing
- Sentences/Documents: Semantics
 - Words/Bigrams encode meanings, but are also sparse
 - Distributional Semantics, Shallow semantics
 - Patterns: bootstrapping
- Documents: Coreference, Discourse
- Applications
 - Information Extraction, Machine Translation, Summarization, Dialog

Key Points: Modeling

- Models
 - Logical vs. Probabilistic vs. Neural
 - Representation: Bag of Word-level, Linear, Tree
 - Generative vs. Discriminative
 - Smoothing / Regularization
 - Pipeline vs. Joint inference vs. end-to-end inference
- Training Data
 - A lot: Supervised
 - None: Unsupervised/Self-supervised
 - A little: Semi-supervised
 - Distantly supervised
 - Active
- Features, Features, Features: local or global
- Learning Representations

Key Points: Data Insights

- Data annotation: be a linguist!
- Crowdsourced Data Curation
- Data harvesting
 - Example: bootstrapping
 - Example: Summarization
- Indirectly related data
 - Example: distant supervision
- 6 • Data augmentation

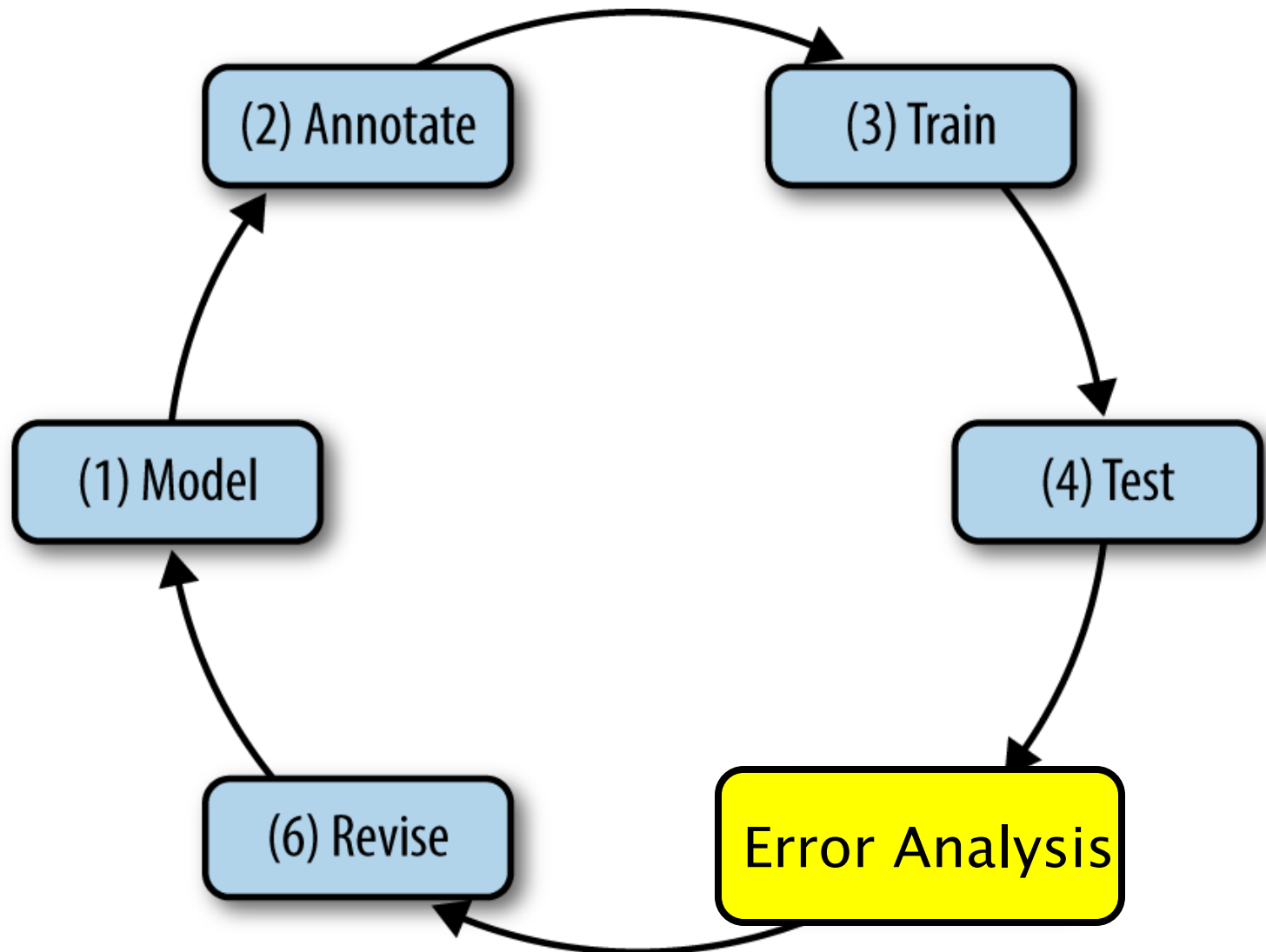
Neural Models

- Shallow NNs
 - Bag(words)
- Convolutional NNs
 - Handle bag (fixed length n-grams)
- Recurrent NNs
 - Handle small variable length histories
- LSTMs/GRUs
 - Handle larger variable length histories
- Bi-LSTMs
 - Handle larger variable length histories and futures
- Recursive NNs
 - Handle variable length partially ordered histories

Neural Models (contd)

- Neural language models
- Conditioned language models
 - Encoder-Decoder Models
- Attention models
 - attach non-uniform importance to histories based on evidence (question)
 - maybe even more important than recurrence?
- Transformer: all-word self attention + position embeddings
- Pre-trained language models
 - Fine-tuning
 - Light-weight finetuning: adapter, LoRA
 - In-context learning

Key Points: ML Cycle



Revise Tweaks

- Retrieval augmentation
- Use similar tasks; pretrain; then fine-tune few-shot
- Transfer learning; Multi-task learning
- Data augmentation
- Auxilliary loss
- Add human-features
- Add constraints
- Change architectures
- LLM workflows (agents)

The World of LLMs

- Pre-Training → Instruction Fine Tuning → Alignment
- Efficient Training: LoRA; Multi-GPU Training
- Efficient Inference: Quantization, Pruning, Distillation, Speculative Sampling
- Performance
 - Generation: Diversity-Quality tradeoff
 - Retrieval Augmentation: reduce hallucinations
 - Prompting techniques; Coding/other tools; LLMs as Idea Generators

Other Ideas: Scaling Laws

- Large Model, Small Data: Undertraining
- Small Model, Large Data: Underfitting
- Kaplan (2020) suggested power law b.w. performance & #params
 - People kept scaling parameters
- Given a computing budget, how can we trade off model size (number of parameters) and number of tokens (dataset size) to yield the best model possible?

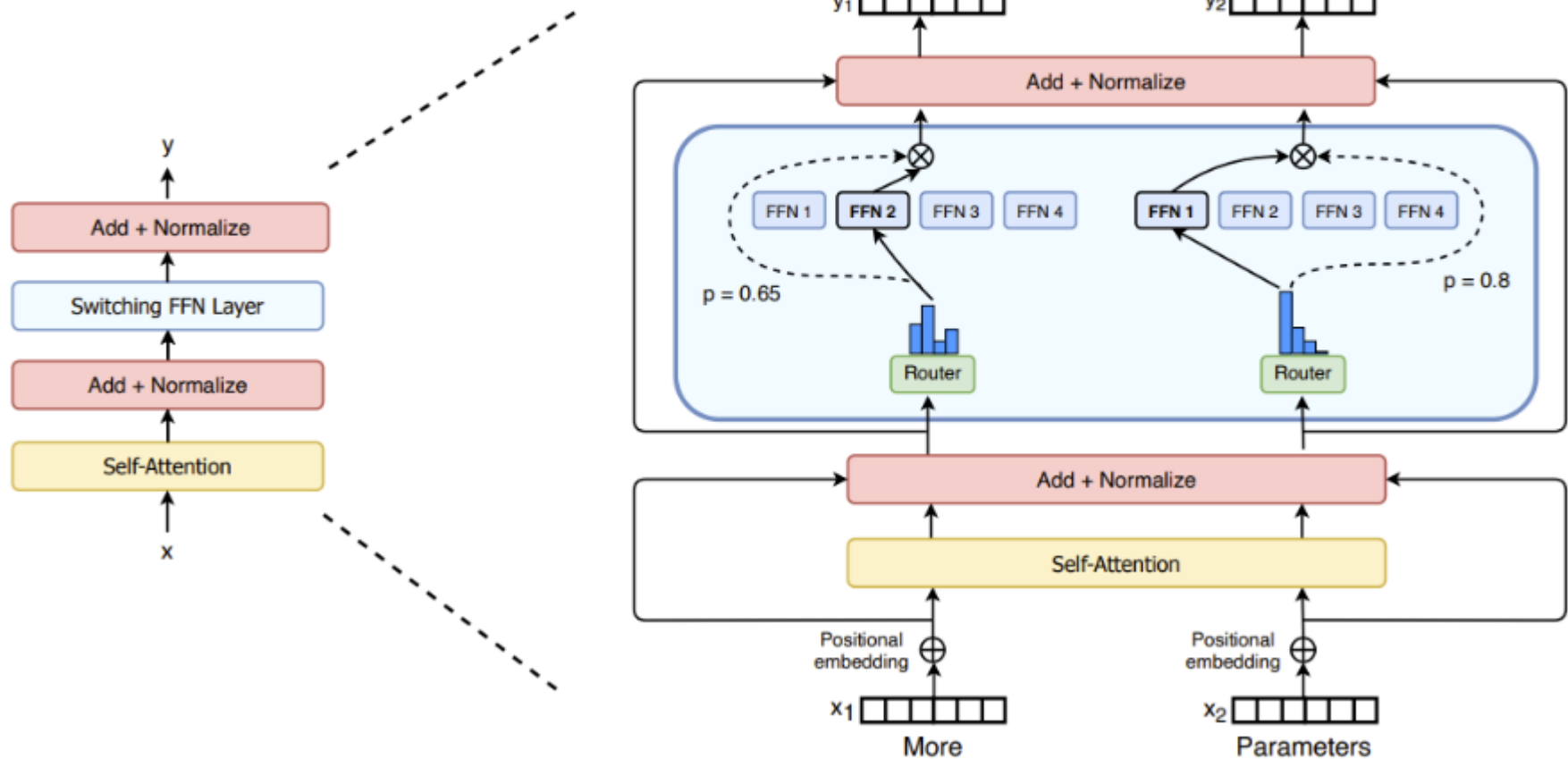
Other Ideas: Scaling Laws of LLMs (2022)

- Tradeoff: #parameters, dataset size, training budget
- DeepMind researchers trained over 400 LLMs, ranging from 70 million to over 16 billion parameters on 5 to 500 billion tokens.
- The striking conclusion is that model size, and the number of training tokens should scale equally!
 - Double the model size, double the number of tokens
 - Typically number of tokens = 20xnumber of parameters
- The scaling laws showed that GPT-3 was under-trained!
 - Chinchilla (4x smaller, trained with 4.6x more data) > GPT-3
 - GPT3.5

Llama3

- We made several new observations on scaling behavior during the development of Llama 3. For example, while the Chinchilla-optimal amount of training compute for an 8B parameter model corresponds to $\sim 200\text{B}$ tokens, we found that model performance continues to improve even after the model is trained on two orders of magnitude more data. Both our 8B and 70B parameter models continued to improve log-linearly after we trained them on up to 15T tokens. Larger models can match the performance of these smaller models with less training compute, but smaller models are generally preferred because they are much more efficient during inference.

Other Ideas: Mixture of Experts



MoE Benefits

- **Specialized experts:** Each expert can be trained to excel at particular tasks or subdomains within the overall problem space.
- **Increased efficiency:** Only the relevant experts are activated for each input, reducing computational overhead compared to a single monolithic model.
- **Scalability:** The MoE architecture can be easily scaled by adding more expert models, making it suitable for handling increasingly complex tasks.

Other Ideas: LLM Agents

