HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

Large Language Models and Generative AI for NLP - Lecture 4

Aarne Talman

Machine Learning & Data Science Lead, EMEA Centre for Advanced AI at Accenture
Visiting Researcher and Lecturer at University of Helsinki

Contents

Syllabus

Week 1: Introduction to Generative AI and Large Language Models (LLM)

- Introduction to Large Language Models (LLMs) and their architecture
- Overview of Generative AI and its applications in NLP
- Lab: Tokenizers

Week 2: Using LLMs and Prompting-based approaches

- Understanding prompt engineering and its importance in working with LLMs
- Exploring different prompting techniques for various NLP tasks
- Hands-on lab: Experimenting with different prompts and evaluating their effectiveness

Week 3: Evaluating LLMs

- Understanding the challenges and metrics involved in evaluating LLMs
- Exploring different evaluation frameworks and benchmarks
- Hands-on lab: Evaluating LLMs using different metrics and benchmarks

Week 4: Fine-tuning LLMs

- Understanding the concept of fine-tuning and its benefits
- Exploring different fine-tuning techniques and strategies
- Hands-on lab: Fine-tuning an LLM for a specific NLP task

Week 5: Retrieval Augmented Generation (RAG)

- Understanding the concept of RAG and its advantages
- Exploring different RAG architectures and techniques
- Hands-on lab: Implementing a RAG system for a specific NLP task

Week 6: Use cases and applications of LLMs

- Exploring various real-world applications of LLMs in NLP
- Discussing the potential impact of LLMs on different industries
- Hands-on lab: TBD

Week 7: Final report preparation

- Students work on their final reports, showcasing their understanding of the labs and the concepts learned.

Contents

Lecture 1: What are language models?

- **Language modeling** is the task of **estimating the probability distribution of sequences of words or tokens in a given language**.
- This can also be expressed as the **conditional probability of the next word given the previous words**.
- In essence, **language modeling aims to capture the statistical patterns and structures of a language**, allowing it **to predict the likelihood of a given sequence of words** or to **generate new text** based on existing sequences in the language.
- Current state-of-the-art large language models are **decoder-only transformers**, due to their parallelizability which allows training them on huge datasets in a reasonable time.

Lecture 2: Prompting

- **Prompting** is essentially how we communicate with and guide large language models to generate useful outputs.
- It involves **providing an input or instruction**, known as a prompt, **to the LLM**, which it then uses to predict and generate a relevant response.
- By prompting we can **trigger the model to generate desired type of content** (content, style, tone, domain, etc.) and to surface models' **emergent capabilities**.
- Some of the useful **prompting techniques** include:
  - Zero-shot prompting
  - Few-shot prompting
  - Chain-of-thought prompting
  - In-context learning

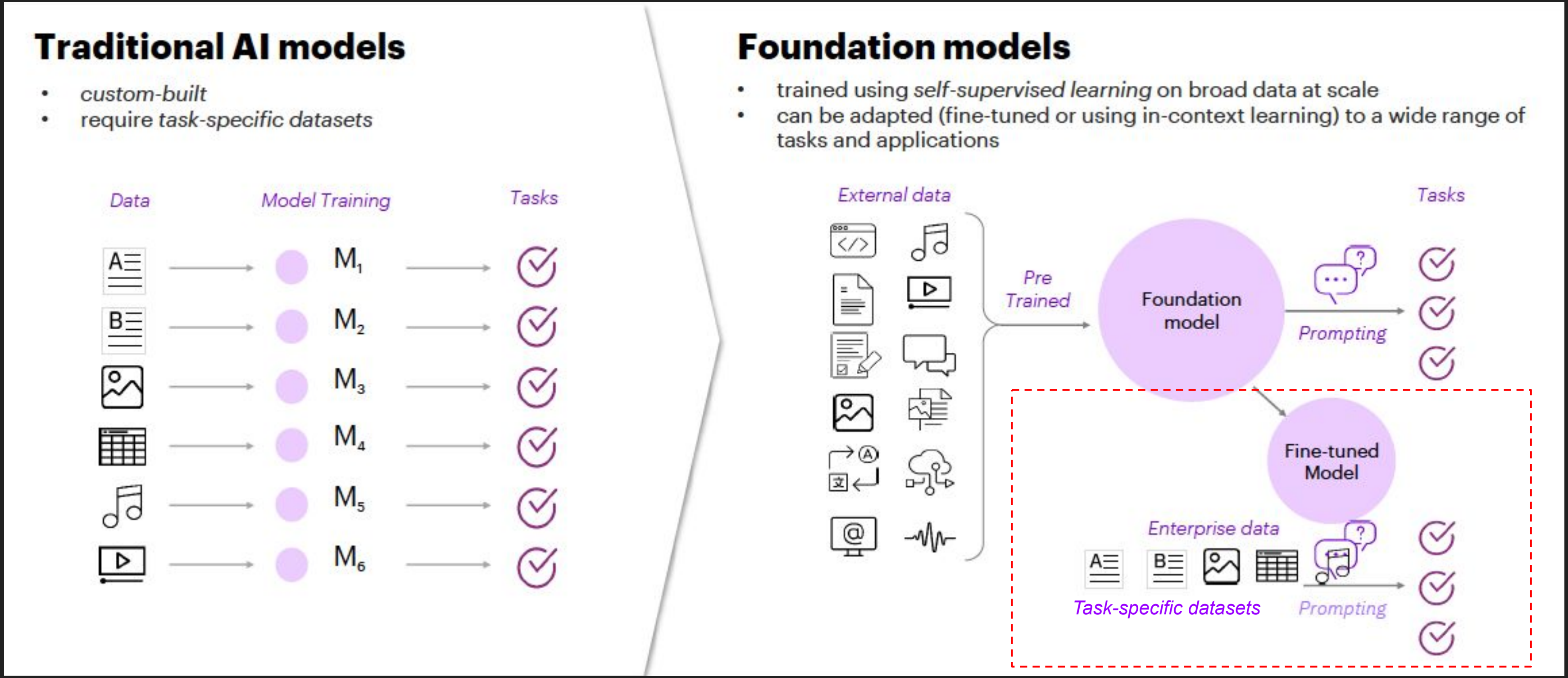
Lecture 3: Evaluation

- **Benchmarking:**
    - Evaluate and compare the performance of different LLMs on standardized tasks and datasets.
    - Aims to establish a baseline for measuring progress in the field and identify areas for improvement.
    - Some typical benchmarks: MMLU, GLUE, HellaSwag

- **Validation:**
    - Specific to a use case
    - Topical knowledge important
    - Might have use case -specific requirements for style, structure, vocabulary etc.

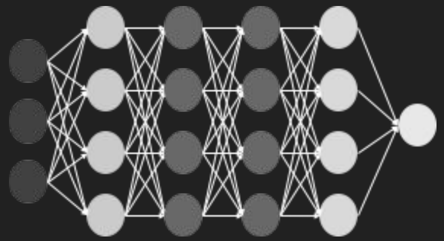
Contents

Introduction to Fine-tuning



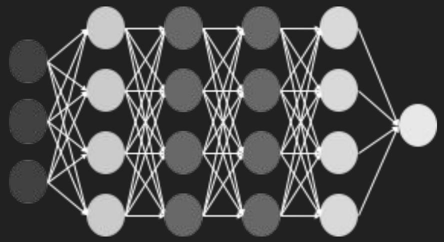
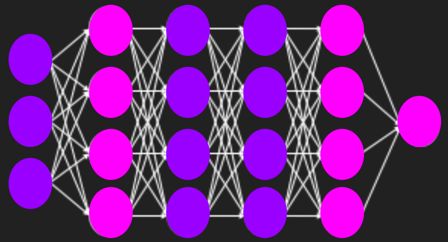
Prompting vs. Fine-tuning

PROMPTING

Prompt →  → Output

Model parameters do not change

FINE-TUNING

Fine-tuning data →  → 

Model parameters updated by fine-tuning

Updated model parameters

Why Fine-tuning?

- **Advantages:**
  - **Improved Performance (in some tasks):** Fine-tuning typically leads to significant improvements in performance on some target task compared to using the pre-trained model directly.
  - **Reduced Data Requirements:** It often requires less data than training a model from scratch, making it more feasible for many applications.
  - **Efficiency:** Fine-tuning leverages the pre-trained model's existing knowledge, making it faster and more computationally efficient than training from the ground up.
- **Example:**
  - A pre-trained language model can generate fluent text, but it might not be accurate at classifying customer reviews as positive or negative.
  - Fine-tuning on a dataset of labeled reviews improves its sentiment analysis capabilities.

How Fine-Tuning Works?

- **Process:**
  - **Start with a Pre-trained Model:** A large language model pre-trained on a massive corpus of text data.
  - **Prepare a Targeted Dataset:** Gather a dataset relevant to the specific task, often labeled examples.
  - **Continue Training:** Resume the training process using the targeted dataset, allowing the model to adapt and specialize.
  - **Evaluate and Iterate:** Assess the model's performance on the target task and fine-tune further if necessary.
- **Key Parameters:**
  - **Learning Rate:** Controls how much the model's parameters are adjusted during each training step. A smaller learning rate is usually preferred for fine-tuning to avoid overfitting.
  - **Number of Epochs:** The number of times the model sees the entire training dataset.

Contents

Fine-Tuning Methods

- **Supervised Fine-Tuning (SFT)**
  - **Process:** The most common approach, involves training the model on a dataset of input-output pairs, where the outputs are human-generated or labeled examples.
  - **Use Cases:** Widely used for tasks like text classification, question answering, and machine translation, where labeled data is available.
  - **Strengths:** Simple to implement and generally effective when sufficient labeled data is available.
  - **Limitations:** Can be data-intensive, and the model's performance is limited by the quality and diversity of the labeled data.

Fine-Tuning Methods

- **Instruction Tuning (a form of supervised fine-tuning)**
  - **Process:** Fine-tunes the model on a dataset of instructions and corresponding desired outputs, aiming to make the model better at following various instructions and prompts.
  - **Use Cases:** Enhances the model's ability to perform diverse tasks based on natural language instructions, making it more versatile and user-friendly.
  - **Strengths:** Improves zero-shot and few-shot performance, meaning the model can generalize to new tasks with minimal or no examples.
  - **Limitations:** Constructing a diverse and high-quality instruction dataset can be challenging.

Fine-Tuning Methods

- **Reinforcement Learning from Human Feedback (RLHF)**
  - **Process:** Combines supervised fine-tuning with reinforcement learning, where the model receives rewards or penalties based on human feedback on its generated outputs.
  - **Use Cases:** Useful for tasks where clear evaluation metrics are difficult to define, such as generating creative text or engaging in open-ended conversations.
  - **Strengths:** Aligns the model's behavior with human preferences and values, potentially leading to more helpful and harmless outputs.
  - **Limitations:** Can be computationally expensive and requires careful design of the reward function and human feedback collection process.

Fine-Tuning Methods

- **Direct Preference Optimization (DPO)**
  - **Process:** A more recent approach that directly optimizes the model to generate outputs that humans prefer, given multiple options (typically: chosen and rejected).
  - **Use Cases:** Similar to RLHF, suitable for tasks where human preference is the primary evaluation criterion.
  - **Strengths:** Potentially more sample efficient than RLHF, as it directly learns from human preferences without the need for explicit rewards.
  - **Limitations:** Still an active area of research, and the effectiveness can depend on the quality and diversity of human preferences collected.

Fine-Tuning Methods

- **Direct Preference Optimization (DPO)**
  - **Process:** A more recent approach that directly optimizes the model to generate outputs that humans prefer, given multiple options (typically: chosen and rejected).
  - **Use Cases:** Similar to RLHF, suitable for tasks where human preference is the primary evaluation criterion.
  - **Strengths:** Potentially more sample efficient than RLHF, as it directly learns from human preferences without the need for explicit rewards.
  - **Limitations:** Still an active area of research, and the effectiveness can depend on the quality and diversity of human preferences collected.

Contents

Parameter-efficient Fine-Tuning (PEFT)

- **Motivation:** Fine-tuning large language models can be computationally expensive and memory-intensive. PEFT aims to reduce these costs while maintaining performance.
- **Core Idea:** Instead of updating all the parameters of the pre-trained model, PEFT methods only modify a small subset of parameters or introduce new, lightweight modules.
- **Benefits:**
  - **Reduced Memory Footprint:** PEFT methods require significantly less memory than full fine-tuning, enabling fine-tuning on devices with limited resources.
  - **Faster Training:** Training is often faster due to fewer parameters being updated.
  - **Preservation of Pre-trained Knowledge:** PEFT methods tend to preserve the general knowledge of the pre-trained model better than full fine-tuning, reducing the risk of overfitting.

Parameter-efficient Fine-Tuning (PEFT)

- **Popular PEFT Techniques:**
  - **LoRA (Low-Rank Adaptation):** Injects low-rank matrices into specific layers of the model, allowing efficient fine-tuning.
  - **Prefix Tuning:** Adds a trainable prefix to the input sequence, guiding the model's generation without modifying its core parameters.
  - **Adapter Layers:** Inserts small, additional layers within the model's architecture, enabling task-specific adaptation.

Quantization

- **Motivation:** Large language models have billions of parameters, leading to large storage requirements and slower inference. Quantization addresses this by reducing the precision of model parameters.
- **Core Idea:** Quantization represents model parameters using fewer bits (e.g., 4-bit or 8-bit) instead of the usual 32-bit floating-point representation.
- **Benefits:**
    - Reduced Model Size: Quantized models are significantly smaller, making them easier to store and deploy.
    - Faster Inference: Computation on lower-precision numbers can be faster, especially on hardware optimized for such operations.
- **Trade-offs:**
    - **Potential Loss in Performance:** Quantization can lead to a slight decrease in model performance, although the impact is often minimal with careful techniques.

Contents

Lab

1. Run the supervised_finetuning.ipynb notebook in Google Colab
2. Change the base model used (search for small <7B parameter models in Hugging Face)
3. Change the dataset used in fine tuning
4. Bonus challenge:
    a. Change the fine-tuning method from supervised fine-tuning to DPO:
        i. Change the code accordingly, see: https://huggingface.co/docs/trl/en/dpo_trainer
        ii. Select an appropriate DPO dataset. Search Hugging Face Datasets.