

CSCE 771: Computer Processing of Natural Language

Lecture 14: LLMs Together - Large and Small Language Models

PROF. BIPLAV SRIVASTAVA, AI INSTITUTE

3rd OCT 2024

Carolinian Creed: "I will practice personal and academic integrity."

Organization of Lecture 14

- Opening Segment
 - Review of Lecture 13
- Main Lecture
- Concluding Segment
 - About Next Lecture – Lecture 15



Main Section

- LM Basics and Types
- Large Language Models
 - LLM Architecture
 - Instruction-Tuning
 - Fine-tuning for Task.
 - Efficient Inferencing
- CV/Resume Summarization Examples

Recap of Lecture 13

- We covered
 - Data and Tasks
 - BERT, Mamba, and some Evaluation
- We saw
 - how BERT can be used in two ways for two CV-related tasks
 - and saw Mamba and the some Evaluation strategies

Focused Classes

Sep 24 (Tu)	Language Model – PyTorch, BERT, {Resume data, two tasks} – Guest Lecture
Sep 26 (Th)	Language Model – Finetuning, Mamba - Guest Lecture
Oct 1 (Tu)	Language model – comparing arch, finetuning - Guest Lecture
Oct 3 (Th)	Language model – comparison of results, discussion, ongoing trends– Guest Lecture

GUEST LECTURES ON LANGUAGE MODELS

About Me


<https://github.com/kauroy1994/CSCE-771-NLP-Class/>



📍 South Carolina

🏛️ University of South Carolina

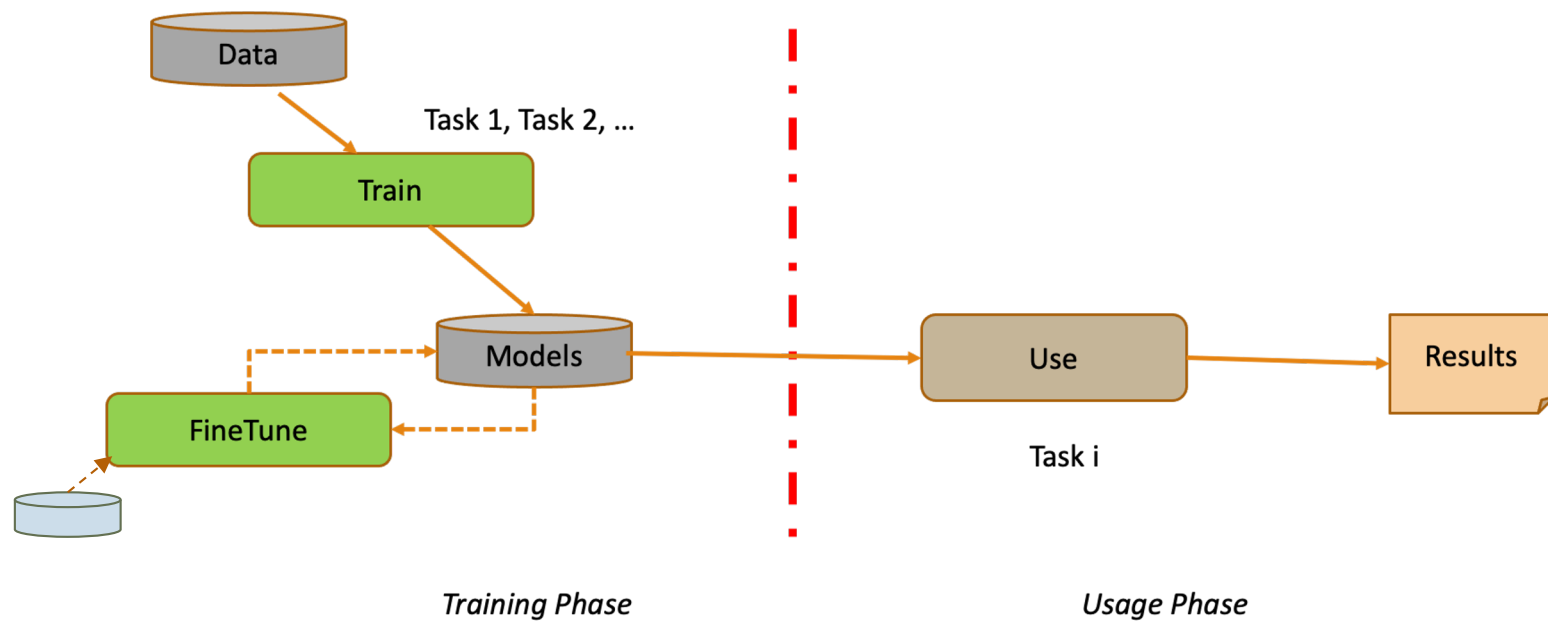
[Visit My Webpage](#)

I am Kaushik Roy, a Ph.D. candidate at the  [Artificial Intelligence Institute, University of South Carolina](#). My research focuses on developing neurosymbolic methods for declarative and process knowledge-infused learning, reasoning, and sequential decision-making, with a particular emphasis on social good applications. My academic journey has taken me from R.V. College of Engineering in Bangalore for my Bachelor's to Indiana University Bloomington for my Master's, and briefly to the University of Texas at Dallas before settling at the University of South Carolina for my doctoral studies. My research interests span machine learning, artificial intelligence, and their application in social good settings. I'm passionate about pushing the boundaries of AI, particularly in areas where it intersects with human understanding and decision-making.

Topics of Interest to Me: [Neurosymbolic AI](#) [Knowledge-infused Learning](#) [AI for Social Good](#) [Healthcare Informatics](#)

Main Lecture

LM Basics



Major LM Types

- ✓ Large
 - Large training dataset
 - Large number of parameters
- ✓ General purpose
 - Commonality of human languages
 - Resource restriction
- ✓ Pre-trained and fine-tuned



Credits: Google Cloud Skills Boost

LLMs have three different architectures - (a) encoder-only, (b) decoder-only, and (c) encoder-decoder, each with their own benefits.

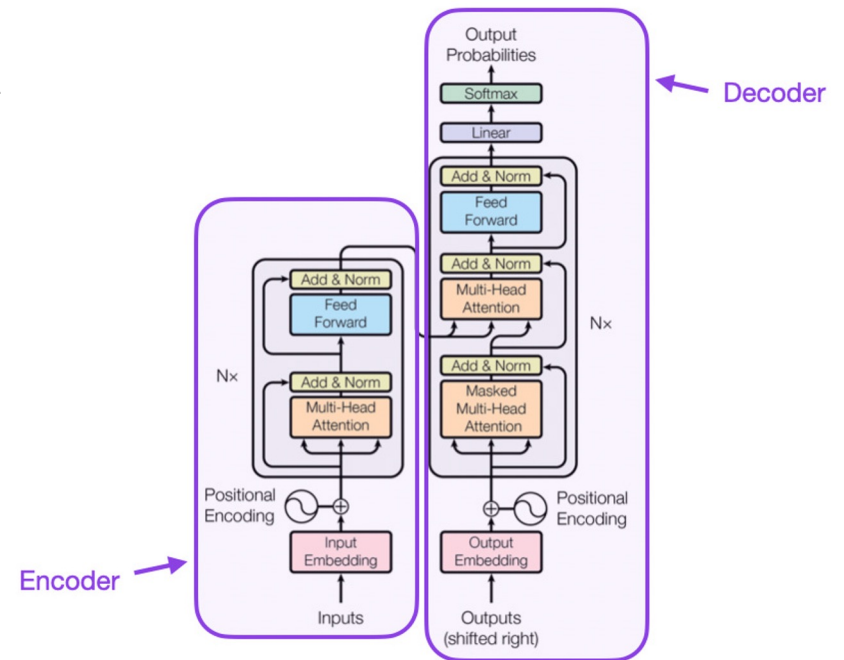
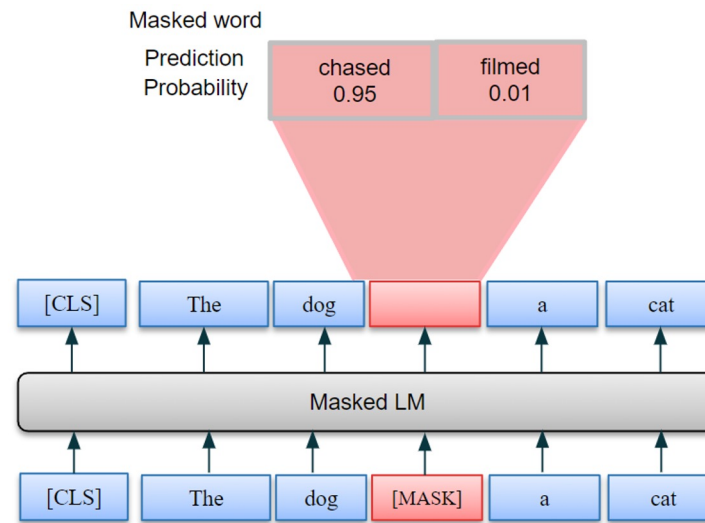


Figure. The Transformer - model architecture.

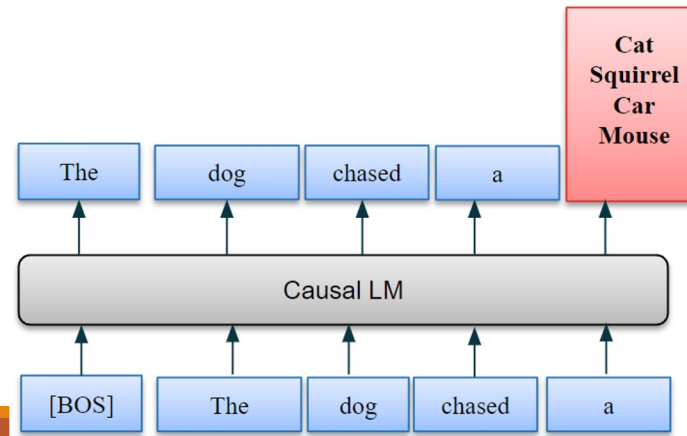
Encoder-only

- Encoder-only architectures are trained to understand the bidirectional context by predicting words randomly masked in a sentence.
- **Example:** BERT
- **Effective for:** sentiment analysis, classification, entailment.



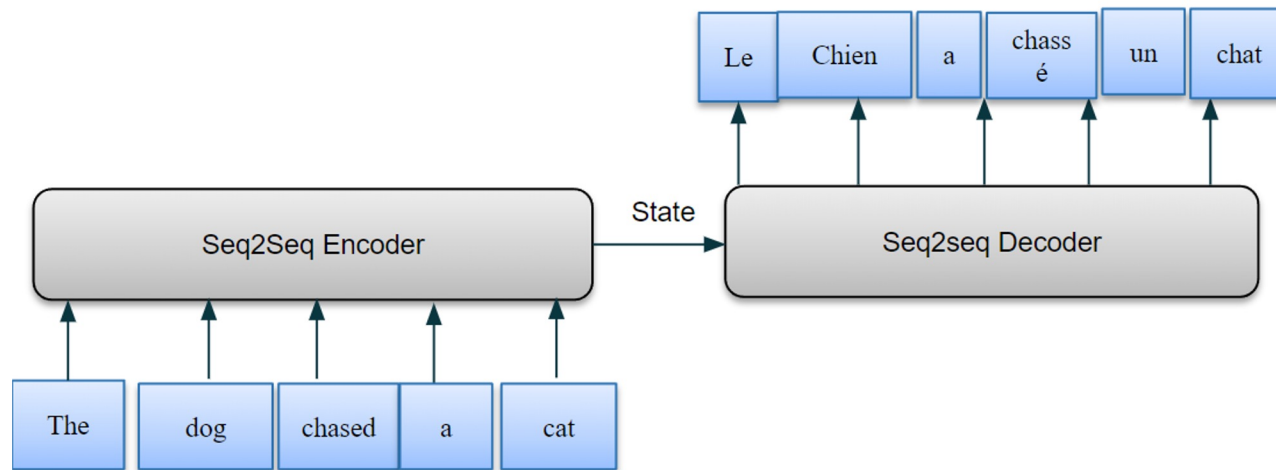
Decoder-only

- Decoder-only architectures are designed for tasks where text generation is sequential and dependent on the preceding context.
- They predict each subsequent word based on the preceding words, modeling the probability of a word sequence in a forward direction.
- **Example:** GPT-4, Llama series, Claude, Vicuna.
- **Effective for:** Content generation.



Encoder-Decoder

- Encoder-Decoder architectures are designed to transform an input sequence into a related output sequence.
- **Example:** T5, CodeT5, FlanT5
- **Effective for:** summarization, language translation.



Large Language Models (LLMs)

- **Large Language Models.**
 - **What are LLMs?:** Large Language Models are deep learning models trained on massive amounts of text data to understand and generate human-like language.
 - **Importance:** LLMs power many modern AI applications such as chatbots, machine translation, text summarization, and code generation.
- **Example Models:**
 - **GPT-3**, with 175 billion parameters, is a well-known example of an LLM used in tools like OpenAI's ChatGPT.
 - **Llama, Mixtral, Gemma**, by Meta, Mistral, and Google, respectively.
- **Key Advantages:**
 - Understanding context across long sequences of text.
 - Generating coherent and contextually accurate language outputs.

Autoregressive Models

- **Autoregressive Architecture:**
 - **How GPT works:** GPT predicts the next word in a sequence based on the words that came before it. This makes it **unidirectional** or **autoregressive**.
 - **Transformer Backbone:** GPT uses transformer layers to model long-range dependencies between words, which is a significant improvement over older models like RNNs and LSTMs.
- **Key Features:**
 - **Self-Attention Mechanism:** GPT uses self-attention to focus on important words in a sequence, allowing it to learn context.
 - **Scaling GPT:** GPT-2 had 1.5 billion parameters, and GPT-3 scaled up to 175 billion parameters, greatly improving performance on a wide range of tasks.
- **GPT Variants:**
 - **GPT-2:** 1.5 billion parameters, trained on 8 million web pages.
 - **GPT-3:** 175 billion parameters, trained on hundreds of billions of tokens.

Code Snapshot

```
# Sample CV text
cv_text = """
Experienced data scientist with expertise in machine learning, natural
language processing, and AI applications.
Proficient in Python, TensorFlow, and cloud computing. Published in top-tier
AI journals and conferences.
Skilled in data analysis, statistical modeling, and developing end-to-end AI
pipelines.
"""

resume_llm = LLM()
resume_llm.set_prompt(cv_text)
resume_llm.respond_to_prompt()
```

Instruction-Tuning and In-Context Learning (RLHF)

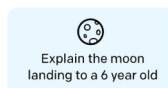
- **Instruction-Tuning:**
 - Models are fine-tuned on tasks where they are given explicit instructions. This makes them more adaptable to following human commands in natural language.
 - **Example:** ChatGPT, which can answer questions, summarize, or write essays based on user prompts.
- **In-Context Learning:**
 - **How it works:** Instead of retraining the model, we provide examples within the input prompt, allowing the model to learn on-the-fly.
 - **Advantage:** No need for model retraining—simply provide the right context in the prompt to guide the model.
- **Reinforcement Learning from Human Feedback (RLHF):**
 - **What is RLHF?:** This technique uses human feedback to fine-tune models and make them align better with human preferences.
 - **How it works:** Models generate outputs that are scored by human reviewers. These scores are used as rewards in a reinforcement learning algorithm to adjust the model's behavior.
 - **Use Case:** ChatGPT is fine-tuned with RLHF to generate more human-like and useful responses.

Instruction-Tuning and In-Context Learning (RLHF)

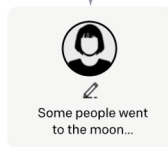
Step 1

Collect demonstration data, and train a supervised policy.

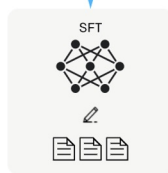
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



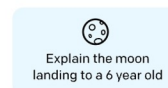
This data is used to fine-tune GPT-3 with supervised learning.



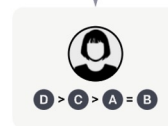
Step 2

Collect comparison data, and train a reward model.

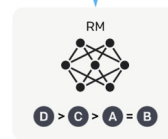
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

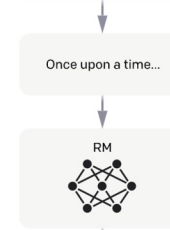


The policy generates an output.



Once upon a time...

The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



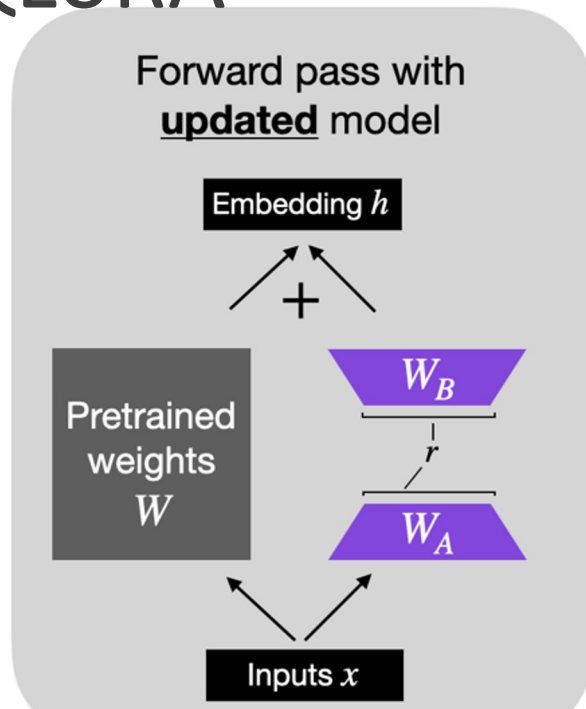
Code Snapshot

<https://github.com/kauroy1994/CSCE-771-NLP-Class/tree/Class-14>

Supervised Fine-Tuning with LoRA and QLoRA

- **LoRA (Low-Rank Adaptation):**
 - **What it does:** Instead of updating all model parameters during fine-tuning, LoRA introduces small low-rank matrices into specific layers of the transformer, significantly reducing the number of trainable parameters.
 - **Why it's useful:** Fine-tuning large models like GPT-3 requires enormous resources. LoRA reduces the cost by updating fewer parameters while maintaining performance.
- **QLoRA (Quantized LoRA):**
 - **What it does:** Builds on top of LoRA by quantizing the model parameters to further reduce the memory and computational cost. This allows fine-tuning even very large models on smaller hardware.
 - **Benefits:** Ideal for scenarios where hardware resources are limited but high accuracy is needed for fine-tuning.
- **Use Case:** Fine-tuning a GPT-3 model on a domain-specific dataset (e.g., finance, healthcare) using LoRA allows companies to adapt LLMs to their needs without requiring massive computational resources.

Supervised Fine-Tuning with LoRA and QLoRA



```
1  # LoRA Hyperparameters
2  lora_r = 8
3  lora_alpha = 16
4  lora_dropout = 0.05
5  lora_query = True
6  lora_key = False
7  lora_value = True
8  lora_projection = False
9  lora_mlp = False
10 lora_head = False
```

Credit:

- <https://lightning.ai/pages/community/lora-insights/>

Code Snapshot

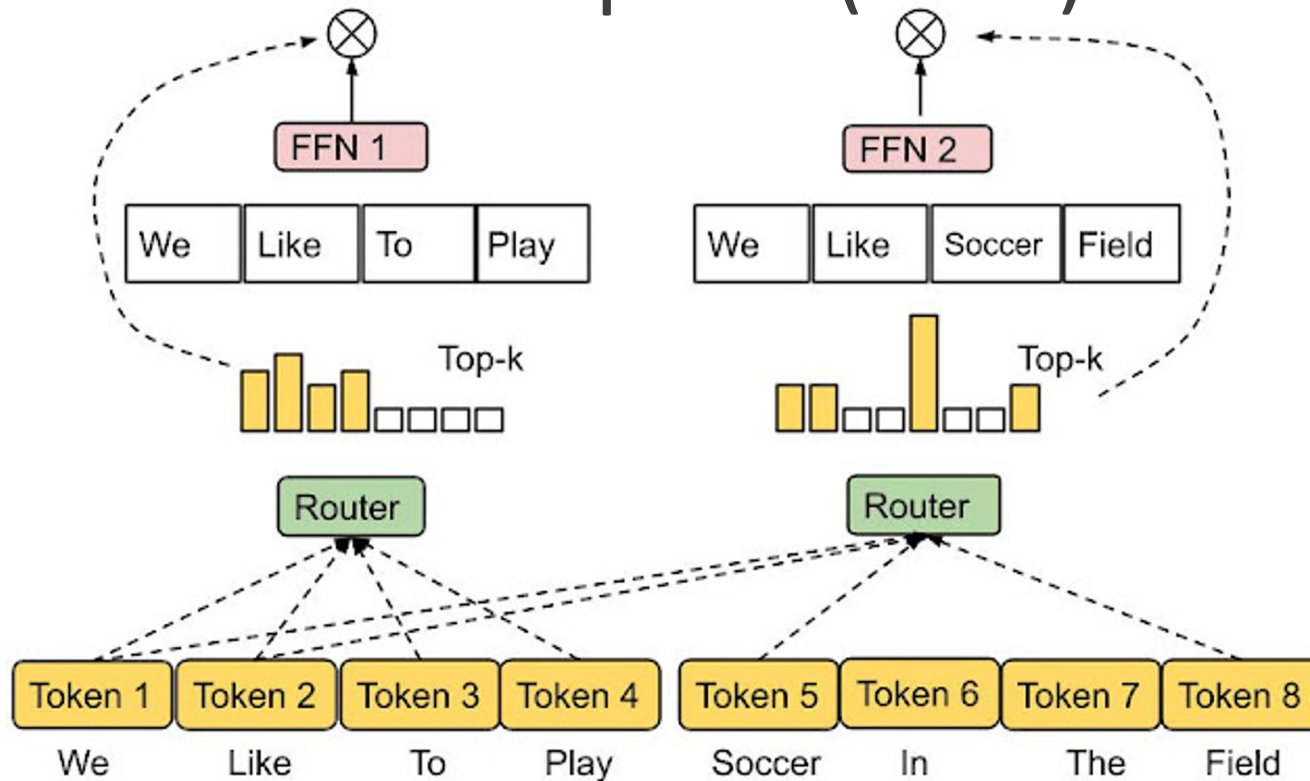
```
# LoRA fine-tuning setup (pseudo-code example)
from peft import LoraConfig, get_peft_model
lora_config = LoraConfig(
    r=8, lora_alpha=16, target_modules=["q_proj", "v_proj"],
    lora_dropout=0.05, bias="none"
)

model = GPT2LMHeadModel.from_pretrained("gpt2")
lora_model = get_peft_model(model, lora_config)
```

Mixture-of-Experts (MoE)

- **What is MoE?**
 - **MoE architecture** splits the model into multiple "experts," each trained to specialize in certain types of data or tasks.
 - **Efficiency:** Not all experts are activated for every input. Instead, a small number of experts are chosen based on the input, which saves computational resources.
- **How MoE works:**
 - **Routing mechanism:** When a model processes input text, a gating mechanism decides which experts to route the input through. This reduces the overall computation since only a subset of experts is active.
- **Benefits:**
 - **Scaling:** MoE allows models to scale up to trillions of parameters without needing to use all parameters for every input.
 - **Specialization:** Different experts can specialize in different tasks, improving overall model performance.
- **Use Case:** Mixtral's **Transformer** is an example of MoE in action, enabling multibillion-parameter models to be deployed efficiently.

Mixture-of-Experts (MoE)



Credit: <https://hkaift.com/the-next-llms-development-mixture-of-experts-with-expert-choice-routing/>

Resume Information Extraction and Summarization

Key Concepts:

- Using GPT for unstructured-to-structured information transformation.
- Prompt engineering to guide GPT in parsing specific fields from a resume.

Code Snapshot

```
from transformers import GPT2Tokenizer, GPT2LMHeadModel

tokenizer = GPT2Tokenizer.from_pretrained("gpt2")
model = GPT2LMHeadModel.from_pretrained("gpt2")

# Example resume text
resume_text = """
John Doe, Data Scientist
Education: M.S. in Data Science, University of ABC, 2020
Experience: 3 years working with machine learning and data analysis.
Skills: Python, SQL, TensorFlow
"""

# Prompt for structured information extraction
prompt = """
Extract key information from the resume:
- Name:
- Education:
- Experience:
- Skills:
"""

inputs = tokenizer(prompt + resume_text, return_tensors="pt")
outputs = model.generate(inputs["input_ids"], max_length=150)
print(tokenizer.decode(outputs[0]))
```


Instruction-Tuned GPT for Customized Summarization

Key Concepts:

- Instruction-tuned models can follow specific summarization patterns.
- Useful for HR or recruitment tasks where certain details (skills, projects) are prioritized.

```
# Instruction-tuning for summarizing a resume
instruction = """
Summarize this resume focusing on the candidate's skills and experience:
"""

inputs = tokenizer(instruction + resume_text, return_tensors="pt")
outputs = model.generate(inputs["input_ids"], max_length=100)
print(tokenizer.decode(outputs[0]))
```

Fine-Tuning for Domain-Specific Resume Parsing (LoRA)

Key Concepts:

- LoRA allows efficient fine-tuning on a resume-specific dataset, requiring fewer computational resources.

```
from peft import LoraConfig, get_peft_model

# LoRA configuration for fine-tuning on resume data
lora_config = LoraConfig(
    r=4, lora_alpha=16, target_modules=["q_proj", "v_proj"],
    lora_dropout=0.1, bias="none"
)

# Load pre-trained GPT2 model
model = GPT2LMHeadModel.from_pretrained("gpt2")
lora_model = get_peft_model(model, lora_config)

# Fine-tuning would involve loading resume-specific training data here
# (pseudo-code for fine-tuning loop)
def fine_tune_model():
    # Placeholder function for the actual fine-tuning process
```

Evaluation for Resume Summarization

Evaluate the quality of resume extraction and summarization using metrics like **ROUGE**, **BLEU**, and domain-specific accuracy.

```
# Evaluate the quality of summarization
from sklearn.metrics import accuracy_score
# Assume we have ground truth and predicted summaries
ground_truth = ["Skills: Python, SQL", "Experience: 3 years in ML"]
predicted_summary = ["Skills: Python, SQL", "Experience: 3 years with ML"]

accuracy = accuracy_score(ground_truth, predicted_summary)
print(f"Summarization Accuracy: {accuracy * 100}%")
```

Small Language Models (SLMs)

- **Small Language Models.**
 - **What are SLMs?:** Small language Models are a family of models that have a modest number of parameters meant to run on consumer-grade hardware, while at the same time retaining most of the response quality
 - **Importance:** LLMs power many modern AI applications such as chatbots, machine translation, text summarization, and code generation.
- **Example Models:**
 - **Quantized implementations**, with < **10** billion parameters, a well-known example of an SLM toolkit is **Ollama**.
 - **Models** such as **Flan-T5** are also considered SLMs, however, are not as popular.
- **Key Advantages:**
 - Provide consumers an avenue to implement LLMs in their business use-cases
 - Still generating coherent and contextually accurate language outputs, including code, JSON, etc.

How Small is Defined?

- By looking at a catalog. E.g., <https://console.groq.com/docs/models> (infer from metadata)
- By referring to a survey, <https://arxiv.org/abs/2409.15790> (defines as 5B parameters or below)
- By trying it out on your device. Usually, a model that can run without additional hardware acceleration

Code Snapshot

<https://github.com/kauroy1994/CSCE-771-NLP-Class/tree/Class-14>

Concluding Comments

- We looked at Language Modeling - Large Language Models, and Small Language Models
 - How they are trained
 - How they are instruction tuned
 - How they can be efficiently fine-tuned and inference
 - We covered small language models as well.

About Next Lecture – Large Language Modeling (GPTs and Family)

- Large language models
- Running on our tasks and comparisons

Concluding Segment

Course Project

Discussion: Course Project

Theme: Analyze quality of official information available for elections in 2024 [in a state]

- Take information available from
 - Official site: State Election Commissions
 - Respected non-profits: League of Women Voters
- Analyze information
 - State-level: Analyze quality of questions, answers, answers-to-questions
 - Comparatively: above along all states (being done by students)
- Benchmark and report
 - Compare analysis with LLM
 - Prepare report

- Process and analyze using NLP
 - Extract entities
 - Assess quality – metrics
 - Content – *Englishness*
 - Content – *Domain* -- election
 - ... other NLP tasks
 - Analyze and communicate overall

Major dates for project check

- Sep 10: written – project outline
- Oct 8: in class
- Oct 31: in class // LLM
- Dec 5: in class // Comparative

Obtaining Election Data

Here are a few things to do:

- A) **Official data** backed by laws: state election commission
 - a) Find the state's election commission
 - b) Find the Q/As they provide. They may be as FAQs or on different web pages.
 - c) Collect the Q/A programmatically

- B) **Secondary data** sources: non-profit
 - a) Find Q/As from Vote 411 which is supported by the non-profit: LWV.

For reference, for SC,

- A) Official - <https://scvotes.gov/voters/voter-faq/>
- B) Secondary - <https://www.vote411.org/south-carolina>

For extraction, one or more approaches:

- Manually annotating
- BeautifulSoup,
- Tika
- or other open source libraries.

Discussion

- How will you use a LLMs for election data analysis ?
- When and Why? (conversely, not)

<Student Name>

CSCE 771 Fall 2024: Milestone#1

1. State Selected:
2. Election data sites:
 - Official site (e.g., State Election Commission) url
 - Secondary site (e.g., League of Women Voters) url
1. Report how data collected and Q/A statistics
1. Take on NLP methods you will use and why for Q/A analysis
 1. State-level (right)
 2. Comparatively: above along states being done by peers

Initial analysis of questions (Q)

*

Initial analysis of answers (A)

*

Initial analysis of an answer (a_i)
for a question (q_i)

*

About Next Lecture – Lecture 15

Lecture 15 Outline

- Review of project in-class
- Template to be shared; create presentation and upload in common folder
 - 5 minute per student

7	Sep 10 (Tu)	Statistical parsing, QUIZ
8	Sep 12 (Th)	Evaluation, Semantics
9	Sep 17 (Tu)	Semantics, Machine Learning for NLP, Evaluation - Metrics
10	Sep 19 (Th)	Towards Language Model: Vector embeddings, Embeddings, CNN/ RNN
11	Sep 24 (Tu)	Language Model – PyTorch, BERT, {Resume data, two tasks} – Guest Lecture
12	Sep 26 (Th)	Language Model – Finetuning, Mamba - Guest Lecture
13	Oct 1 (Tu)	Language model – comparing arch, finetuning - Guest Lecture
14	Oct 3 (Th)	Language model – comparison of results, discussion, ongoing trends– Guest Lecture
15	Oct 8 (Tu)	PROJ REVIEW