DESN9002: Sustainable Leadership

Welcome, and let's start with some house-keeping items.

What is this course about?

Learn about Sustainability,

Observe, Experience and Practice Leadership.

Who to work with

- **Instructor:** Dr. Hongshan Guo
  - Lectures, Sustainability, Time Series (Predictive Analytics), Prototyping, Project/Idea scoping
- **Teaching Assistants**
  - Brad: Ice-breaking, presentation, project management, team formation
  - Abel: Logistics, announcements, social science, stats in general
  - Hsiang: Design, visualization, UI/UX design
  - Ishawn (Hackathon only): General Programming, LLMs & App-Building
- **Guest Speakers:** External, TBC

In [26]:
```
!pip install qrcode
```

```
Collecting qrcode
  Downloading qrcode-7.4.2-py3-none-any.whl (46 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46.2/46.2 kB 140.0 kB/s eta 0:00:
00a 0:00:01
Collecting pypng
  Downloading pypng-0.20220715.0-py3-none-any.whl (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.1/58.1 kB 415.3 kB/s eta 0:00:
00a 0:00:01
Requirement already satisfied: typing-extensions in /Users/hongshanguo/anacond
a3/lib/python3.10/site-packages (from qrcode) (4.5.0)
Installing collected packages: pypng, qrcode
Successfully installed pypng-0.20220715.0 qrcode-7.4.2
```

In [27]:
```
import qrcode
```

In [29]:
```
slackchannel=qrcode.make('https://tinyurl.com/desn9002')
```

In [30]:
```
survey =qrcode.make('http://etc.ch/sZVF')
```

How to work together

- Course Announcements & Assignment Submissions: Moodle
- **IM Platform**: Slack

https://tinyurl.com/desn9002

In [32]:
```
slackchannel
```

Out[32]:

- **>Remote Options**: Teams/Outlook
  - ping TAs if invites don’t show up on your outlook calendars.

How are you graded

Essays, Group Project (Hackathon) Critique and Peer-evaluation

- Essays reflecting on guest lectures, use your own experience (chatGPT?)
- Group Project aka Hackathon Output: Peer-grading (50%) + Instructor grading (50%)
- Peer-evaluation: Group contribution agreed and recognized within each group.

Group Project/Hackathon

Early detection of humanitarian crisis through multi-platform monitoring of multi-(social)media information.

1. Form Groups (each group should have representative of all majors), i.e. group size ≤ 5
2. Programmers, designers and researchers - but what about PMs?
3. Hackathon is time-boxed but research can start **now**.

Let's kick-off with a small survey

In [31]:
```
survey
```

Out[31]:

Introduction to LLMs: What they are, why they're powerful, why do we need to know about them

Hongshan Guo

September 22, 2023

What do you already know about LLMs/chatGPT?

tldr version?

- Language model trained on large text corpus,
- somehow capable of generating text sequence
- Tech industry/stock market went crazy, for a while
- Hallucination - what does it mean? Let’s have a look at an example for non-hallucination first.

PreRequisite

Temperature-checking - how many of these concepts are we not familiar with?

1. Introduction to Machine Learning: ML, Supervised/Unsupervised;
2. Deep Learning (Basics): Neural networks, activation functions, backpropagation, gradient desent;
3. Convolutional Neural Networks: CNN & application in CV tasks;
4. Recurrent Neural Networks (RNNs): Basics, LSTM and GRU & usage in sequence data;
5. Intro to NLP: Overview of NLP - tokenization, word embeddings, sentiment analysis;
6. Seq2Seq models: how they work and their role in tasks like machine translation;
7. Attention Mechanism: Attention, Significance in transforming sequences;
8. Transformers and BERT

Lecture Overview

1. **Introduction to LLMs**: What LLMs are and why they're important
2. **Training LLMs**: On LLM's training including challenges and techniques involved
3. **Understanding GPT Architecture**: Understand GPT architecture used in LLMs.
4. **Fine-tuning Large Language Models**: Concept of fine-tuning and its application in LLMs.
5. **Applications of LLMs**: Examine real-world applications of LLMs within various use cases.
6. **Evaluating LLMs**: Learn about different metrics and methods and their *caveats* to evaluate the performance of LLMs.
7. **Bias in LLMs**: Discuss the potential for bias in LLMs and how to mitigate it.
8. **Limitations and Future of LLMs**: Discuss the current limitations of LLMs and future research directions.
9. **Hackathon-related refreshers**: Stochastics of time seires, API-based call of LLMs and Graphical Programming Interface that requires little-to-zero coding

Introduction to LLMs and how they are trained

A type of ML model designed to understand, generate and converse in human language, 'large' due to the vast number of parameters.

- Ability to generate human-like texts
- Patterns in data used to train the model learnt allows the models to generate text based on received inputs
- Rule of thumb: 7B parameters takes one sota GPU to run, i.e. 13B takes two, etc.
- LLMs can perform natural language processing (NLP) tasks, note LLM ≠ NLP model
- OpenAI first released GPT-1 in 2018, and GPT 3 in 2020, where terrabytes of data were used to train these models

Let's see some LLMs in action

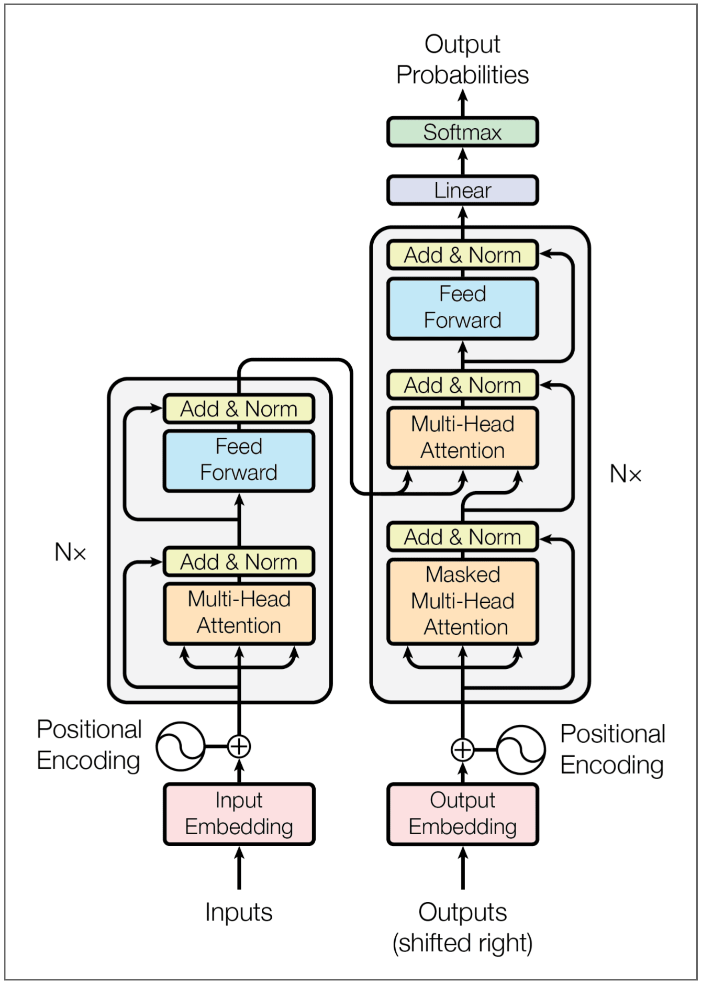
Architecture of LLM

'Attention is All you Need', seminal paper on the most commonly used architecture known as **transformer** was first proposed, revoluntionized the field of NLP.

- Transformers are based on the **attention** mechanism, which allows the model to better associate words w.r.t. their positions, of primarily two types:
    - self-attention
    - multi-head attention
- Transformer = **encoder** (input) + **decoder** (Output)

In [17]:
```
transformerarch
#For those more keen on looking at the two components - note that GPT-3 is decoder onl
```

Out[17]:



Architecture of LLM - Transformer Layer Types

1. Self-Attention Layer: Scaled Dot-Product Attention
    - Allows model to focus on different parts of the input sequence when producing an output.
        - e.g. "The cat, which already ate...., was full." can prioritize "The cat" with "was full" before "already ate";
2. Position-wise Feed-Forward Layer
    - Fully-connected feed-forward network that is applied to each position separately and identically, each consists two linear transformations with a ReLU activation in between;
    - Used to sequentially process the output of the self-attention layer, appying the same learnt weights at every position;
3. Output layer
    - Linear layer followed by a softmax function, transforms the final hidden states to predictions for the next word in the sequence for each possible

word in the vocabulary.

Architecture of LLM - Data Handling by GPT Models

Tokenization

Process of converting sequence of text into a sequence of tokens - a (part of) word.
Byte Pair Encoding platform.openai.com/tokenizer
https://chatgpt.hku.hk is now available thanks to Prof. Ian Holiday. 50K tokens per month - do you know what tokens are?

BPE

- subwod tokenization method,
    - starts by splitting text into individual characters and
    - iteratively merges the most frequently adjacent pair of symbols.
- helps handle out-of-vocabulary words and makes the model more robust to spelling errors and variations

Sequence length

- context window size (2048 for GPT-3, 8K for GPT-4 or 32K)
- Limit on the number of tokens they can be passed at once due to memory constraints
- For an decoder-only model takes the input sequence all at once, where each token and its relationships with every other token in the sequence in memory
- quadratic increase in memory usage at the sequence length grows
- Primarily to make the model feasible to run on available hardware
- **Longer (input + output) sequence takes very long to generate**

Architecture of LLM - Implementation and deployment of GPT

Implementation

- Unlikely going to have retraining schedules.
- May require initial fine-tuning or tweaking OR
- Vanilla
- GPT-in-general: Implemented with Tensorflow or PyTorch: efficient, GPU-accelerated operations

Deployment Process

- Models needs to reside on robust hardware or cloud-based solutions: consider numerous simultaneous requests
- Typically involves API-calls made to request services from models deployed on network
- Requests (e.g. text generation, text-completion) are handled and responses sent back

Architecture of LLM - Back to the SOTA models

GPT-3, GPT-J, etc.: An **autoregressive, decoder-only transformer model** designed to solve natural language processing (NLP) tasks by **predicting** how a piece of text will continue.
This is different from traditional encoder-decoder transformer models like BERT where the inputs are first encoded and thrown together at the model as a whole when making the prediction.
GPT-3-like decoder-only models puts more emphasis on the more recent inputs, maknig the prediction continuously being more relevant with the more recent information.

- Advantage of decoder-only architecture:
    - Simplicity: easier to train less computationally expensive
    - Good at generative tasks, producing contextually relevant text - model is built to generate output one token at a time;
- Advantage of encoder-decoder model:
    - Better at classification tasks - tasks where specific structure is needed, e.g. translation, summarization, particularly good when input information

comes all at once

Note here these are just high-level benefits, YMMV with different use cases.

A few notes:

1. *Autoregressiveness equals no parallelized during inference*
2. *Autoencoder* models should be differentiated from encoder-decoder model:
    - the prior also has an encoder and a decoder but primarily serves the purpose of dimension-reduction and denoising, aka 'learning' the input while
    - the latter is designed to work in sequence-to-sequence tasks.
    - i.e. autoencoder's output is representation of its input
    - encoder-decoder is not just a reconstruction of the input

Tying this back to GPT-3 models (and its alikes): LLMs performs sequence-to-sequence tasks but puts more weights to more recent context, yet it still considers all previous tokens (history) to predict the upcoming token.

Architecture of LLM - Generative Pre-trained Transformer aka OpenAI's proposal

1. GPT-1: Improving Language Understanding by Generative Pre-training" in June 2018. 12 transformer layers and 117m parameters;
2. GPT-2: Released in 2019, has 1.5B parameters, text can become indistinguiashable to text written by humans;
3. GPT-3: Launched in 2020, (disclosed) largeset variant at 175B parameters, generate coherent and contextually relevant passages of text, introduction of 'few-shot learning'.
    - translation
    - Q&A
    - Writing creative contents like poems and stories
4. GPT-4: Released in 2023, no. of parameter undisclosed, still known as the most powerful LLM that has been released;

GPT-3.5: sub-class of GPT-3 first released in Apr. 2022, leading to '**chatGPT**' in Nov. 2023.

Applications of LLMs: Examine real-world applications of LLMs within various use cases.

- Chatbots: User intention and logic. Short back-and-forth limits the need for larger context window
- Script-writing: Hollywood writers still on strike - although for a different reason now.

Evaluating LLMs: Learn about different metrics and methods and their caveats to evaluate the performance of LLMs. Bias in LLMs: Discuss the potential for bias in LLMs and how to mitigate it. Limitations and Future of LLMs: Discuss the current limitations of LLMs and future research directions. Hackathon-related refreshers: Stochastics of time seires, API-based call of LLMs and Graphical Programming Interface that requires little-to-zero coding

Common Selection Strategies seen in LLM Application Development

Inference-time paramters that affect how tokens are generated by a LLM.

- Temperature: Deterministic to Creativity
    - Randomness of model output (0-1)
    - When set to 0, outputs is completely reproducible
    - When set to 1, model is more random, aka has more 'creative' outputs
- Top_K Sampling: Number of top tokens considered for best next word
    - limits us to a certain number of the top tokens to consider
- Top_P Sampling: Probability threshold
    - Recall the models are statistical in nature, use a threashold of probability, e.g. 90%
- Token length & Max tokens: Number of tokens fed to/generated by LLMs
    - Relevant due to the amount of time of solution generated
    - Also relevant to costs (Most close models charge by # of tokens used)

Break

Quick Recap

- Talked about onboarding to GCP and some high-level contents w.r.t. how LLMs work
- In particular touch upon the following concept:
    - Temperature
    - top_N
    - probabilities
- Now switching gear back to the learning series, we will talk about Application of LLMs today.
- Let's begin with the five common pillars recognized by OpenAI (or GPT-4 itself)

Key Applications Recognized

1. Content Creation
    - write articles
    - generate ideas/scripts for writers
    - **creativity and fluency**
2. Conversation Agents:
    - chatbot/virtual assistant (Pershin X)
    - Human-like **interaction**
    - *Can answer queries, provide recommendations and hold conversations*
3. Translation:
    - not explicitly trained to do this
    - *Can handle various languages nonetheless*
4. Education
    - Tutor various subjects
    - *Can provide explanations, stimulate creative thinking*

5. Programming Help: CoPilot
    - *Can generate code snippets and assist with debugging*

Limitations and Ethical Considerations of Large Language Models

- Limitations
  - Understanding vs. Pattern Matching
  - Sensitivity to Input Phrasing
  - Verifiability of information: generating plausible-sounding but incorrect/nonsensical information.
    - no inherent way to verify accuracy of the information it generates
    - **but doesn't 'grounding' solve this?**
- Ethical Considerations
  - Bias in Training Data
    - gender, racial cultral biases and more
  - Misuse of Technology
    - deepfake text for nefarious purposes
  - Social/Environmental Impact
    - training of LLM is very energy intensive ad requires significant

amount of computational resources

- Role of human/smart copiers' role in RLHF

Is this Hallucination? (llama2-13b from Meta)

- **Q**: When did Mickey Mouse die?
- **A1**: Mickey Mouse, the iconic cartoon character created by Walt Disney and Ub Iwerks in the 1920s, has not died. He is a fictional character and has been featured in countless animated films, television shows, comics, and other forms of media. Therefore, he has never actually died. However, if you're asking about the character's where- abouts or activities, it's important to note that Mickey Mouse is a fictional character and does not have a physi- cal presence or real-world experiences. He exists solely as a cartoon character and is not capable of dying or experienc- ing death in the same way that real people do. (llama2- 13b)

Is this Hallucination? (llama2-7b from Meta)

- **Q**: When did Mickey Mouse die?
- **A2**: Mickey Mouse, the beloved cartoon character created by Walt Disney and Ub

> Iwerks in 1928, is not a real per- son and has not died. Unterscheidung von Mickey Mouse und realen Menschen. Mickey Mouse is an fictional char- acter and has never been alive, so he has not died. He was created as a cartoon character for the Disney Broth- ers Cartoon Studio, which later became The Walt Disney Company, and has been a beloved character around the world for generations. (llama2-7b)

The decoder-only transformer models are **autoregressive** in nature, i.e. they use the given context to predict words that are most likely going to occur according to the relationship they've already 'learnt'.

How does ’Grounding’ work?

- What is 'grounding':
    - Refers to the concept of connecting the LLM with real-world concrete data or knowledge source
    - May include:
        - linking to specific databases,
        - using factual knowledge graphs, or
        - providing the model with access to up-to-date information on the internet.
- MSFT once claimed it will resolve most of our concerns/problems.
- **Verfiability issue is more than just factual grounding.**

Why doesn't 'Grounding' work?

- Grouding will improve accuracy of responses and keep them factual
- Creates a form of question-answering system that performs semantic search over a known corpus of documents
- Some points to consider:
    1. Model interpretation (temp=0, aka model outputs more deterministic): model understanding ≠ human understanding;
    2. Accuracy within context: errors acn arise from:
        - biases/inaccuracies in training data,
        - misinterpretation of context,
        - limitations in the model's understanding of complex language structures;
    3. Handling Ambiguity: question or context is ambigous, the model might struggle to provide an accurate and relevant answer.

'Grounding' may improve verifiability of a model's response, it won't eliminate the

issue.

Mitigating LLM Risks

- Fine-tuning on specific datasets: improve behavior for **specific** applications
    - Refers to the concept of connecting the LLM with real-world concrete data or knowledge source
    - May include:
        - linking to specific databases,
        - using factual knowledge graphs, or
        - providing the model with access to up-to-date information on the internet.
- **Verfiability issue is more than just factual grounding.**

Quick Re-Cap on what we discussed

- Basics of LLMs
    - What they are and how they are trained
- Architecture of GPT models (Decoder-only ones started by OpenAI)
    - Transformer architecture
    - Decoder-only vs. encoder-decoder architectures
- Implementation and Deployment
    - APIs
    - Real-world usage and applications we built
- Limitations and Concerns
    - Lack of human-intent-understanding capability
    - Inability to provide verification
    - Proneness to biases
- Implications of LLMs
    - Wide-ranging capabilities and applications of these models

- Potential risks and concerns
- Strategies to mitigate these risks

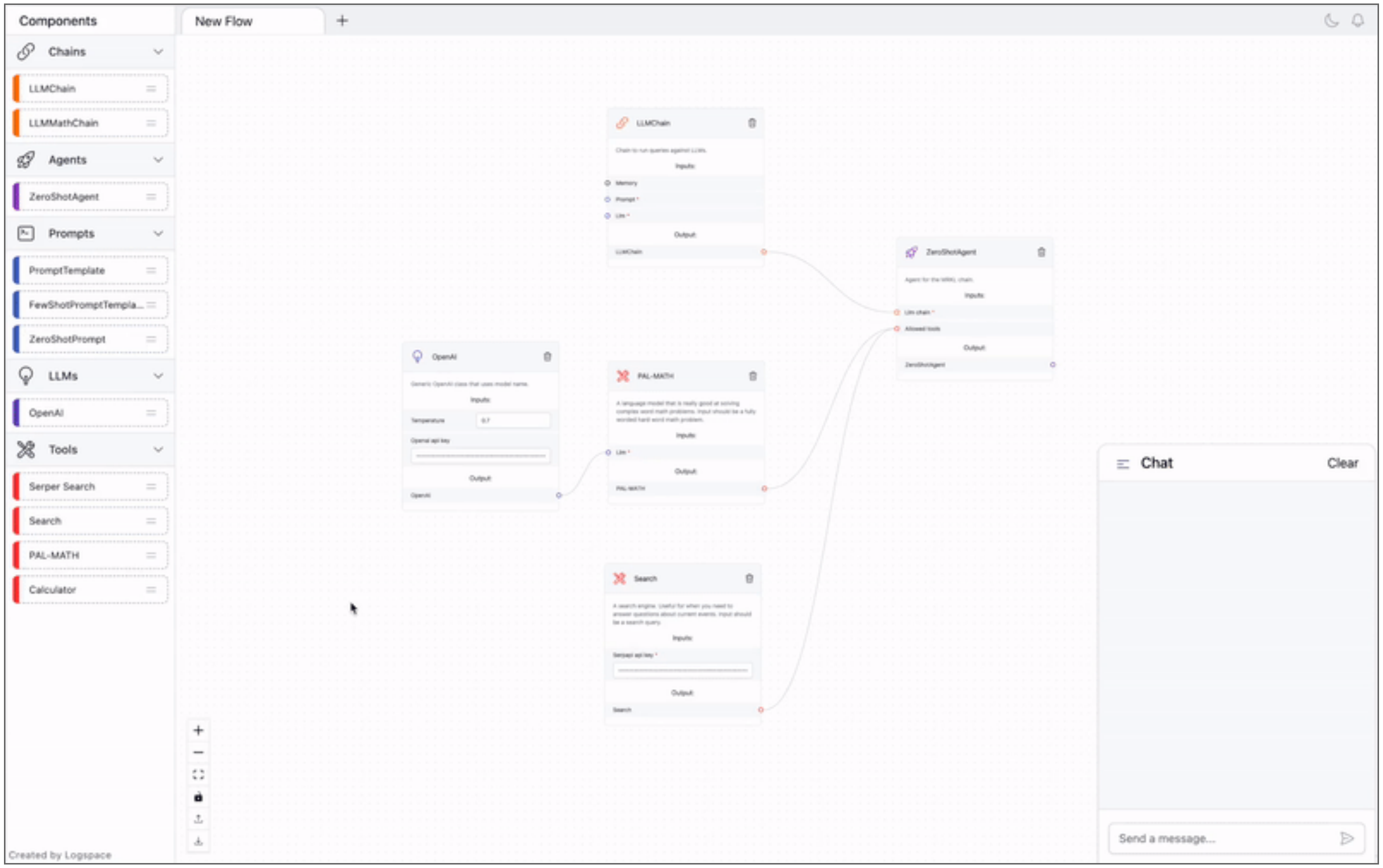
Quick Re-Cap on what we discussed - Synthesized

- Basics of LLMs
    - Generative Pre-Trained (on large text corpora) Models
- Architecture of GPT models (Decoder-only ones started by OpenAI)
    - Decoder-only, no parallization
- Implementation and Deployment
    - OpenAI & Google: APIs
    - Open-Source models: Google Colab, i.e. local/remote GPUs(CLI)
- Limitations and Concerns
    - It doesn't 'understand' prompts and human intentions
    - But can sound very convincing as trained on great writing examples - is it a better writer?
- Implications of LLMs
    - Different learning experiences
    - Revolutionize coding experience (CoPilot)

- Further reduction of workforce (e.g. Customer Service, call center reps)

Some Options to test/understand LLMs

- Langflow: Python implementation



Some Options to test/understand LLMs

- Flowise: Node.js Implementation