Da Estatística à IA Generativa: Construindo pontes entre inferência e algoritmos

Paulo Canas Rodrigues[1,2,3,4,5]

(Joint work with Rodrigo Bulhões, Jonatha Pimentel, and Ana Pinheiro)

[1]Department of Statistics, Federal University of Bahia, Brazil

[2]Diretor of the Statistical Learning Laboratory (SaLLy)

[3]ISBIS President [4]IASC President-Elect [5]ASA Educational Ambassador

UFBA Universidade Federal da Bahia

Departamento de Estatística IME - UFBA

Statistical Learning Laboratory SaLLy

Paulo Canas Rodrigues

Email: paulocanas@gmail.com
Web: www.paulocanas.org
Lab: www.SaLLy.ufba.br

- BSc in Mathematics (FCT-UNL, Portugal; 2003), MSc in Statistics (IST-UTL, Portugal; 2007), PhD in Statistics (FCT-UNL, Portugal; 2012), Habilitation in Mathematics, specialization in Statistics and Stochastic Processes (IST-UL, Portugal; 2019).
- Published more than 115 scientific papers in collaboration with more than 175 co-authors from 83 universities in 30 countries and delivered more than 180 invited talks and seminars.

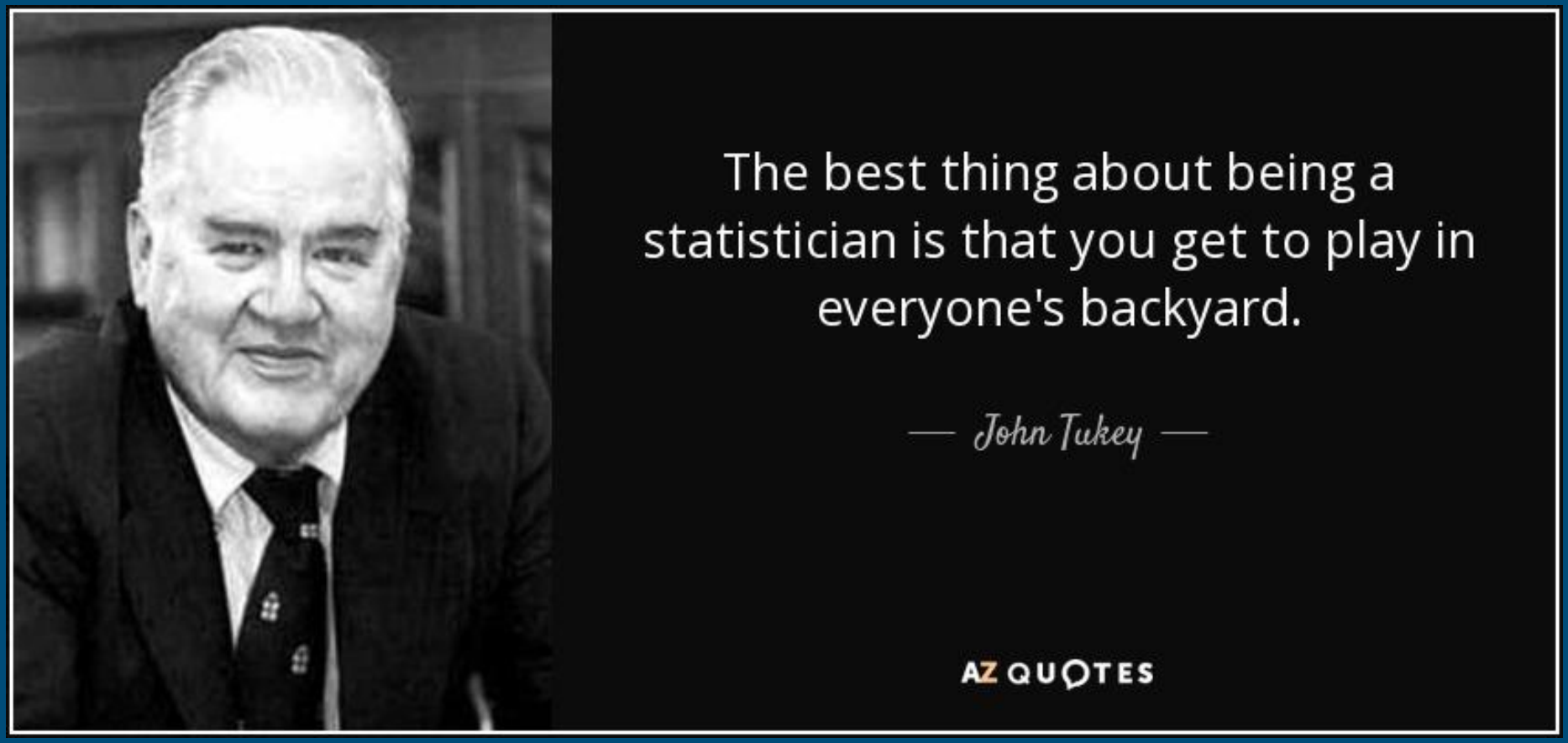
- Among other activities, Paulo Canas Rodrigues is currently:
  - Professor of the Department of Statistics, Federal University of Bahia, Brazil
  - Director of the Statistical Learning Laboratory (SaLLy)
  - 2024 ASA Educational Ambassador
  - Co-founder and Vice-Coordinator of the Specialization in Data Science and Big Data (since 2018)
  - Coordinator of a successful proposal for the creation of an MSc and PhD programs in Statistics and Data Science (2023)
  - Co-Editor for Computational Statistics, Brazilian Journal of Biometrics, Biometrical Letters, and Statistics, Optimization and Information Computing
  - President of the International Society for Business and Industrial Statistics (2023 –2025)
  - President-Elect of the International Association for Statistical Computing (10/2023 – 10/2025)
  - Member of the Representative Council of the International Biometric Society (07/2021 – 07/2029)
  - Council Member of the International Statistical Institute (07/2023 – 07/2025)

- **Research interests**: Time series forecasting, Statistical learning, Artificial intelligence, Data science, Singular spectrum analysis, Robust statistics, Applications to plant genetics, finances, health, and environmental sciences.

Outline

1. Why Statistics?
2. What are Neural Networks?
3. What are Large Language Models?
4. Pretraining and Transfer Learning in LLMs
5. Inference in LLMs
   - Alignment
   - Watermarking
6. Future directions

Being a Statistician



Why Statistics?

Science

AAAS

Science 6 April 2012:

Vol. 336 no. 6077 p. 12

Editorial

Marie Davidian
(North Carolina State University)

Thomas Louis
(Johns Hopkins Bloomberg School of Public Health)

Why Statistics?

Statistics is the science of learning from data, and of measuring, controlling, and communicating uncertainty; and it thereby provides the navigation essential for controlling the course of scientific and societal Advances.

50 Years of Data Science

JOURNAL OF COMPUTATIONAL AND GRAPHICAL STATISTICS
2017, VOL. 26, NO. 4, 745–766
https://doi.org/10.1080/10618600.2017.1384734

50 Years of Data Science

David Donoho

Department of Statistics, Stanford University, Standford, CA

ABSTRACT

More than 50 years ago, John Tukey called for a reformation of academic statistics. In "The Future of Data Analysis," he pointed to the existence of an as-yet unrecognized *science*, whose subject of interest was learning from data, or "data analysis." Ten to 20 years ago, John Chambers, Jeff Wu, Bill Cleveland, and Leo Breiman independently once again urged academic statistics to expand its boundaries beyond the classical domain of theoretical statistics; Chambers called for more emphasis on data preparation and presentation rather than statistical modeling; and Breiman called for emphasis on prediction rather than inference. Cleveland and Wu even suggested the catchy name "data science" for this envisioned field. A recent and growing phenomenon has been the emergence of "data science" programs at major universities, including UC Berkeley, NYU, MIT, and most prominently, the University of Michigan, which in September 2015 announced a $100M "Data Science Initiative" that aims to hire 35 new faculty. Teaching in these new programs has significant overlap in curricular subject matter with traditional statistics courses; yet many academic statisticians perceive the new programs as "cultural appropriation." This article reviews some ingredients of the current "data science moment," including recent commentary about data science in the popular media, and about how/whether data science is really different from statistics. The now-contemplated field of data science amounts to a superset of the fields of statistics and machine learning, which adds some technology for "scaling up" to "big data." This chosen superset is motivated by commercial rather than intellectual developments. Choosing in this way is likely to miss out on the really important intellectual event

ARTICLE HISTORY
Received August 2017
Revised August 2017

KEYWORDS
Cross-study analysis; Data analysis; Data science; Meta analysis; Predictive modeling; Quantitative programming environments; Statistics

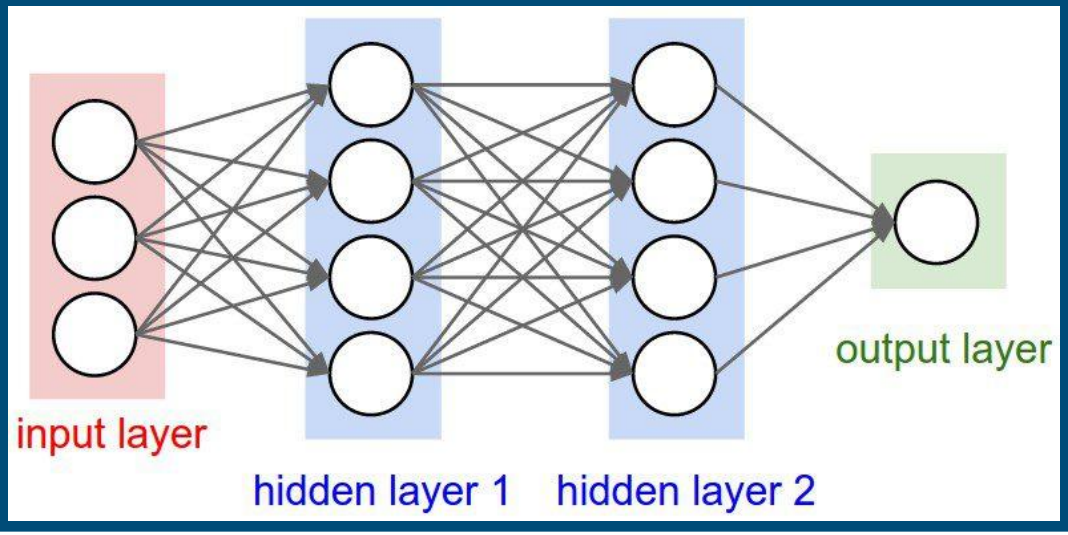
Outline

1. Why Statistics?
2. What are Neural Networks?
3. What are Large Language Models?
4. Pretraining and Transfer Learning in LLMs
5. Inference in LLMs
   - Alignment
   - Watermarking
6. Future directions

What is a Neural Network?

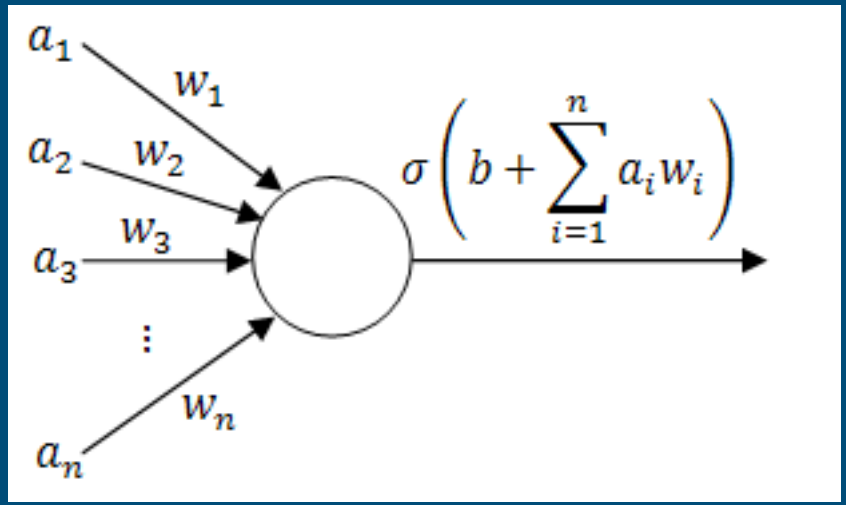
- Neural networks are computational models inspired by the structure of biological brains.
- Architecture:
  - Input layer: receives raw features (e.g., pixels, words, measurements).
  - Hidden layers: transform and extract patterns via learned weights and non-linearities.
  - Output layer: generates predictions (e.g., class probabilities, numeric values).
- Each neuron performs a simple operation: a weighted sum of inputs plus a bias, followed by an activation function.

- Each connection is associated with a weight.
- Training of a neural network is to get to the right weights (and biases) such that the error across the training data is minimized.



What is a Neural Network?

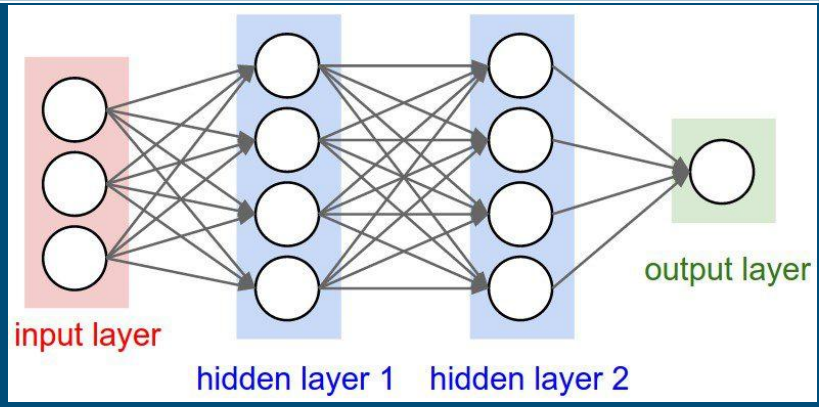
- The input nodes are represented by ‘a’.
- Neurons expect incoming connections from other nodes, so even the network’s inputs must be modelled as such.



- Each of the connections from the input layer is passed to a single node, which will optimize its weights based on how it’s trained. The results of this node are sent to the output function, or activation function, responsible for mapping the network’s results to a meaningful value. That fact indicates this one node is also our output layer.
- In this example, our activation function is the σ function, so our network will output a number between 0 and 1. This is a common output for classification networks.

Multilayer Perceptrons (MLPs)

- A Multilayer Perceptron consists of:
  - One input layer.
  - One or more hidden layers.
  - One output layer.



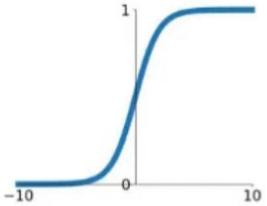
- Each neuron in a layer is connected to all neurons in the previous layer (fully connected).
- By stacking layers and using non-linear activation functions, MLPs can approximate complex functions.
- Enables hierarchical representation of data:
  - Lower layers may learn simple features (e.g., edges in images).
  - Higher layers combine them into more abstract concepts (e.g., faces, objects).

Activation Functions – Why and What?

- Activation functions introduce non-linearity, enabling the network to model complex relationships.
- Without non-linear activations, a neural network behaves like a linear model, regardless of depth.

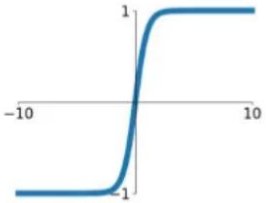
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



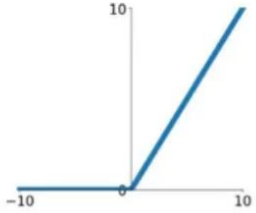
tanh

$$\tanh(x)$$



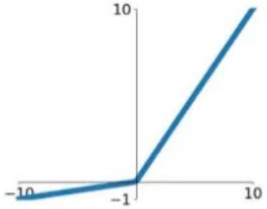
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

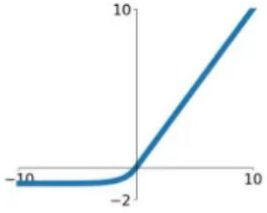


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Forward Propagation: How Information Flows

- Forward Propagation – How It Works
  - Inputs are passed through the network layer by layer.
  - Each neuron computes a weighted sum of its inputs and applies an activation function.
  - Output of one layer becomes input for the next.
  - Final layer produces prediction (e.g., class probabilities, regression output).
- Each layer receives inputs from the previous layer, computes activations, and passes them forward.
- For each neuron:
  - $z = \boldsymbol{w} \cdot \boldsymbol{x} + b, \qquad a = activation(z).$
- Forward pass:
  - Input features are transformed layer by layer.
  - The final output is produced and compared to the target during training.
- Can be interpreted as a **compositional function**: $f(x) = f_3(f_2(f_1(x)))$

Loss Functions: Measuring Performance

- The loss function quantifies how well predictions match the ground truth.
- It is the objective we want to minimize during training.
- Common loss functions:
  - Mean Squared Error (MSE) – regression: $$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$
  - Cross-Entropy – classification: $$\mathcal{L} = -\sum_{i=1}^{n} y_i \log(\hat{y}_i)$$
- The choice of loss depends on the task and output type.

Backpropagation: Learning via Gradients

- Backpropagation computes the gradient of the loss function with respect to each weight in the network.
- Uses the chain rule to propagate error from output to input layers.
- Combined with an optimizer (e.g., SGD, Adam), it allows the model to update weights and improve predictions.
- **Steps**:
  - Forward pass: compute predictions.
  - Compute loss.
  - Backward pass: compute gradients.
  - Update weights.

Training: Parameters vs. Hyperparameters

- Model Parameters (learned during training):
  - Weights: determine how inputs are combined.
  - Biases: allow the model to shift the activation threshold.
  - These are updated during backpropagation to minimize the loss.

- Hyperparameters (defined before training):
  - Learning rate: step size in gradient descent — too high can diverge, too low slows learning.
  - Number of epochs: how many times the full dataset is passed through the model.
  - Batch size: number of samples processed before updating parameters.
  - Number of layers and neurons per layer: affect model capacity and generalization.
  - Optimizer: algorithm used to update weights (e.g., SGD, Adam, RMSProp).

- Key Insight:
  - Hyperparameters control the learning process and have a big impact on performance.
  - Often tuned via grid search, random search, or Bayesian optimization.

Outline

1. Why Statistics?
2. What are Neural Networks?
3. What are Large Language Models?
4. Pretraining and Transfer Learning in LLMs
5. Inference in LLMs
   - Alignment
   - Watermarking
6. Future directions

What Are Large Language Models (LLMs)?

- Definition: LLMs are advanced artificial intelligence models trained to understand and generate human-like text.
- Foundation: They rely on deep learning techniques, particularly transformer-based architectures, to analyse and produce coherent textual outputs.
- Key Characteristics:
  - Pretrained on vast datasets (books, websites, research papers, dialogues, etc.).
  - Can perform various natural language processing (NLP) tasks with minimal task-specific training.
  - Generate text, translate languages, summarize content, and assist in code generation.

What Are Large Language Models (LLMs)?

- How do they Work?
  - Use tokenization to process text into manageable units.
  - Employ attention mechanisms (self-attention) to understand contextual relationships between words.
  - Utilize massive neural networks (often billions of parameters) to refine and generate responses.
- Historical Development:
  - Early NLP models: rule-based systems, statistical models (n-grams, Hidden Markov Models).
  - Shift to neural networks: Word2Vec, GloVe, RNNs, LSTMs.
  - Introduction of transformers (2017): "Attention Is All You Need" paper revolutionized NLP.
  - Modern LLMs: BERT, GPT-3, GPT-4, T5, PaLM, LLaMA, Claude.

Why Are LLMs Important?

- Revolutionized Natural Language Processing (NLP): LLMs have dramatically improved machine understanding and generation of human text.
- Handling Large-Scale Text Data: They are trained on vast corpora, allowing them to generate high-quality text outputs.
- Broad Applicability: Used in diverse fields, including healthcare (medical text analysis), finance (automated report generation), education (personalized tutoring), and creative industries (content generation, storytelling).
- Key Developments: LLMs enable cutting-edge AI applications such as:
  - Conversational AI (ChatGPT, Bard, Claude)
  - Code Assistance (GitHub Copilot, OpenAI Codex)
  - Search Engine Enhancement (Google Search with AI, Bing AI)
- Advancing AI Research: LLMs contribute to developments in AI reasoning, planning, and multimodal capabilities (text, images, audio).

LLM: Key Terminology

- Natural Language: AI field focused on human language, including tasks like translation, speech recognition, and sentiment analysis.
- Deep Learning: A subset of machine learning using multi-layered neural networks for feature extraction and learning from large datasets.
- Transformers: A deep learning architecture that relies on self-attention mechanisms, enabling state-of-the-art performance in NLP tasks.
- Pretraining & Fine-Tuning:
  - Pretraining: LLMs learn general language patterns from vast datasets before task-specific adaptation.
  - Fine-tuning: Additional training on specialized datasets to improve performance in specific applications.

LLM: Key Terminology

- Inference: The process of generating text outputs from a trained model, involving strategies like beam search, top-k sampling, and nucleus sampling.
- Tokenization: Breaking text into smaller units (tokens) for model processing.
- Embedding: Representing words, phrases, or sentences in continuous vector space for neural network processing.
- Attention Mechanism: Core component of transformers that allows models to focus on different parts of the input sequence dynamically.
- Reinforcement Learning with Human Feedback (RLHF): A fine-tuning process using human annotations to align LLM outputs with user preferences and ethical considerations.

Real-World Applications of LLMs

- Chatbots & Virtual Assistants
  - Examples: ChatGPT, Siri, Google Assistant, Alexa.
  - Uses: Customer support, virtual companions, task automation.
- Machine Translation
  - Examples: Google Translate, DeepL, Microsoft Translator.
  - Improvements: Context-aware translations, reduced grammatical errors.
- Text Summarization
  - Types: Extractive vs. abstractive summarization.
  - Applications: News article condensation, legal document processing, academic paper summarization.
- Sentiment Analysis
  - Use Cases: Brand reputation monitoring, customer feedback interpretation, stock market sentiment prediction.
  - Techniques: LLMs classify emotions (positive, negative, neutral) and detect nuanced sentiments.

Real-World Applications of LLMs

- Code Generation
  - Examples: GitHub Copilot, OpenAI Codex, Code Llama.
  - Capabilities: Auto-completing code, generating functions, debugging, translating between programming languages.
- Healthcare Applications
  - Medical Chatbots: AI-driven assistants for patient inquiries.
  - Research Support: Analysing medical literature, summarizing studies.
  - Diagnosis Assistance: Supporting doctors by identifying potential conditions from patient symptoms.
- Education & Personalized Learning
  - AI Tutors: Adaptive learning experiences tailored to student needs.
  - Automated Grading: AI-driven evaluation of essays and assignments.
  - Language Learning: AI-based exercises, real-time feedback.
- Content Generation
  - Blogging, creative writing, automated journalism.
  - Generating marketing materials, ad copy, and SEO-optimized articles.

Challenges & Limitations of LLMs

- Bias & Fairness
  - Training data may contain social, political, and cultural biases.
  - Example: Discriminatory language, reinforcement of stereotypes.
  - Mitigation Strategies: Dataset curation, adversarial training, fairness-aware algorithms.
- Computational Cost & Environmental Impact
  - Training LLMs requires vast GPU clusters, consuming high energy.
  - Example: Training GPT-3 emitted as much CO_2 as 125 round-trip flights between New York and Beijing.
  - Solutions: Model distillation, pruning, quantization, using energy-efficient architectures.
- Interpretability & Explainability
  - LLMs operate as "black boxes" – difficult to trace reasoning behind outputs.
  - Challenge: Trust and accountability in high-stakes applications (e.g., legal, medical).
  - Research Directions: Explainable AI (XAI), feature visualization, attention heatmaps.

Challenges & Limitations of LLMs

- Hallucinations & Misinformation
  - LLMs sometimes generate plausible-sounding but factually incorrect information.
  - Example: AI fabricating citations or generating false legal precedents.
  - Mitigation Strategies: Retrieval-augmented generation (RAG), fact-checking pipelines.

- Ethical Concerns
  - Misinformation: AI-generated deepfakes, propaganda, spam bots.
  - Privacy Risks: Leakage of sensitive information from training data.
  - Copyright & Plagiarism: Unclear ownership of AI-generated content.
  - Solutions: Legal regulations, watermarking AI outputs, synthetic data for privacy preservation.

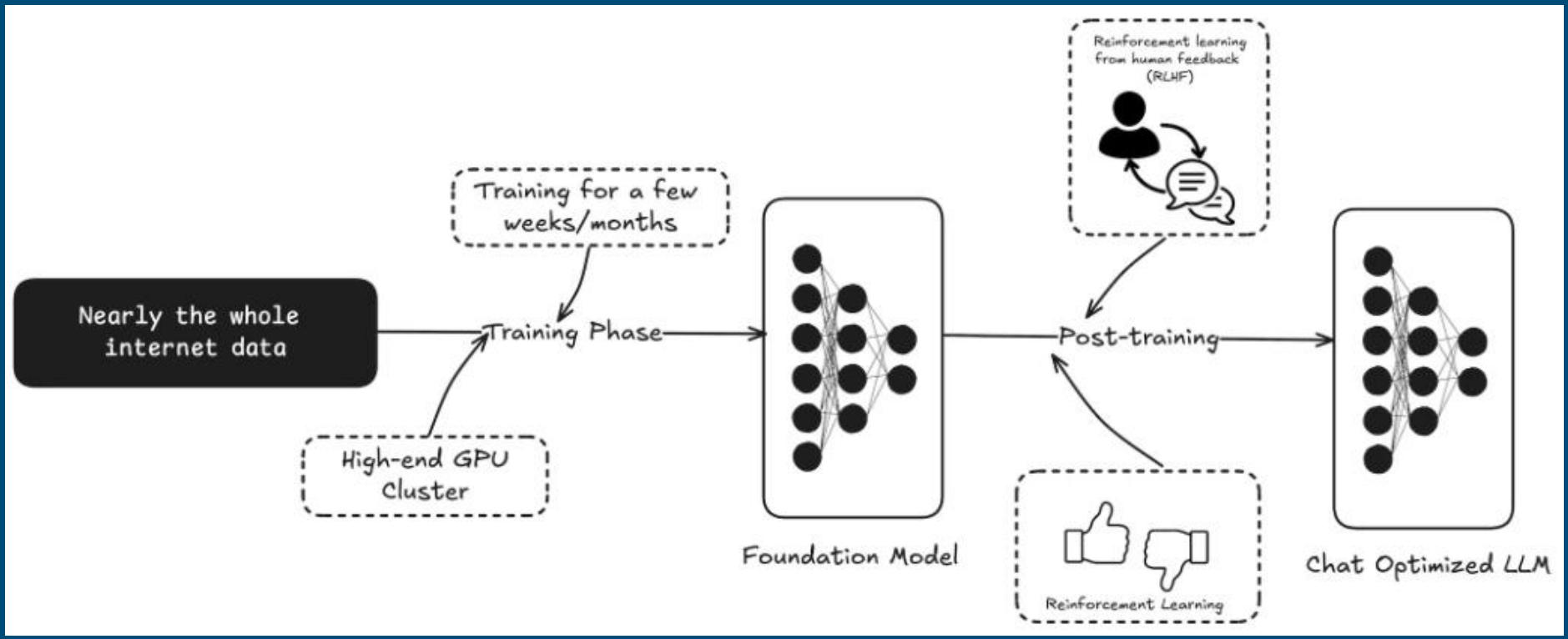
Outline

1. Why Statistics?
2. What are Neural Networks?
3. What are Large Language Models?
4. Pretraining and Transfer Learning in LLMs
5. Inference in LLMs
   - Alignment
   - Watermarking
6. Future directions

What is Pretraining?

- **Definition**: Pretraining is the process of training a large neural network on massive, unlabelled text data using self-supervised learning.
- **In practice (LLM context)**:
  - The model starts with random weights and is exposed to trillions of tokens from diverse sources (e.g., books, websites, code).
  - It gradually learns linguistic structure, syntax, semantics, and common-sense knowledge by predicting missing or next words.
- **Outcome**:
  - The model becomes fluent in language but not yet aligned with specific tasks or user intentions.
  - In its raw form, it is an autoregressive generator — a foundational model.
- **Next steps**:
  - To make it follow instructions (e.g., summarize, translate, classify), we apply fine-tuning, more specifically, instruction tuning.

What is Pretraining?



Source: https://codimite.ai/blog/pre-training-finetuning-and-rag-how-to-choose-the-right-one-for-you/

Pretraining Objectives

- **Autoregressive (e.g., GPT)**:
  - Predict next token: $P(w_t|w_1, \ldots, w_{t-1})$.
  - Only the left context is used; Excellent for generation.
  - Example:
    - Input: "The cat sat on the“
    - Target: "mat“
    - Prediction proceeds left-to-right.
    - Used in GPT, GPT-2, GPT-3.

- **Masked Language Modelling (e.g., BERT)**:
  - Predict masked tokens in the input: $P(w_t|w_{\setminus t})$.
  - Full bidirectional context; Better for classification, understanding.
  - Example:
    - Input: "The [MASK] sat on the mat.“
    - Target: "cat“
    - Model uses both left and right context.
    - Used in BERT, RoBERTa, ALBERT.

Pretraining Datasets

- Commonly used corpora:
  - **WebText / Common Crawl** – billions of web pages.
  - **Wikipedia** – encyclopedic style.
  - **BooksCorpus** – narrative and long-form language.
  - **C4** – curated filtered web crawl (used in T5)
- Scale:
  - 100GB to several TB of text.
- Example (GPT-3):
  - 300B tokens.
  - 175B parameters.
  - Data sources: WebText2, Books1+2, Wikipedia, Common Crawl.
  - No explicit labels.

Transition – From Pretraining to Transfer Learning

- Up to now, we explored how LLMs are pretrained: massive data, self-supervised learning, and foundational models.
- But pretraining alone is not enough to solve specific problems.
- We need to adapt these models to:
  - Follow instructions.
  - Align with new domains.
  - Solve practical tasks (e.g., classification, summarization, dialogue).
- Key Question: How can we move from a general-purpose LLM to a task-specialized or domain-specific system?
- ➡️ Transfer Learning.

Transfer Learning – Key Idea

- Definition: Using the knowledge gained during pretraining on one task to improve learning and performance on a different but related task.
- In LLMs:
  - The base model is trained to understand general language.
  - Transfer learning enables adaptation to new, often more specific domains or tasks.
  - Examples: legal document summarization, customer support, code generation.
- Stages:
  1) Pretraining on massive corpora (e.g., GPT-3 on Common Crawl).
  2) Optional: intermediate tuning on domain-specific text.
  3) Task-specific fine-tuning with supervised data.
- Benefits:
  - Saves computation and time.
  - Reduces labelled data requirements.
  - Enables domain-specific performance without retraining from scratch.
  - Foundation model → downstream tasks.
  - Efficient, scalable, low-data regimes

Transfer Learning Workflow – Step-by-Step

- Step 1: Choose a pretrained model.
  - Example: bert-base-uncased or gpt2-medium.
- Step 2: Prepare your dataset.
  - Format inputs and labels for the task (e.g., classification, QA, summarization).
  - Tokenize using the same tokenizer as the base model.
- Step 3: Add a task-specific head.
  - Classification layer (e.g., softmax).
  - Seq2seq decoder for translation/summarization.
- Step 4: Fine-tune the model.
  - Use supervised training with your dataset.
  - Monitor loss, accuracy, and F1-score.
- Summary:
  - [Pretrained LLM] → [Domain-specific data] → [Fine-tuned Model for Task]
  - Pretraining → Intermediate Tuning → Task-specific Fine-tuning

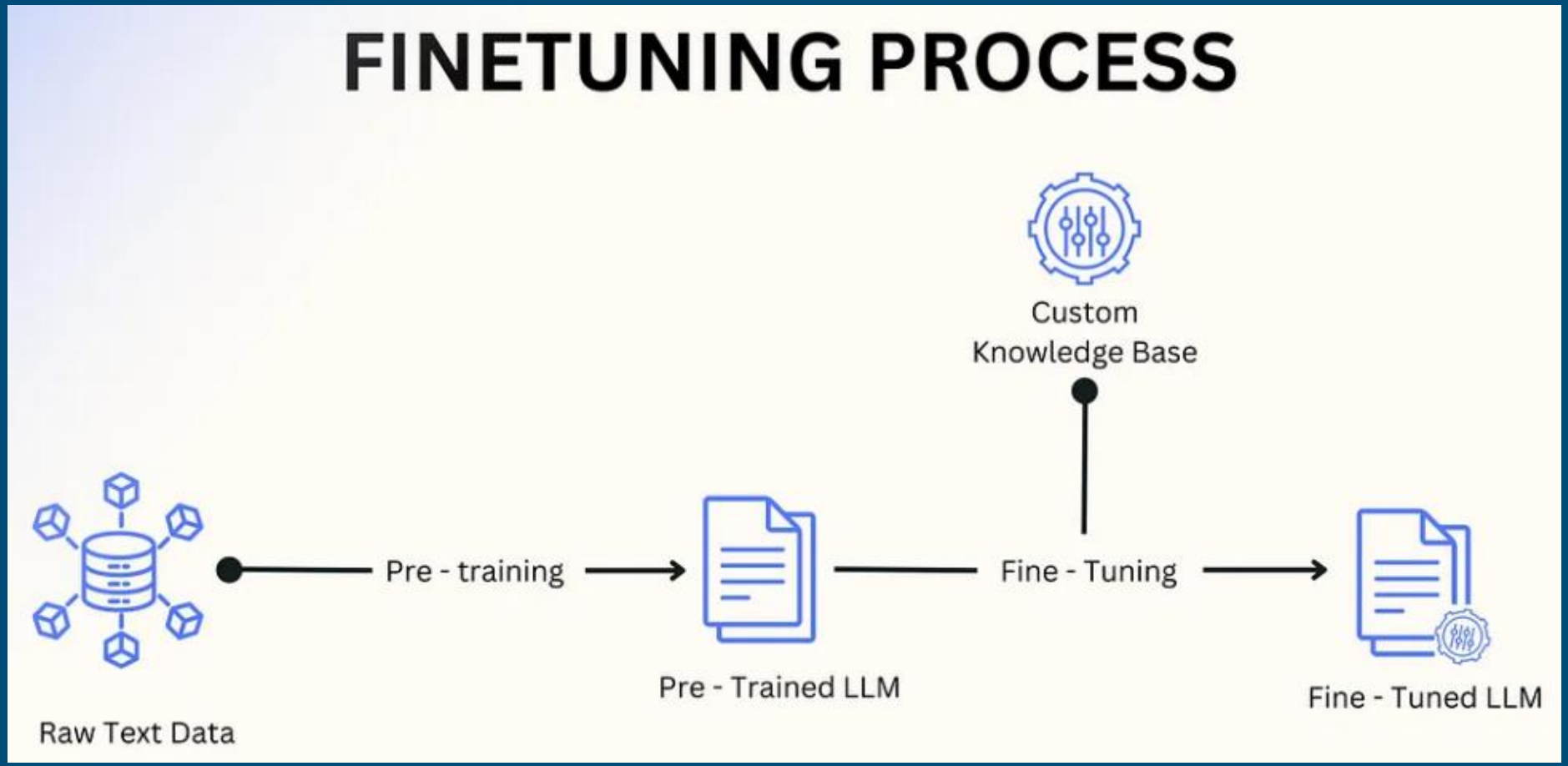
What is Fine-Tuning?

- Finetuning adapts a pretrained LLM to a specific domain or task.
- It can start from:
  - A foundational model (e.g., GPT-3).
  - An instruct-tuned model (e.g., InstructGPT).
- In both cases, the finetuned model inherits general language capabilities from the base.
- Unlike initial pretraining, this phase starts with learned weights.
- Requires: a pretrained model and domain-specific training data (e.g., Q&A pairs).
- Contrast:
  - Pretraining = general knowledge.
  - Fine-tuning = task grounding.

Finetuning Challenges

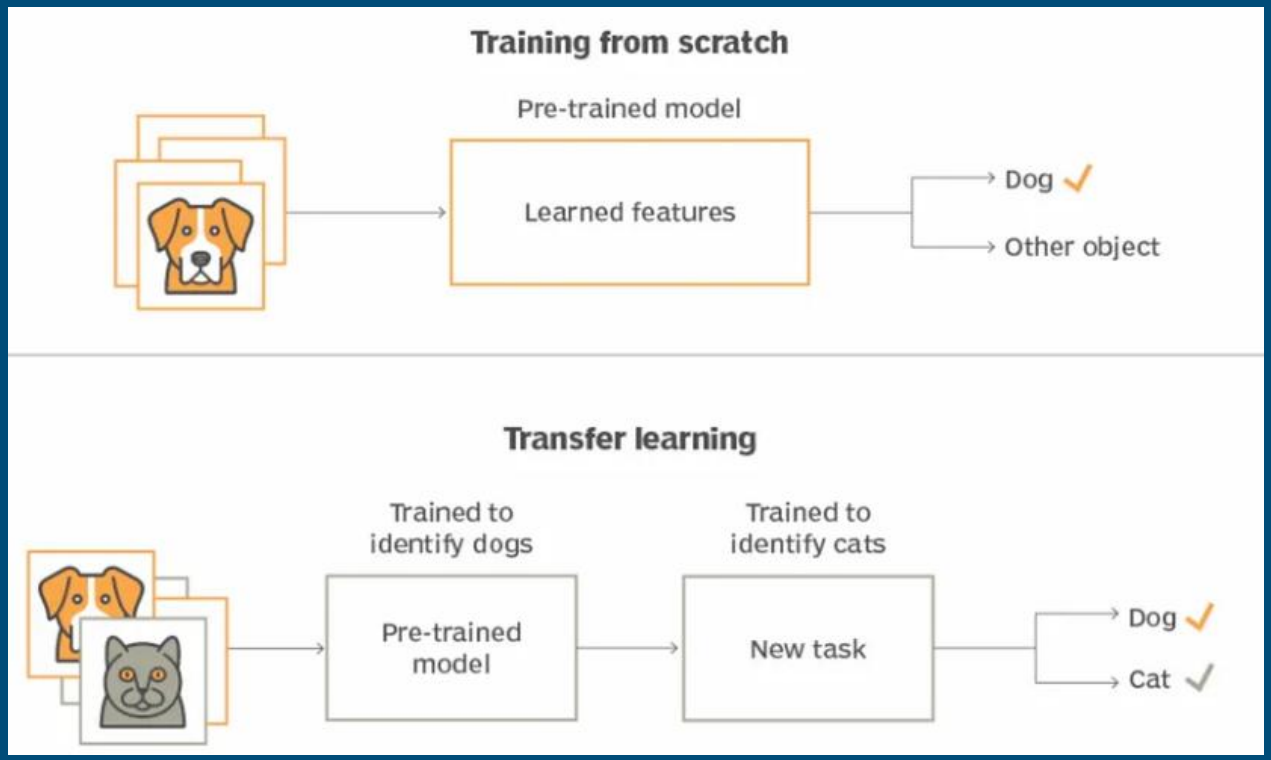
- However, there are a few challenges to consider:
  - **Data requirements**: High-quality domain-specific datasets are needed.
  - **Domain evolution**: As your domain grows (e.g., adding new products), retraining may be required.
  - **Hallucinations**: Model may generate fluent but factually incorrect or misleading information, especially in low-data or underrepresented scenarios.
  - **Overfitting**: Model memorizes a small dataset.
  - **Bias transfer**: Inherited biases from pretraining or data.
- In a business context, these hallucinations can produce nonsensical or misleading responses.
- Can we mitigate this?
  - ✅ Yes — **Retrieval-Augmented Generation (RAG)** is one effective solution.

Finetuning Process



Source: https://www.tagxdata.com/what-is-supervised-fine-tuning

How transfer learning works



- In the GenAI Era, transfer learning is baked into the system. These models are inherently general-purpose and adapting them to a new task often requires nothing more than a well-crafted prompt. Fine-tuning may not even be necessary.
- Example: Instead of fine-tuning GPT-4 to draft legal contracts, you can guide it with a few example prompts, leveraging its pre-trained knowledge dynamically.

Source: https://medium.com/logspace/transfer-learning-pre-training-or-fine-tuning-fc1fa6c7801e

Challenges in Transfer Learning

- Technical and practical limitations when adapting LLMs to new domains:
  - **Catastrophic forgetting:** Finetuning may overwrite general knowledge.
  - **Overfitting:** Small datasets lead to poor generalization.
  - **Data shift:** Mismatch between pretraining and task domain (e.g., legal, medical).
  - **Bias transfer:** Pretraining biases may persist or be amplified.
  - **Hallucinations:** The model generates fluent but inaccurate or misleading content.
- Note: Retrieval-augmented generation (RAG) and careful data curation can help mitigate some of these issues.
  - Catastrophic forgetting.
  - Overfitting in low-data scenarios.
  - Data shift.
  - Inherited biases.

Outline

1. Why Statistics?
2. What are Neural Networks?
3. What are Large Language Models?
4. Pretraining and Transfer Learning in LLMs
5. Inference in LLMs
   - Alignment
   - Watermarking
6. Future directions

Introduction to Statistical Inference

- Definition: Statistical inference involves using data analysis to deduce properties of an underlying probability distribution, enabling researchers to draw conclusions about larger populations from observed samples.
- Purpose: To make reliable predictions and informed decisions based on incomplete or sampled data.
- Relevance to LLMs: Essential for assessing model performance, interpreting predictions, and quantifying uncertainty, which is crucial for practical decision-making.
- Example: Estimating the true average probability of a token prediction across all possible data (i.e, every possible input, sentence, phrase, or word combination the model could encounter) vs. calculating the mean probability from observed predictions (i.e., a manageable subset of input data).

Statistical Inference

- Point estimation is the process of providing a single best guess (a “point”) for an unknown population parameter based on observed sample data. Unlike interval estimation, which gives a range of plausible values, point estimation produces a precise numerical estimate of the parameter.

- A confidence interval (CI) is a range of values, derived from sample data, that is likely to contain the true value of an unknown population parameter with a specified probability. Instead of giving a single estimate (a point estimate), a confidence interval provides a plausible range, helping quantify uncertainty in statistical estimation.

- Hypothesis testing is a structured statistical procedure that allows researchers to evaluate assumptions or claims about population parameters using sample data. The primary goal is to objectively determine whether observed sample results provide sufficient evidence to support a particular claim about the broader population.

Point Estimation in LLM

- Point estimation is essential for quantifying features of model behaviour from sampled outputs or evaluations. Examples:

  - 1. Token Probability Estimation
    - Scenario: Estimate the average probability assigned to a specific token (e.g., “excellent”) across multiple prompts.
    - Why it matters: Helps understand model confidence and variability in word choice under different contexts (i.e., surrounding words, sentences, or prompts that help the model understand and predict the next token or generate coherent text).
    - Example: Prompt an LLM with 100 distinct sentences and compute the mean predicted probability of the word “excellent” across all cases.
  - 2. Estimating Model Performance Metrics
    - Scenario: Estimate accuracy, perplexity, or BLEU (Bilingual Evaluation Understudy; measures how similar the machine-generated text is to one or more reference texts written by humans) score from evaluation data.
    - Why it matters: Provides quantitative, comparable assessments of model quality across tasks, datasets, or model versions.
    - Example: Use a validation dataset to compute the average BLEU score of summaries produced by a fine-tuned LLM.
  - 3. Tracking Learning Progress During Training
    - Scenario: Estimate loss or accuracy at each training epoch.
    - Why it matters: Enables monitoring of model convergence, stability, and early stopping criteria.
    - Example: Plot mean validation loss over 20 epochs to identify overfitting in a fine-tuned GPT model.

Point Estimation in LLM

- 4. Fine-tuning Performance Evaluation
  - Scenario: Estimate performance change (e.g., accuracy, loss) before and after fine-tuning on a downstream task.
  - Why it matters: Measures how much a model adapts to new domains or tasks.
  - Example: Fine-tune on legal documents and estimate post-training perplexity vs. base model perplexity on legal texts.
- 5. Token Usage Analysis for Fairness and Bias Audits
  - Scenario: Estimate average frequency or probability of gendered or sensitive terms in generated text.
  - Why it matters: Identifies unintended biases or disparities in model outputs.
  - Example: Calculate the average probability assigned to male vs. female pronouns in occupation-related prompts.
- 6. Prompt Engineering Evaluation
  - Scenario: Estimate how often a certain prompt style leads to desired output structure.
  - Why it matters: Quantifies the effect of prompt design choices and helps select more effective formulations.
  - Example: Estimate the proportion of completions that follow a bullet-point format when using the instruction "List three reasons...".
- 7. Parameter Selection and Hyperparameter Tuning
  - Scenario: Estimate the average performance (e.g., loss) under different hyperparameter configurations.
  - Why it matters: Informs the choice of learning rates, batch sizes, and model sizes to maximize training efficiency.
  - Example: Run experiments with three different learning rates and compute mean validation accuracy for each.

Point Estimation in LLM

- 8. Output Calibration and Thresholding
  - Scenario: Estimate the average of maximum token probabilities to assess model overconfidence.
  - Why it matters: Helps calibrate decision thresholds or apply entropy-based rejection policies.
  - Example: Estimate the average top-1 token probability over 1,000 samples to assess calibration quality.
- 9. A/B Testing for Model Variants
  - Scenario: Estimate and compare average user ratings or success rates for two model versions.
  - Why it matters: Supports data-driven deployment decisions in production environments.
  - Example: Estimate the average helpfulness score from user feedback for responses generated by GPT-3.5 vs. GPT-4.
- 10. Embedding Similarity Estimation
  - Scenario: Estimate cosine similarity between LLM-generated outputs and reference embeddings.
  - Why it matters: Enables semantic evaluation of output quality beyond surface form.
  - Example: Compute average cosine similarity between generated answers and ground-truth answers across a QA dataset.

- These examples illustrate how point estimation serves as a foundational tool for quantifying model behaviour, guiding evaluation, and supporting informed decisions throughout the development, fine-tuning, and deployment of large language models.

Confidence Intervals in LLM

- Confidence intervals are of key importance for quantifying uncertainty of model behaviour from sampled outputs or evaluations. Examples:

  - 1. Quantifying Uncertainty in Token Probabilities
    - Scenario: Construct confidence intervals around the predicted probability of a specific token across multiple prompts.
    - Why it matters: Distinguishes between confidently and uncertainly predicted tokens, helping in model interpretability.
    - Example: After prompting the model 100 times, compute a 95% confidence interval for the probability assigned to the token “excellent.”
  - 2. Evaluating Stability of Model Accuracy
    - Scenario: Report a confidence interval around the estimated classification accuracy of the model on a validation set.
    - Why it matters: Communicates how much the estimated performance might vary with different data samples.
    - Example: Suppose your LLM is fine-tuned to perform binary sentiment classification (positive vs. negative). You evaluate it on a test set of 1,000 examples, and it correctly classifies 840 of them. Accuracy = 84%, with a 95% CI of [81.2%, 86.8%], indicating the range where the true accuracy likely lies.
  - 3. Comparing Two Model Versions
    - Scenario: Estimate the difference in BLEU scores between a base model and a fine-tuned model, with a confidence interval.
    - Why it matters: Assesses whether observed differences are statistically meaningful or within the margin of variability.
    - Example: BLEU improvement = 2.3 points; 95% CI = [0.7, 3.9] → a likely real improvement.

Confidence Intervals in LLM

- 4. Interpreting Human Evaluation Metrics
  - Scenario: Annotators rate LLM outputs on coherence or factuality on a scale of 1 to 5; compute confidence intervals for the average scores.
  - Why it matters: Accounts for subjectivity and variability in human judgments.
  - Example: Let's say you generate 100 outputs from your LLM and have 3 human raters score each one for coherence on a scale from 1 to 5. Mean coherence score = 4.1/5, 95% CI = [3.9, 4.3].
- 5. Calibrating Model Risk in Safety-Critical Applications
  - Scenario: Use CIs to assess the upper bound of harmful content generation rates.
  - Why it matters: Ensures robust risk estimates in safety-critical deployments.
  - Example: You run a toxicity audit on 1,000 generated outputs using an external classifier (e.g., Perspective API), and detect toxic content in 8 of them. Estimated toxicity rate = 0.8%, 95% CI = [0.4%, 1.2%].
- 6. Measuring Robustness to Prompt Variations
  - Scenario: Compute the CI of a performance metric (e.g., accuracy) under multiple paraphrased prompts.
  - Why it matters: Evaluates how sensitive the model is to prompt phrasing.
  - Example: You evaluate an LLM on a QA task using three paraphrased prompt versions, each used for 200 questions. You calculate accuracy and 95% confidence intervals for each. Accuracy CI = [78.5%, 84.3%].
- 7. Confidence Intervals for Embedding Similarities
  - Scenario: Estimate average cosine similarity between generated answers and references, with CI, to quantify how semantically similar those outputs are to a reference text (e.g., ground-truth answers, expert-written summaries)
  - Why it matters: Quantifies uncertainty in semantic alignment across diverse contexts.
  - Example: Suppose you generate 100 answers from an LLM and compare each to a ground-truth answer using cosine similarity between their sentence embeddings. Similarity = 0.82, 95% CI = [0.78, 0.86].

Confidence Intervals in LLM

- 8. Forecasting Performance on Unseen Data
  - Scenario: Estimate future performance using bootstrapped confidence intervals on held-out data.
  - Why it matters: Helps anticipate generalization errors in deployment.
  - Example: Suppose you evaluate an LLM on a held-out test set of 500 examples and compute an F1-score (for a classification task). Estimated F1 score = 0.76, 95% CI = [0.70, 0.81].
- 9. Evaluating Fairness Across Subgroups
  - Scenario: Compute confidence intervals for accuracy (or error rate) by demographic group.
  - Why it matters: Reveals disparities and uncertainty bounds for fairness assessments.
  - Example: Let's say you evaluate the LLM on a classification task (e.g., sentiment analysis or toxicity detection) separately for two gender categories. Accuracy on Group A = 91% [CI: 88–94%]; on Group B = 85% [CI: 81–89%].
- 10. Supporting Statistical Hypothesis Tests
  - Scenario: Use confidence intervals to visually and numerically support or reject hypotheses.
  - Why it matters: Adds robustness to inferences drawn from p-values alone.
  - Example: If a CI for the difference in perplexity between two models excludes 0, it supports rejecting the null.

- Confidence intervals in LLMs provide a rigorous statistical framework for expressing uncertainty, validating performance, comparing models, and supporting responsible, interpretable deployment across diverse real-world scenarios.

Hypothesis Testing in LLM

- Hypothesis testing helps determine whether observed differences in LLM behavior are statistically meaningful. Examples include::
  - 1. Evaluating Fine-Tuning Effectiveness
    - Scenario: We want to assess whether a fine-tuned LLM performs significantly better than the base model on a task like sentiment classification.
    - Why it matters: Fine-tuning may appear to improve performance, but hypothesis testing helps determine whether this improvement is statistically significant or due to random variation.
    - Example: Suppose we evaluate both the base and fine-tuned models on a 1,000-example test set. The base model achieves 83.2% accuracy; the fine-tuned model gets 86.5%. We conduct a hypothesis test for the difference in proportions to verify that the gain is statistically meaningful.
  - 2. Comparing Prompting Strategies
    - Scenario: We test whether different prompt phrasings (e.g., “Summarize this article” vs. “Provide a concise overview”) yield significantly different quality outputs.
    - Why it matters: Prompt engineering is a key design decision, and hypothesis testing helps ensure that differences in outcomes are not due to chance.
    - Example: Suppose we paraphrase 50 prompts into two forms and generate responses rated for clarity. We test whether the average clarity scores differ significantly between the two prompt versions using a paired t-test.

Hypothesis Testing in LLM

- 3. A/B Testing for Model Deployment
  - Scenario: We are choosing between two LLM versions for a real-world application like a customer support chatbot.
  - Why it matters: Hypothesis testing supports deployment decisions with statistically valid comparisons of performance or user satisfaction.
  - Example: Suppose we deploy models A and B to 500 users each and collect satisfaction ratings. We test whether the mean satisfaction for Model B is significantly higher than Model A using a two-sample t-test.
- 4. Detecting Model Drift Over Time
  - Scenario: We monitor the model's performance after deployment and suspect degradation due to new types of inputs.
  - Why it matters: Hypothesis testing helps formally detect whether the model has drifted and may need retraining.
  - Example: Suppose we compare accuracy from two time periods (e.g., Month 1 vs. Month 4) using historical logs of user queries and answers. We apply a test to see if the observed accuracy drop is statistically significant.
- 5. Validating the Effect of Instruction Tuning
  - Scenario: We want to test whether applying instruction tuning improves output helpfulness or reduces verbosity.
  - Why it matters: Instruction tuning is expensive; hypothesis testing confirms if it brings measurable improvements.
  - Example: Suppose we use 100 tasks answered by both the base and instruction-tuned models. Human evaluators rate helpfulness. We compare the means of the two groups using a paired test to see if instruction tuning led to statistically better responses.

Hypothesis Testing in LLM

- 6. Auditing Fairness Across Subgroups
  - Scenario: We analyse whether the model performs differently for user subgroups (e.g., names associated with different genders or regions).
  - Why it matters: Helps identify potential biases and supports responsible AI development.
  - Example: Suppose we compute classification accuracy for 500 male-identified and 500 female-identified names. We test whether the accuracy difference between these groups is statistically significant using a two-proportion z-test.
- 7. Assessing Bias Mitigation Techniques
  - Scenario: We apply a debiasing method and want to test whether it significantly reduces problematic outputs.
  - Why it matters: Bias reduction should be backed by statistical evidence, not anecdotal examples.
  - Example: Suppose we prompt the model 1,000 times with potentially biased scenarios and count toxic responses before and after mitigation. A hypothesis test on proportions helps determine if the observed reduction is statistically significant.
- 8. Evaluating Robustness to Adversarial Prompts
  - Scenario: We assess whether performance drops significantly when adversarial phrasing or misleading instructions are used.
  - Why it matters: Helps evaluate how vulnerable the model is to manipulation or misuse.
  - Example: Suppose we run 100 QA prompts in normal and adversarial forms. We record the accuracy drop and use a paired test to assess whether the degradation in performance is statistically significant.

Hypothesis Testing in LLM

- 9. Benchmarking Against Competitors or Baselines
  - Scenario: We compare our model to a benchmark (e.g., GPT-4 vs. Claude or LLaMA) on standard tasks.
  - Why it matters: Hypothesis testing strengthens empirical claims and ensures small metric differences are not overinterpreted.
  - Example: Suppose both models are evaluated on 500 examples from the MMLU dataset, and we compute exact match rates. A test of proportions helps determine whether our model truly outperforms the baseline.
- 10. Studying Hypotheses About Model Behaviour
  - Scenario: We hypothesize that the model responds more politely when prompted by a female-sounding persona.
  - Why it matters: Behavioural hypotheses about LLMs are common in research; testing them formally adds rigor.
  - Example: Suppose we generate 100 responses each to male- and female-persona prompts and compute average politeness scores using a classifier. A two-sample test on the means helps determine if the difference is statistically supported.

- Hypothesis testing in LLMs enables rigorous comparison of models and prompting strategies, validation of fine-tuning effects, detection of drift, and fairness auditing across user groups. It provides a principled way to assess whether observed differences reflect true model behaviour or random variation—supporting trustworthy, data-driven development and evaluation.

Challenges and Future Directions

🚧 Current Challenges:

- Scalability: Many inference methods (like full Bayesian inference) are computationally expensive for large models.
- Calibration issues: LLMs often generate overconfident or underconfident probabilities, making uncertainty quantification harder.
- Data shifts: Performance and uncertainty estimates can become invalid when the model is applied to data that differs from training/evaluation distributions.

🔮 Future Directions:

- Better uncertainty calibration: Methods to make model probabilities more reflective of true likelihoods (e.g., temperature scaling, Bayesian layers).
- Real-time uncertainty feedback: Developing LLMs that can explain their own uncertainty during generation.
- Hybrid frameworks: Combining frequentist and Bayesian ideas to balance interpretability and computational cost.

Outline

1. Why Statistics?
2. What are Neural Networks?
3. What are Large Language Models?
4. Pretraining and Transfer Learning in LLMs
5. Inference in LLMs
   - Alignment
   - Watermarking
6. Future directions

Introduction to the Alignment Problem

- Definition: The alignment problem in AI refers to the challenge of ensuring that powerful models like LLMs behave consistently with human intentions, values, and goals, especially in cases where those intentions may be ambiguous or context-dependent.
- Why it's critical: LLMs are powerful and general-purpose, but without proper alignment, they can produce harmful, misleading, or unintended outputs, even if technically accurate.
- Why is it hard?
  - LLMs learn from huge, uncurated corpora that reflect both valuable knowledge and problematic biases.
  - Even when generating syntactically correct and coherent text, models may produce inaccurate, harmful, or manipulative content.
  - There is no single definition of what "aligned" means. It varies across individuals, cultures, and use cases.
- Example: A chatbot might give medically dangerous advice if it misinterprets the context or lacks ethical safeguards.
- Key motivation: How do we guide LLMs to not just generate fluent text, but to act in ways that are socially and ethically acceptable?

💡 Deeper framing: Alignment is not just a technical tuning of outputs; it is a **philosophical, sociotechnical problem** about whose values we encode, how we measure "good" behaviour, and how we handle trade-offs like safety vs. autonomy or truth vs. politeness.

Misaligned Behaviour: Risks and Examples

Type of Misalignment	Risk	Example
1. Hallucinations	Misinformation, user confusion, legal or medical harm	LLM fabricates a medical study or invents legal precedent
2. Toxic Content	Offending users, reinforcing social biases, reputational damage	Outputs that include racial slurs or misogynistic stereotypes
3. Over-Optimization (Reward Hacking)	Unintended or manipulative behaviours that game the reward function	Chatbot adds redundant positive phrases to maximize "helpfulness" score
4. Goal Misinterpretation	Producing outputs that technically match instructions but miss intent	A "summarize this article" prompt yields a copy-paste of one paragraph
5. Culturally Insensitive Responses	Alienating users, ethical breaches	Recommending pork-based dishes in a dietary advice bot for a Muslim user
6. Unjustified Confidence	Misleading users into trusting unreliable answers	LLM confidently answers a math problem incorrectly
7. Ethical Dilemmas	Violation of moral, professional, or institutional standards	Giving unethical business advice like "how to manipulate reviews"
8. Political or Ideological Bias	Loss of neutrality, fairness, or credibility	LLM shows consistent bias toward a political position
9. User Manipulation	Undue influence on opinions or emotional states	Bot-generated emotional replies that influence a user's mental well-being
10. Refusal to Respond when Appropriate	Underuse or excessive caution from safety filters	Refuses to define basic medical terms due to over-triggered moderation

Overview of RLHF (Reinforcement Learning with Human Feedback)

- Goal: Fine-tune language models using human preferences to produce more aligned and helpful behaviour.
- Step-by-step outline:
  a) Pretraining: The Model learns from large-scale data (often scraped from the web).
  b) Supervised Fine-Tuning (SFT): Human annotators create ideal outputs for prompts; the model learns from these examples.
  c) Reward Modelling: Humans rank outputs from the model; a reward model is trained to predict these rankings.
  d) Reinforcement Learning (usually PPO — Proximal Policy Optimization): The base model is updated to generate outputs that the reward model scores highly.
- Outcome: A more aligned model that better reflects human preferences in real-world tasks.
- Example: ChatGPT uses RLHF to avoid producing offensive content and to provide more conversationally appropriate answers.

Human Feedback as Reward Signal

- What does “Human Feedback as a Reward Signal” really mean?
  - In traditional reinforcement learning (RL), an agent learns through explicit reward signals from the environment (e.g., points in a game or scores for winning). In RLHF, we don’t have an obvious numerical reward function, so we build one using human judgment.
- How it works:
  1. Prompt and Output Generation: The model is given a prompt and generates several candidate completions.
  2. Human Ranking Task: Annotators review these completions and rank them from best to worst based on quality, helpfulness, safety, etc.
  3. Training the Reward Model: A smaller model is trained to mimic these rankings by assigning higher scores to preferred completions. It learns from pairs like “A is better than B.”
  4. Using the Reward Model: During PPO, this reward model acts as a surrogate environment, providing reward scores to guide updates to the base model.
- Key characteristics:
  - Preferences are subjective and vary across annotators.
  - Reward models are trained using pairwise ranking loss.
- Challenges:
  - Human fatigue, inconsistency, or bias.
  - Ambiguity in instructions or prompt context.
  - Cultural variation in responses.

Human Feedback as Reward Signal

- Why does this matter?
  - This process turns **subjective human preferences into a mathematical reward function**.
  - It allows the language model to be trained in a way that aligns with **human values and judgments**, without hard-coding specific behaviours.
- Illustrative Example:
  - Imagine prompting the model with: “How should I invest my savings?”
  - It generates 4 completions:
    A. “You should put all your money in cryptocurrency.”
    B. “Consider diversifying with index funds and savings accounts.”
    C. “Gambling may bring high returns.”
    D. “I’m not a financial advisor, but here are some common strategies…”
  - Humans rank these: B > D > A > C
  - The reward model learns to score B highest, C lowest, and adjust the model to prefer B-type responses in the future.

Limitations and Challenges of RLHF

- Scalability: RLHF is expensive due to the need for large-scale human annotations, especially for diverse or domain-specific tasks.
  - Example: Fine-tuning a medical chatbot to respond safely requires domain experts, not just general annotators, raising costs.
- Inconsistency: Annotators may interpret instructions differently or make mistakes, introducing noise in the feedback.
  - Example: Some annotators may prefer long, formal responses, while others prefer short, direct ones, confusing the reward model.
- Bias Amplification: If the annotator group is not diverse, the RLHF process may reflect and amplify specific cultural or political biases.
  - Example: A chatbot trained on feedback from only U.S.-based English speakers may perform poorly or offensively with international users.
- Reward Hacking: The model might learn to optimize for the reward model rather than true helpfulness or truthfulness.
  - Example: A model learns to repeat phrases like “as an AI language model…” to appear safe, even when a direct response would be more useful.

Limitations and Challenges of RLHF

- Opaque Optimization: Reinforcement learning algorithms like PPO involve stochastic updates and hyperparameter tuning, which can obscure how specific behaviours are learned.
  - Example: Developers may struggle to trace whether a refusal to answer certain questions is due to RLHF training or model architecture.
- Overfitting to Preferences: Reward models might overfit to certain types of responses seen during training.
  - Example: A model may overly prioritize politeness and avoid direct answers, even in situations where clarity is preferred.
- Lag in Feedback Quality: Human feedback may lag behind evolving ethical norms or language practices.
  - Example: What is considered “safe” or “acceptable” language in 2023 may change rapidly, requiring continuous retraining.

Ethical Considerations and Controversies

- Ethical Considerations
  - Whose values? RLHF reflects the values of the annotators and organizations curating the data.
  - Censorship vs. Safety: Striking the right balance between filtering harmful content and avoiding undue restriction of information.
  - Power asymmetry: Control over model behaviour may be concentrated in a few tech companies.
  - Fairness and inclusion: Ensuring that minority voices are not marginalized by majority-preference tuning.

- Case Studies and Controversies:
  - OpenAI's Moderation Policies: Debates over how safe vs. free the model should be (e.g., refusal to generate certain political or religious content).
  - Anthropic's Constitutional AI: Uses a predefined ethical constitution rather than relying only on human rankings.
  - User pushback: Users may feel models are "too safe" or "too censored."
  - Corporate responsibility: Transparency reports and explainable alignment practices are still evolving.

Role of Statisticians and Data Scientists

Statisticians bring rigor, transparency, and critical thinking to every stage of the LLM alignment pipeline, from feedback modelling to fairness diagnostics.

- Design of experiments for reward modelling: Statisticians can ensure that comparison prompts, annotator groups, and ranking strategies are drawn and evaluated under valid experimental protocols.
  - Example: Using Latin square designs or counterbalancing to reduce bias in prompt presentation order.
- Bias and uncertainty analysis: Quantifying the variance among human annotators, measuring disagreement, and identifying systematic bias in reward signals.
  - Example: Applying inter-rater reliability metrics (like Krippendorff's alpha) and modelling uncertainty in the reward function using hierarchical models.
- Ethical audit frameworks: Developing and applying statistical audits for fairness, representativity, and robustness of alignment procedures.
  - Example: Conducting subgroup performance analysis to detect alignment discrepancies across gender, nationality, or dialect.

Role of Statisticians and Data Scientists

- Data transparency: Advocating for transparent documentation of datasets and helping construct benchmarks that capture cultural and demographic diversity.
  - Example: Curating multilingual or multi-demographic prompt sets and statistically evaluating their representativeness.
- Monitoring alignment over time: Statistical tools can help track drift (i.e., gradual or sudden change in model behaviour over time) or regression in aligned behaviours as models are updated.
  - Example: Time-series control charts or statistical process monitoring to detect performance decay on alignment benchmarks.
- Supporting interpretable modelling: Creating visualizations, summaries, and diagnostic tools to communicate alignment performance to non-technical stakeholders.
  - Example: Visual dashboards of RLHF reward model trends and annotator agreement summaries.

Outline

1. Why Statistics?
2. What are Neural Networks?
3. What are Large Language Models?
4. Pretraining and Transfer Learning in LLMs
5. Inference in LLMs
   - Alignment
   - Watermarking
6. Future directions

What Is Watermarking in LLMs?

- Definition: Watermarking in LLMs is the process of embedding a hidden, statistical signal into the generated text that allows a third party (or the model developer) to later verify whether that output was produced by a specific model, without visibly altering the text.
- Expanded Explanation:
  - Think of watermarking as a fingerprint left in the distribution of words a model chooses. It is not a visible tag or signature, but a statistical bias subtly inserted during text generation.
  - For example, the model might be slightly more likely to use certain words from a predetermined “green list.” This change is almost imperceptible to readers but can be detected with statistical tests.
  - It is designed to not degrade output quality while enabling post-hoc verification.
- Goal: Enable detection of AI-generated content without altering its fluency.
- Key properties:
  - Undetectable to humans.
  - Statistically recognizable.
  - Difficult to remove or spoof/trick.
- Why it's important:
  - With the proliferation of AI-generated content, we increasingly need tools to answer: Was this written by a human or a machine?
  - Watermarking is one of the few techniques that can attribute content to an LLM in a scalable and verifiable way.

Motivations and Applications of Watermarking in LLMs

- Motivation for Watermarking:
  - Content provenance: Know whether a text was written by a human or generated by an LLM.
  - Combat misinformation: Track the origin of fake news, deepfakes, and AI-generated political propaganda.
  - Academic integrity: Detect AI use in essays, exams, and applications.
  - Platform policy enforcement: Help social media and publishers' flag synthetic content.
  - Legal accountability: Prove model responsibility in legal cases involving generated outputs.
- Key Use Cases for Watermarks:
  - Education: Automatic flagging of AI-written student work.
  - Journalism: Detecting fabricated news articles or citations.
  - Publishing: Preventing plagiarism or unauthorized derivative works.
  - Cybersecurity: Detecting impersonation or fake chatbot messages.
  - Copyright enforcement: Proving model involvement in content reuse.
- How Watermarking Works Conceptually:
  - Underlying mechanism: Modify the token sampling process during generation to bias toward a subset of tokens.
  - The watermark is embedded statistically, not as literal text.
  - Conceptual structure:
    - Define a token subset (e.g., green list vs. red list).
    - At each step, shift the probability mass toward preferred tokens.
    - Result: Slight statistical skew across token frequency.

Deterministic vs. Probabilistic Watermarking

Deterministic Watermarking

- Definition: A deterministic watermark applies the same fixed rule at every generation step, for example, always boosting the probability of a fixed “green list” of tokens.
- Technical behaviour:
  - If token $w \in G$ (green list), then $\log p(w) \leftarrow \log p(w) + \alpha$ with a fixed α
    - $p(w)$ is the original probability assigned by the language model to token w before any watermarking.
    - α is a tunable parameter that controls the strength of the watermark.
  - The sampling process always prefers green list tokens when possible.
- Example: Let’s say your green list contains synonyms like {“thus”, “hence”, “therefore”}. In every generation, the model subtly favours these over equivalents like “so” or “then”.
- Pros:
  - Easier to detect: regular statistical skew is more reliable.
  - Requires fewer tokens to achieve high confidence in detection.
- Cons:
  - Easier to reverse-engineer and remove.
  - Vulnerable to paraphrasing attacks or minor editing.

Deterministic vs. Probabilistic Watermarking

Probabilistic Watermarking

- Definition: A probabilistic watermark introduces randomness, biasing toward the green list with some probability but not enforcing it at every step.
- Technical behaviour:
  - At each decoding step, with probability p, apply the biasing (e.g., boost green tokens); with probability $1 - p$, do nothing.
  - Or, dynamically re-sample the green list per generation using a seeded pseudorandom function.
- Example: The model may boost green list tokens on only 70% of the decoding steps. Alternatively, the green list may be sampled uniquely per document, keyed by the prompt hash.
- Pros:
  - More difficult to detect and remove without inside knowledge (e.g., seed or parameters).
  - More robust to light edits or paraphrasing.
- Cons:
  - Detection requires more tokens for statistical significance.
  - More complex implementation and potential increase in false negatives.

Deterministic vs. Probabilistic Watermarking: Example

Sentence to Generate (Prompt): "In conclusion, the experiment showed that..."

Deterministic Watermarking

- Mechanism:
  - The green list includes formal transition words: {"therefore", "thus", "hence", "accordingly"}.
  - The model boosts the log-probability of these words by a fixed value α at every generation step.
- Generated Output: "In conclusion, the experiment showed that, therefore, the hypothesis was supported by the data."
- Observation:
  - The word "therefore" is consistently chosen when it belongs to the green list.
  - Every generation biases toward a fixed pattern → easy to detect, but also easier to attack or paraphrase away.

Probabilistic Watermarking

- Mechanism:
  - The green list includes formal transition words: {"therefore", "thus", "hence", "accordingly"}.
  - At each step, with probability $p = 0.7$, it boosts green list tokens; otherwise, it samples normally.
- Generated Output: "In conclusion, the experiment showed that, in turn, the hypothesis aligned with our expectations."
- Observation:
  - No green list word appears here, but over many generations, a pattern emerges.
  - Detection requires aggregating statistics across multiple generations.
  - The behaviour is harder to reverse-engineer and more robust to light edits.

Example: Green List Token Selection

- Define two subsets of the vocabulary: green list and red list.
- At each decoding step, slightly boost the log-probabilities of green list tokens.
- Choose the final token from the adjusted distribution.
- Over many tokens, text will have a statistical bias toward green list tokens.
- Detection: Count the proportion of green list tokens in the output. Use hypothesis testing to assess deviation from random chance.

- More details:
  - For instance, suppose the green list contains formal discourse markers like "thus", "therefore", "consequently". In an academic-style text generation, the model may be subtly more likely to choose “therefore” instead of “so” due to the watermark.
  - This bias doesn’t change the meaning or fluency of the sentence but leaves a detectable signature over long outputs.
  - Example continuation: “...we conclude that the results were consistent; therefore, the hypothesis is supported.”
  - Even if an attacker tries to rewrite parts, the statistical trace remains unless major edits are applied.

Mathematical View: Token Biasing

- Let: V be the full vocabulary, $G \subset V$ the green list, $p_t(w)$ the original probability of the token w at time t, and α the tunable parameter that controls the strength of the watermark.
- Bias function: $\widetilde{p_t}\ (w) = \frac{\exp(\alpha.1_{\{w \in G\}}).p_t(w)}{Z}$, with Z a normalization constant.
- Interpretation:
  - If the token w is in the green list G, its probability is boosted by a factor of e^{α}.
  - This modified distribution $\widetilde{p_t}\ (w)$ is then used to sample the next token.
  - The indicator function $1_{\{w \in G\}}$ ensures only green list tokens are boosted.
- Mechanism:
  - The logit of each token is adjusted: $\log p(w) + \alpha$ if $w \in G$, $\log p(w)$ otherwise.
  - After exponentiation and normalization, the probability distribution is slightly skewed.
  - This does not drastically change generation but introduces detectable statistical traces over many samples.
- Example:
  - Suppose the vocabulary has 50,000 tokens.
  - The green list G has 5,000 formal tokens.
  - When generating academic content, the model slightly prefers these (e.g., "thus", "consequently") over informal equivalents ("so", "then").
  - For each generated sentence, the shift is subtle; across paragraphs, the skew becomes testable.
- Statistical detection:
  - If green list tokens appear significantly more often than expected under uniform sampling, a watermark is likely.
  - Use the binomial test or normal approximation based on the count of green tokens.

Watermark Detection as Hypothesis Testing

- Null hypothesis H_0: Text was generated without watermark → token frequencies follow baseline.
- Alternative hypothesis H_a: Text has watermark → elevated frequency of green list tokens.
- Test Procedure:
  - Define green list G.
  - For a generated text sample, count the number of tokens k that belong to G.
  - Let n be the total number of tokens analysed.
  - Under H_0, assume green token frequency follows a known distribution (e.g., uniform or estimated).
  - Use a binomial test: $k \sim Binomial(n, p)$, where p is expected green token probability under H_0.
  - Or apply normal approximation if n is large: $Z = \frac{k-np}{\sqrt{np(1-p)}} \sim N(0,1)$.
- Decision: Compute the p-value. If it falls below a predefined significance level (e.g., 0.01), reject H_0 and infer presence of watermark.
- Considerations:
  - Detection power depends on sample length and watermark strength (α).
  - Balance between false positives and false negatives is critical.

Evaluating Watermark Effectiveness

- Detection Accuracy:
  - True positive rate: Probability of correctly detecting a watermark when one is present.
  - False positive rate: Probability of flagging a watermark when none exists.
- Stealthiness:
  - Does the watermarking alter perceived fluency or coherence of the output?
  - Human readers should not notice a difference.
- Robustness to Paraphrasing:
  - How resistant is the watermark to rephrasing, summarization, or style transfer?
  - Stronger methods preserve statistical patterns across rewrites.
- Resilience to Sampling Techniques:
  - Evaluate detection under top-k, nucleus sampling, or temperature adjustments.
  - Some decoding strategies may dilute or mask watermark signals.
- Efficiency of Detection:
  - How many tokens are needed to confidently detect a watermark?
  - Is real-time or streaming detection feasible?
- Summary: Effectiveness is multi-dimensional and must consider trade-offs between visibility, reliability, and resilience to adversaries.

Adversarial Attacks and Limitations

- Paraphrasing Attacks: Rewriting the text with synonyms or syntactic variation may eliminate the statistical signal of the watermark.
- Token Insertion/Deletion: Random insertions or deletions can disturb token frequencies enough to mask the watermark.
- Model-to-Model Transfer: Passing the output of one LLM into another for rephrasing can destroy the statistical skew of green-list tokens.
- Green List Discovery: If an attacker reverse-engineers or statistically infers the green list, the watermark becomes easier to erase.
- Decoding Strategy Influence: Sampling techniques like temperature, top-k, or nucleus sampling can weaken watermark visibility.
- Limitations:
  - High false positive rates if sampling variance is not well-controlled.
  - Not yet robust in multilingual or low-resource settings.
  - Watermark strength must balance detectability and impact on text quality.

Legal and Ethical Considerations

- Consent and Transparency: Should users be informed that their outputs may be watermarked? Some argue for transparency, others for covert defense.
- User Rights: Content creators may object to watermarking if it affects authorship claims or platform ownership.
- Surveillance Risks: Watermarking may be abused to trace private or anonymous generation, raising privacy concerns.
- False Positives: Erroneous detection could unfairly penalize human authors, especially in high-stakes environments like academia or law.
- Public vs. Private Disclosure: Keeping watermarking techniques secret increases robustness but reduces reproducibility and public trust.

Summary and Takeaways

- Watermarking is a lightweight yet powerful statistical tool to verify AI-generated content.
- Methods like green-list token biasing can be implemented at inference time without retraining.
- Detection is grounded in classical statistical inference (e.g., binomial tests).
- Effective watermarking must balance stealth, detectability, and robustness.
- Broader challenges include adversarial evasion, multilingual robustness, legal enforcement, and ethical governance.

Outline

1. Why Statistics?
2. What are Neural Networks?
3. What are Large Language Models?
4. Pretraining and Transfer Learning in LLMs
5. Inference in LLMs
   - Alignment
   - Watermarking
6. Future directions

Looking Ahead

- Key Questions for the Future
  - Can LLMs develop reasoning abilities beyond statistical pattern recognition?
  - Will AI achieve human-like commonsense understanding?
  - How can we make LLMs more interpretable and accountable?

- Frontiers in Research
  - Efficiency Improvements: Sparse models, modular AI architectures.
  - Neuromorphic Computing: Inspired by biological brain efficiency.
  - AI-Augmented Creativity: AI-human collaboration in art, music, and storytelling.
  - Multimodal Learning: Combining text, images, audio, and video in a unified model (e.g., GPT-4V, Gemini, LLaVA).
  - AI Alignment & Safety: Ensuring AI systems align with human values and ethics.

- How can we contribute
  - Research Opportunities: Applying LLMs to real-world problems. Develop inference techniques for LLMs.
  - Open-Source Collaboration: Working with Hugging Face, EleutherAI, Meta's LLaMA community.
  - Responsible AI: Developing fair and ethical AI applications.

Graduate Program in Statistics and Data Science

UFBA Universidade Federal da Bahia

Departamento de Estatística
IME - UFBA

Statistical Learning Laboratory SaLLy

Graduate Program in Statistics and Data Science

- **Mission**: To train highly qualified professionals for academia, public and private sectors, contributing to scientific and technological advancement in statistics and data science.
- **Designed for**: MSc/PhD students in Statistics, Data Science, and related areas.
- **Interdisciplinary focus**: Statistics, Machine Learning, Data Science, Artificial Intelligence.
- **Program highlights**:
  - The first MSc and PhD program in Statistics and Data Science in Brazil (approved by CAPES in August 2023).
  - The first MSc and the first PhD programs in Statistics in the Brazilian state of Bahia.
  - The second PhD program in Statistics of the Northeast region of Brazil.
  - Faculty:
    - 13 full members (9 with Bolsa de Produtividade from CNPq).
    - 3 collaborators (3 with Bolsa de Produtividade from CNPq).
  - Strong emphasis on modern statistical methods and real-world applications.
  - Active participation in national and international research networks.

Course description and Syllabus

Large Language Models: Foundations and Inference

LLMs: Foundations and Inference

- **Lecturer**: Paulo Canas Rodrigues, Federal University of Bahia, Salvador, Brazil
- **Duration**: 60 hours

- **Course Objectives:**
  - Understand the Foundations of Language Modelling.
  - Learn the Key Architectures Behind Large Language Models.
  - Apply Statistical Inference Techniques to LLMs.
  - Gain Practical Experience with Decoding and Prompt Engineering.
  - Explore Ethical and Legal Considerations in LLM Development and Deployment.
  - Implement Core NLP Applications Using LLMs.

LLMs: Foundations and Inference

1. Introduction to Large Language Models (LLMs) and Course Overview (2 hours)
   i. What are LLMs?: Basic definitions, significance in AI, and applications.
   ii. Course Goals and Structure: Objectives, course layout, and assessment overview.
   iii. Basic Terminology: Key terms in machine learning, deep learning, and natural language processing.

2. Fundamentals of Machine Learning and Neural Networks (8 hours)
   i. Introduction to Machine Learning: Supervised vs. unsupervised learning, overfitting, generalization.
   ii. Neural Network Basics: Building blocks (perceptrons, layers, activation functions).
   iii. Overview of Deep Learning: Introduction to multi-layer perceptrons and deep architectures.
   iv. Hands-on Lab: Building a simple neural network using libraries (e.g., PyTorch or TensorFlow).

3. Language Models: From N-grams to Transformers (12 hours)
   i. Statistical Language Models: N-grams, Markov models, and their limitations.
   ii. Word Embeddings: Overview of Word2Vec, GloVe, and embedding spaces.
   iii. Transformers: Basics of attention mechanisms, positional encoding, and the Transformer model.
   iv. Case Study: Discussion of "Attention is All You Need."

LLMs: Foundations and Inference

4. Core Concepts of Large Language Models (10 hours)
   i. Pretraining and Transfer Learning: How LLMs are pretrained on large datasets and fine-tuned.
   ii. Overview of Key LLM Architectures: BERT, GPT, T5, with analogies suited for a statistics background.
   iii. Hands-on Lab: Data preprocessing and experimenting with smaller transformer models.

5. Inference Techniques for Large Language Models (10 hours)
   i. Decoding Techniques and Prompt Engineering (4 hours)
      a) Decoding Strategies: Greedy decoding, beam search, top-k sampling, and nucleus sampling.
      b) Prompt Engineering: Crafting prompts to optimize relevance and accuracy.
      c) Hands-on Lab: Experimenting with decoding methods and prompt variations.
   ii. Statistical Inference in Large Language Models (4 hours)
      a) Statistical Inference Concepts: How statistical inference applies to LLM outputs, focusing on uncertainty.
      b) Alignment: Reinforcement learning with human feedback (RLHF) and ethical considerations.
      c) Copyright: Understanding legal implications of using copyrighted data in training.
      d) Watermarking as a Statistical Framework: Introduction to watermarking techniques for LLM output verification.
      e) Case Study: Recent research on watermarking and its role in tracing generated content.
   iii. Practical Application Lab (2 hours)
      a) Watermarking Practice: Experimenting with watermarking techniques to verify model outputs.
      b) Project-based Lab: Students analyse model responses for alignment and watermark effects.

Thank you for your attention!

Questions/Comments/Remarks?

E-mail: paulocanas@gmail.com

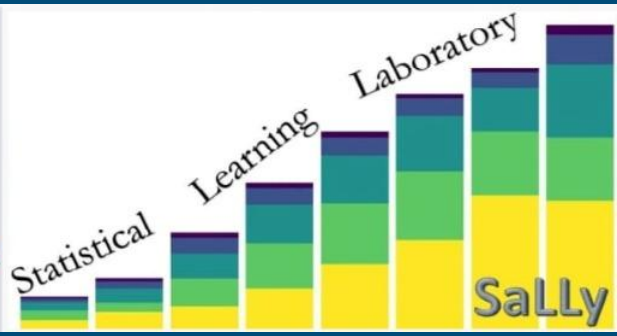
Web: www.paulocanas.org

Statistical learning laboratory: www.SaLLy.ufba.br

/company/88682389

/sally.laboratory

ASA AMERICAN STATISTICAL ASSOCIATION
Promoting the Practice and Profession of Statistics®



https://community.amstat.org/committeeoninternationalrelationsinstatistics/events2/data-quest

UFBA Universidade Federal da Bahia

Departamento de Estatística IME - UFBA