Large Language Models

Introduction and Recent Advances

ELL881 · AIL821

Tanmoy Chakraborty
Associate Professor, IIT Delhi

Semester 1, 2024-2025

Course Instructors

Tanmoy Chakraborty
IIT Delhi

Yatin Nandwani
IBM Research

Dinesh Raghu
IBM Research

Course TA

Anwoy Chatterjee
PhD student, IIT Delhi

Sourish Dasgupta
DA-IICT

Gaurav Pandey
IBM Research

Manish Gupta
Microsoft

Course Directives

- Slot **H** (Mon, Wed: 11-12; Thu: 12-13)
- Website: https://lcs2-iitd.github.io/ELL881-AIL821-2401/
- YouTube: https://www.youtube.com/@lcs2575
- Room: Bharti-301

Marks distribution (tentative)

- Minor: 15%
- Major: 25%
- Quiz (2): 10%
- Assignment (1): 20%
- Mini-project: 30% (group-wise)

- **Audit**: B- (threshold to pass the course)
- **Grading Scheme**: TBD

Course Project

- Some problem statements, and datasets will be floated soon*
- Each group should consist of 1-2 students?
- **Best Project Award** 
- You need to
  - develop models
  - evaluate your models
  - prepare presentation
  - write tech report

Students are encouraged to publish their projects in good conferences/journals

* You are welcome to propose a new idea if you find it fascinating to be qualified for a course project. Instructor opines!

Course Project

- Some problem statements, and datasets will be floated soon*
- Each group should consist of 1-2 students?
- **Best Project Award**
- You need to
  - develop models
  - evaluate your models
  - prepare presentation
  - write tech report

Students are encouraged to publish their projects in good conferences/journals

Deliverables:

1. Final project report (**15%**), 8 pages ACL format. Encouraged to arxiv
2. Repo of dataset and source code (**5%**)
3. Final project presentation (**10%**)

* You are welcome to propose a new idea if you find it fascinating to be qualified for a course project. Instructor opines!

Do Not Plagiarize !

Academic Integrity is of utmost importance. If anyone is found cheating/plagiarizing, it will result in negative penalty (and possibly even more: an F grade or even DisCo).

Collaborate. But do NOT cheat.

- Assignments to be done individually.
- Do not share any part of code.
- Do not copy any part of report from any online resources or published works.
- If you reuse other's works, always cite.
- If you discuss with others about assignment or outside your group for project, mention their names in the report.
- Do not use GenAI tools (like, ChatGPT).

We will check for pairwise plagiarism in submitted assignment code files among you all.

We will also check the probability of any submitted content being AI generated.

Project reports will be checked for plagiarism across all web resources.

Course Content

- This is an **advanced graduate course** and we will be teaching and discussing state-of-the-art papers about large language models.
- The course is mostly presentation- and discussion-based and all the students are expected to come to the class regularly and participate in discussion

Course Content

Basics

- Intro to NLP
- Intro to Deep Learning
- Intro to Language Models (LMs)
- Word Embeddings (Word2Vec, GloVE)
- Neural LMs (CNN, RNN, Seq2Seq, Attention)

Course Content

Basics

- Intro to NLP
- Intro to Deep Learning
- Intro to Language Models (LMs)
- Word Embeddings (Word2Vec, GloVE)
- Neural LMs (CNN, RNN, Seq2Seq, Attention)

Architecture

- Intro to Transformer
- Decoder-only LM, Prefix LM, Decoding strategies
- Encoder-only LM, Encoder-decoder LM
- Advanced Attention
- Mixture of Experts

Course Content

Basics	Architecture	Learnability
• Intro to NLP • Intro to Deep Learning • Intro to Language Models (LMs) • Word Embeddings (Word2Vec, GloVE) • Neural LMs (CNN, RNN, Seq2Seq, Attention)	• Intro to Transformer • Decoder-only LM, Prefix LM, Decoding strategies • Encoder-only LM, Encoder-decoder LM • Advanced Attention • Mixture of Experts	• Scaling laws • Instruction fine-tuning • In-context learning • Alignment • Distillation and PEFT • Efficient/Constraint LM inference

Course Content

Basics	Architecture	Learnability	User Acceptability
• Intro to NLP • Intro to Deep Learning • Intro to Language Models (LMs) • Word Embeddings (Word2Vec, GloVE) • Neural LMs (CNN, RNN, Seq2Seq, Attention)	• Intro to Transformer • Decoder-only LM, Prefix LM, Decoding strategies • Encoder-only LM, Encoder-decoder LM • Advanced Attention • Mixture of Experts	• Scaling laws • Instruction fine-tuning • In-context learning • Alignment • Distillation and PEFT • Efficient/Constraint LM inference	• RAG • Multilingual LMs • Tool-augmented LMs • Reasoning • Vision Language Models • Handling long context • Model editing

Course Content

Basics	Architecture	Learnability	User Acceptability	Ethics and Misc.
• Intro to NLP • Intro to Deep Learning • Intro to Language Models (LMs) • Word Embeddings (Word2Vec, GloVE) • Neural LMs (CNN, RNN, Seq2Seq, Attention)	• Intro to Transformer • Decoder-only LM, Prefix LM, Decoding strategies • Encoder-only LM, Encoder-decoder LM • Advanced Attention • Mixture of Experts	• Scaling laws • Instruction fine-tuning • In-context learning • Alignment • Distillation and PEFT • Efficient/Constraint LM inference	• RAG • Multilingual LMs • Tool-augmented LMs • Reasoning • Vision Language Models • Handling long context • Model editing	• Bias, toxicity and hallucination • Interpretability • Beyond Transformer: State Space Models

Pre-Requisites

- Excitement about language!
- Willingness to learn

Pre-Requisites

- Excitement about language!
- Willingness to learn

Mandatory	Desirable
• Data Structures & Algorithms • Machine Learning • Python programming	• NLP • Deep learning

Pre-Requisites

- Excitement about language!
- Willingness to learn

Mandatory	Desirable
• Data Structures & Algorithms • Machine Learning • Python programming	• NLP • Deep learning

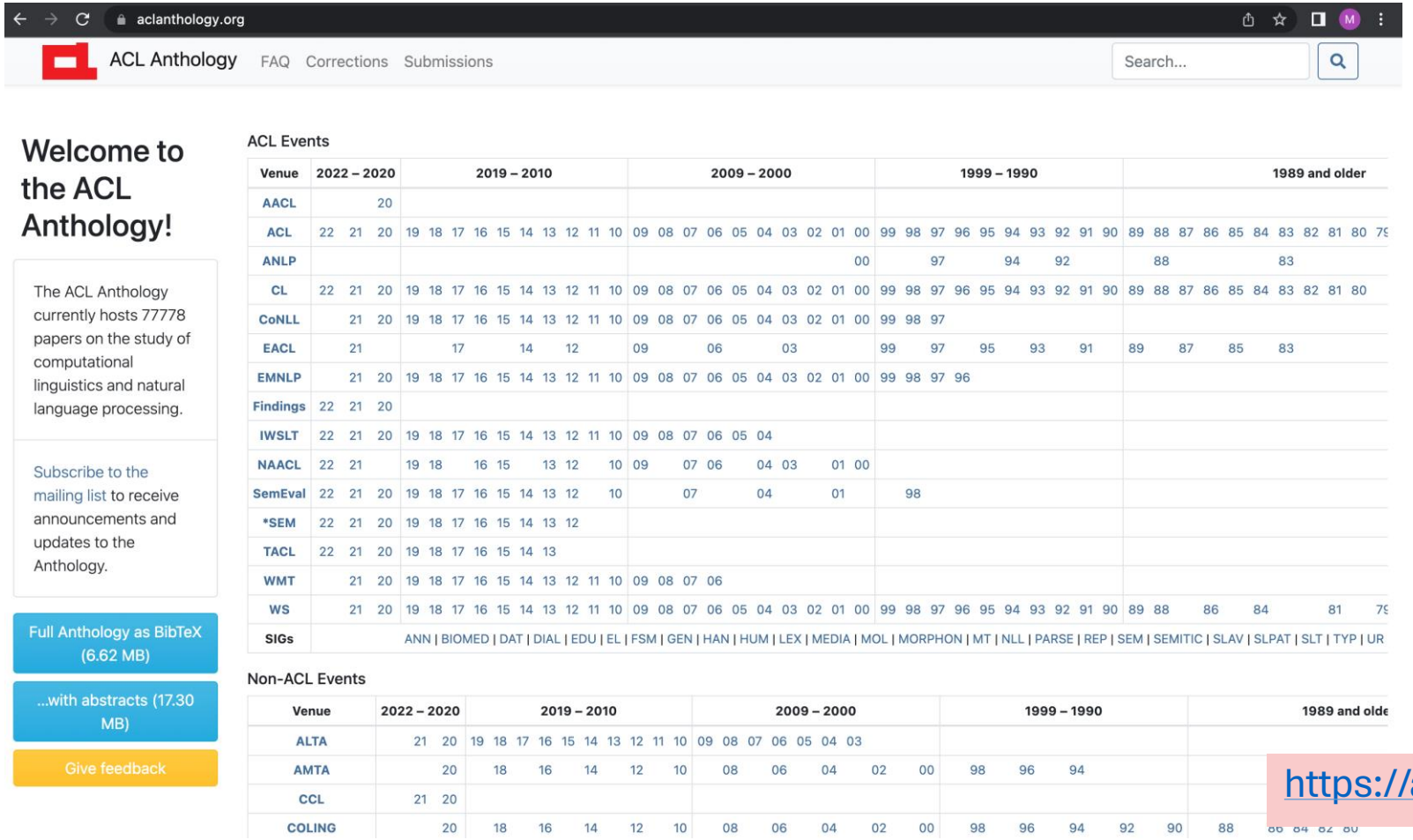
This course will NOT cover:

- Details of NLP, Machine Learning and Deep Learning
- Coding practice
- Generative models for image/vision

Reading and Reference Materials

- Books
  - Speech and Language Processing, Dan Jurafsky and James H. Martin
    https://web.stanford.edu/~jurafsky/slp3/
  - Foundations of Statistical Natural Language Processing, Chris Manning and Hinrich Schütze
  - Natural Language Processing, Jacob Eisenstein
    https://github.com/jacobeisenstein/gt-nlp-class/blob/master/notes/eisenstein-nlp-notes.pdf
  - A Primer on Neural Network Models for Natural Language Processing, Yoav Goldberg
    http://u.cs.biu.ac.il/~yogo/nnlp.pdf
- Journals
  - Computational Linguistics, Natural Language Engineering, TACL, JMLR, TMLR, etc.
- Conferences
  - ACL, EMNLP, NAACL, COLING, AAAI, IJCNLP, ICML, NeurIPS, ICLR, WWW, KDD, SIGIR, etc.

Research Papers Repository



https://aclanthology.org/

Research Papers Repository

arXiv.org > cs > cs.CL

Computation and Language

Authors and titles for recent submissions

- Wed, 19 Aug 2020
- Tue, 18 Aug 2020
- Mon, 17 Aug 2020
- Fri, 14 Aug 2020
- Thu, 13 Aug 2020

[total of 84 entries: **1-25** | 26-50 | 51-75 | 76-84]
[showing 25 entries per page: fewer | more | all]

Wed, 19 Aug 2020

[1] arXiv:2008.07905 [pdf, other]
Glancing Transformer for Non-Autoregressive Neural Machine Translation
Lihua Qian, Hao Zhou, Yu Bao, Mingxuan Wang, Lin Qiu, Weinan Zhang, Yong Yu, Lei Li
Comments: 11 pages, 3 figures, 4 tables
Subjects: **Computation and Language (cs.CL)**

[2] arXiv:2008.07880 [pdf, other]
COVID-SEE: Scientific Evidence Explorer for COVID-19 Related Research
Karin Verspoor, Simon Šuster, Yulia Otmakhova, Shevon Mendis, Zenan Zhai, Biaoyan Fang, Jey Han Lau, Timothy Bal
Comments: COVID-SEE is available at this http URL
Subjects: **Computation and Language (cs.CL)**; Information Retrieval (cs.IR)

[3] arXiv:2008.07772 [pdf, other]
Very Deep Transformers for Neural Machine Translation
Xiaodong Liu, Kevin Duh, Liyuan Liu, Jianfeng Gao
Comments: 6 pages, 3 figures and 3 tables
Subjects: **Computation and Language (cs.CL)**

[4] arXiv:2008.07723 [pdf, other]
NASE: Learning Knowledge Graph Embedding for Link Prediction via Neural Architecture Search
Xiaoyu Kou, Bingfeng Luo, Huang Hu, Yan Zhang
Comments: Accepted by CIKM 2020, short paper
Subjects: **Computation and Language (cs.CL)**

https://arxiv.org/list/cs.CL/recent

Acknowledgements (Non-Exhaustive List)

- Advanced NLP, Graham Neubig http://www.phontron.com/class/anlp2022/
- Advanced NLP, Mohit Iyyer https://people.cs.umass.edu/~miyyer/cs685/
- NLP with Deep Learning, Chris Manning, http://web.stanford.edu/class/cs224n/
- Understanding Large Language Models, Danqi Chen https://www.cs.princeton.edu/courses/archive/fall22/cos597G/
- Natural Language Processing, Greg Durrett https://www.cs.utexas.edu/~gdurrett/courses/online-course/materials.html
- Large Language Models: https://stanford-cs324.github.io/winter2022/
- Natural Language Processing at UMBC, https://laramartin.net/NLP-class/
- Computational Ethics in NLP, https://demo.clab.cs.cmu.edu/ethical_nlp/
- Self-supervised models, CS 601.471/671: Self-supervised Models (jhu.edu)
- WING.NUS Large Language Models, https://wing-nus.github.io/cs6101/
- And many more…

What is a Language Model (LM)?

Language Model gives the probability distribution over a sequence of tokens.

What is a Language Model (LM)?

Language Model gives the probability distribution over a sequence of tokens.

Vocabulary

V = {arrived, delhi, have, is, monsoon, rains, the}

What is a Language Model (LM)?

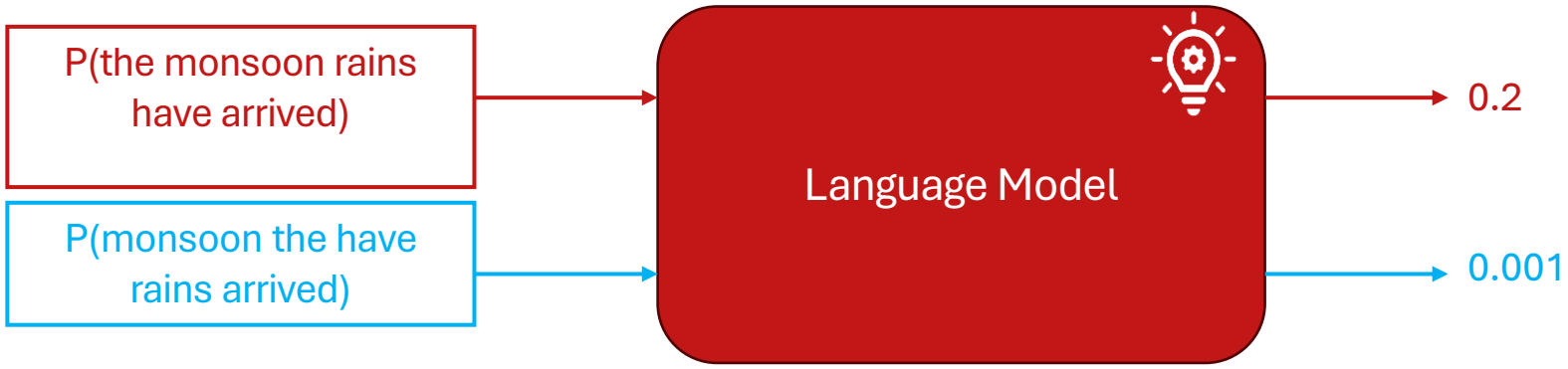
Language Model gives the probability distribution over a sequence of tokens.

Vocabulary

V = {arrived, delhi, have, is, monsoon, rains, the}

What is a Language Model (LM)?

Language Model gives the probability distribution over a sequence of tokens.



Vocabulary

V = {arrived, delhi, have, is, monsoon, rains, the}

LMs Can 'Generate' Text !

- Consider a sequence of tokens $\{x_1, x_2, \ldots, x_L\}$,where $x_1, x_2, \ldots, x_L$ are in vocabulary *V*
- Notation: $P(x_1, x_2, \ldots, x_L) = P(x_{1:L})$
- Using the **chain rule of probability:**

$$P(x_{1:L}) = P(x_1).P(x_2|x_1).P(x_3|x_1, x_2) \ldots P(x_L|x_{L-1}) = \prod_{i=1}^{L} P(x_i|x_{1:i-1})$$

LMs Can 'Generate' Text !

- Consider a sequence of tokens $\{x_1, x_2, \ldots, x_L\}$,where $x_1, x_2, \ldots, x_L$ are in vocabulary V
- Notation: $P(x_1, x_2, \ldots, x_L) = P(x_{1:L})$
- Using the **chain rule of probability:**

$$P(x_{1:L}) = P(x_1).P(x_2|x_1).P(x_3|x_1, x_2) \ldots P(x_L|x_{L-1}) = \prod_{i=1}^{L} P(x_i|x_{1:i-1})$$

Vocabulary

V = {arrived, delhi, have, is, monsoon, rains, the}

Given input 'the monsoon rains have' , LM can calculate

$P(x_i \mid \text{the monsoon rains have}) , \forall x_i \in V$

LMs Can 'Generate' Text !

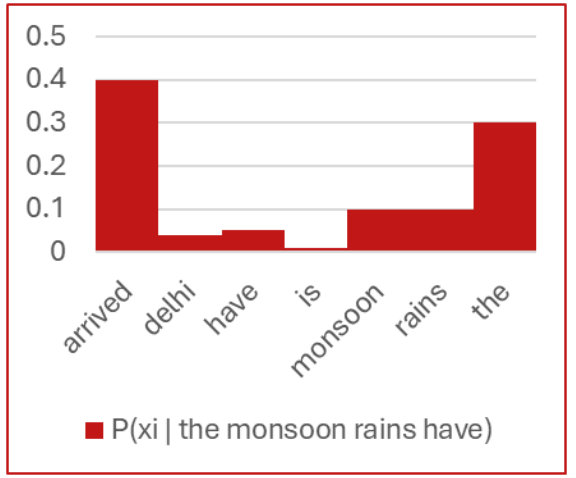
- Consider a sequence of tokens $\{x_1, x_2, \ldots, x_L\}$,where $x_1, x_2, \ldots, x_L$ are in vocabulary V
- Notation: $P(x_1, x_2, \ldots, x_L) = P(x_{1:L})$
- Using the **chain rule of probability:**

$$P(x_{1:L}) = P(x_1).P(x_2|x_1).P(x_3|x_1, x_2) \ldots P(x_L|x_{L-1}) = \prod_{i=1}^{L} P(x_i|x_{1:i-1})$$

Vocabulary

V = {arrived, delhi, have, is, monsoon, rains, the}

Given input 'the monsoon rains have' , LM can calculate $P(x_i \mid \text{the monsoon rains have}) , \forall x_i \in V$



LMs Can 'Generate' Text !

- Consider a sequence of tokens $\{x_1, x_2, \ldots, x_L\}$,where $x_1, x_2, \ldots, x_L$ are in vocabulary V
- Notation: $P(x_1, x_2, \ldots, x_L) = P(x_{1:L})$
- Using the **chain rule of probability:**

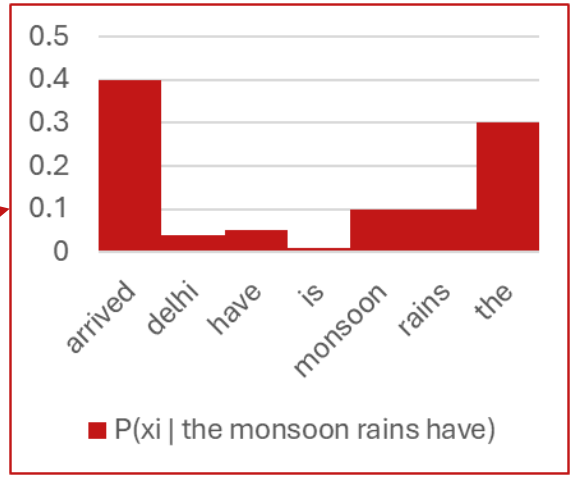
$$P(x_{1:L}) = P(x_1).P(x_2|x_1).P(x_3|x_1, x_2) \ldots P(x_L|x_{L-1}) = \prod_{i=1}^{L} P(x_i|x_{1:i-1})$$

Vocabulary

V = {arrived, delhi, have, is, monsoon, rains, the}

Given input 'the monsoon rains have' , LM can calculate $P(x_i \mid \text{the monsoon rains have}) , \forall\, x_i \in V$

For generation, next token is sampled from this probability distribution



LMs Can 'Generate' Text !

- Consider a sequence of tokens $\{x_1, x_2, \ldots, x_L\}$,where $x_1, x_2, \ldots, x_L$ are in vocabulary V
- Notation: $P(x_1, x_2, \ldots, x_L) = P(x_{1:L})$
- Using the **chain rule of probability:**

$$P(x_{1:L}) = P(x_1).P(x_2|x_1).P(x_3|x_1, x_2) \ldots P(x_L|x_{L-1}) = \prod_{i=1}^{L} P(x_i|x_{1:i-1})$$

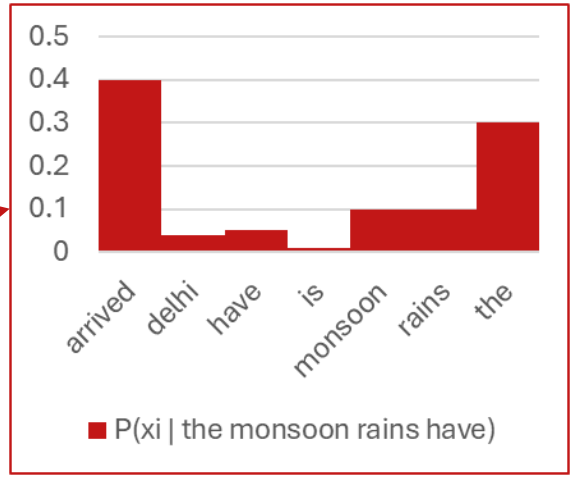
Vocabulary

V = {arrived, delhi, have, is, monsoon, rains, the}

Given input 'the monsoon rains have' , LM can calculate $P(x_i \mid \text{the monsoon rains have}) , \forall x_i \in V$

For generation, next token is sampled from this probability distribution

$$x_i \sim P(x_i \mid x_{1:i-1})$$



LMs Can 'Generate' Text !

- Consider a sequence of tokens $\{x_1, x_2, \ldots, x_L\}$,where $x_1, x_2, \ldots, x_L$ are in vocabulary V
- Notation: $P(x_1, x_2, \ldots, x_L) = P(x_{1:L})$
- Using the **chain rule of probability:**

$$P(x_{1:L}) = P(x_1).P(x_2|x_1).P(x_3|x_1, x_2) \ldots P(x_L|x_{L-1}) = \prod_{i=1}^{L} P(x_i|x_{1:i-1})$$

Vocabulary

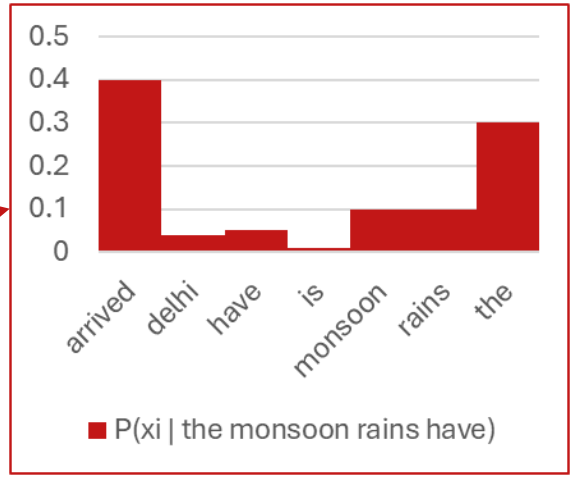
V = {arrived, delhi, have, is, monsoon, rains, the}

Given input 'the monsoon rains have' , LM can calculate $P(x_i \mid \text{the monsoon rains have}) , \forall x_i \in V$

For generation, next token is sampled from this probability distribution

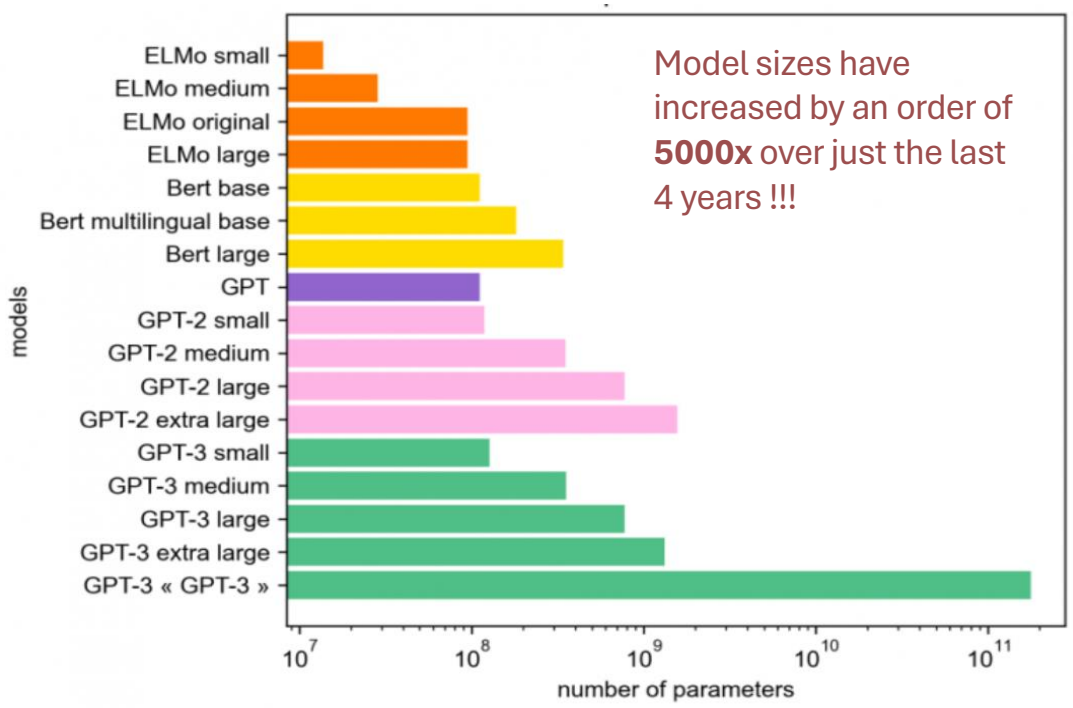
$$x_i \sim P(x_i \mid x_{1:i-1})$$

Auto-regressive LMs calculate this distribution efficiently, e.g. using 'Deep' Neural Networks



‘Large’ Language Models

The ‘Large’ in Large Language Model refers to model's size in terms of parameters as well as the massive size of training dataset.



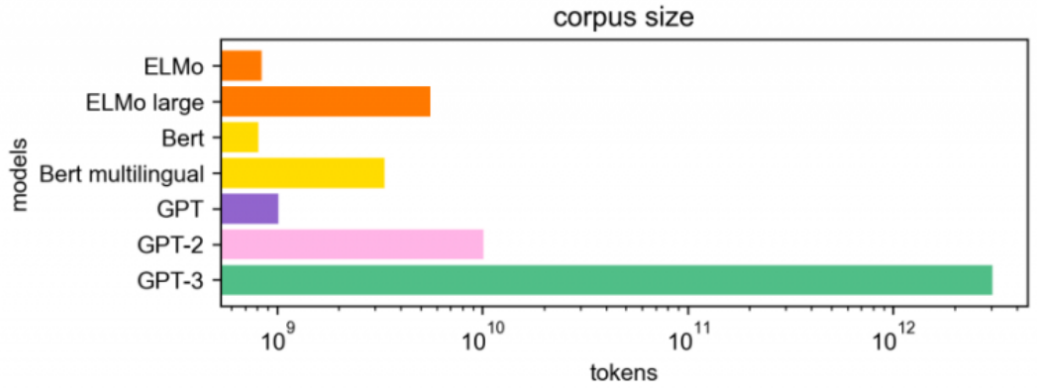
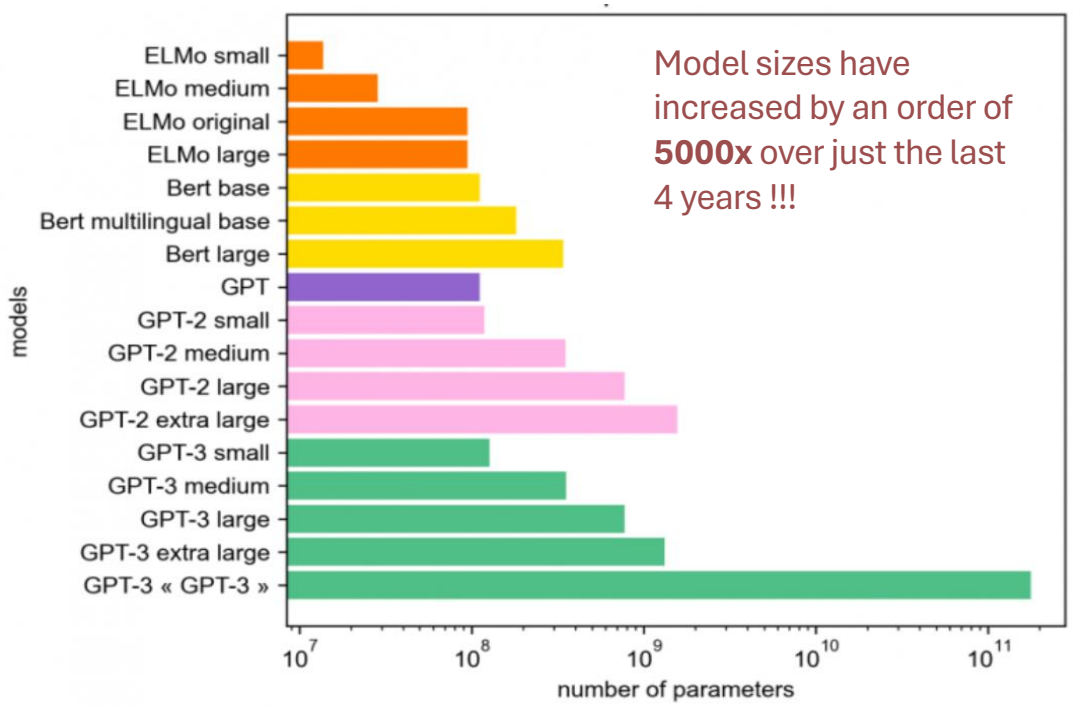
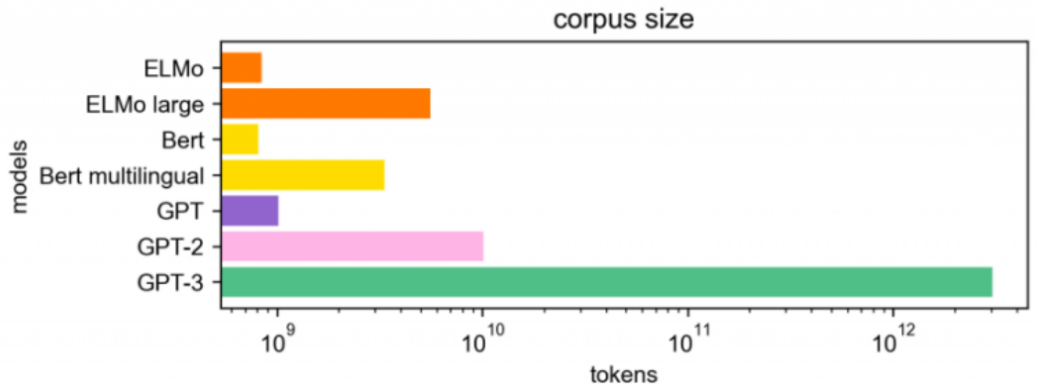


Image source: https://hellofuture.orange.com/en/the-gpt-3-language-model-revolution-or-evolution/

‘Large’ Language Models

The ‘Large’ in Large Language Model refers to model's size in terms of parameters as well as the massive size of training dataset.





Other recent models: PaLM (540B), OPT (175B), BLOOM (176B), Gemini-Ultra (1.56T), GPT-4 (1.76T)

Disclaimer: For API-based models like GPT-4/Gemini-Ultra, the number of parameters are not announced officially – these are rumored numbers as on the web

Image source: https://hellofuture.orange.com/en/the-gpt-3-language-model-revolution-or-evolution/

LLMs in AI Landscape

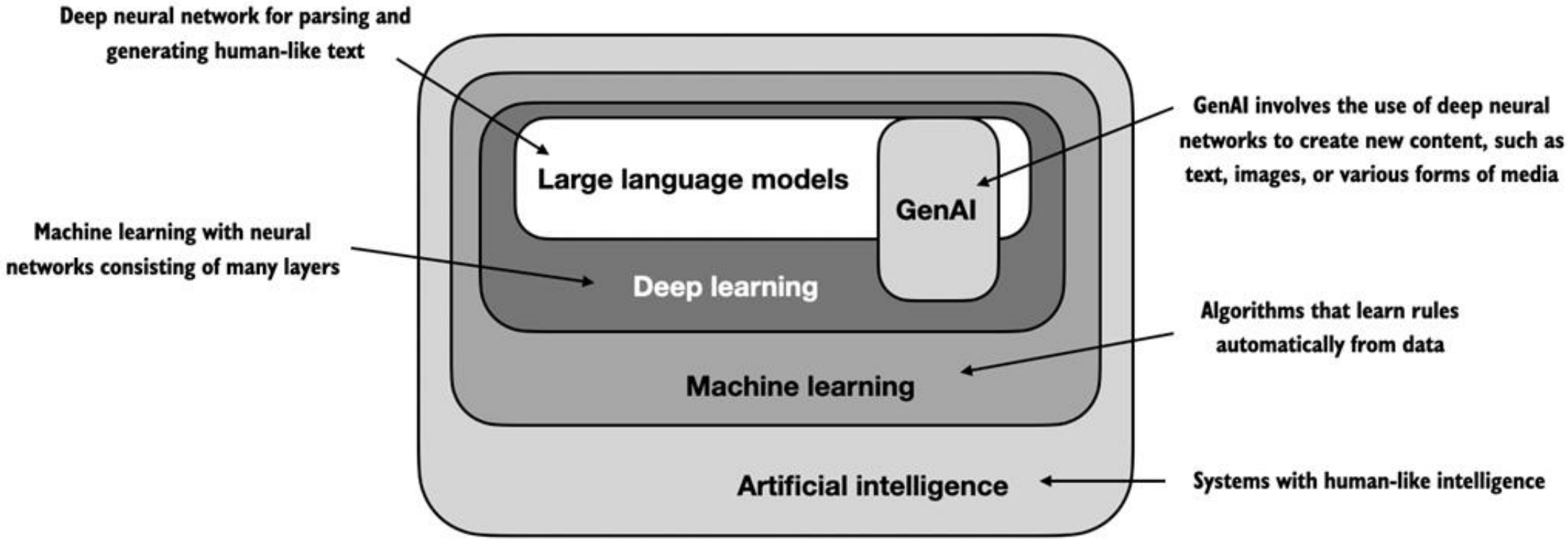


Image source: https://www.manning.com/books/build-a-large-language-model-from-scratch

Evolution of (L)LMs

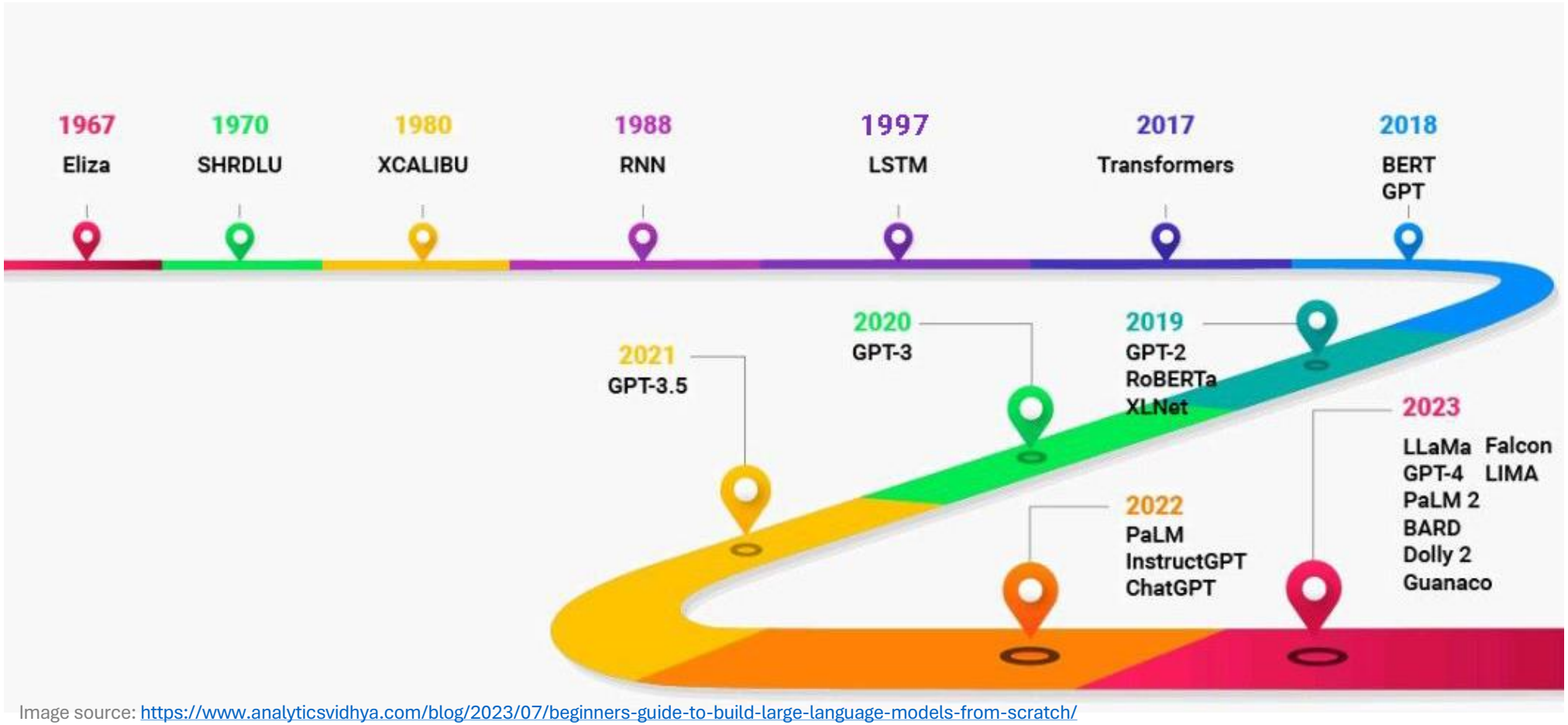


Image source: https://www.analyticsvidhya.com/blog/2023/07/beginners-guide-to-build-large-language-models-from-scratch/

We will discuss about many of them in this course!

Why Does This Course Exist?

Why do we need a separate course on LLMs? What changes with the scale of LMs?

Content credits: https://stanford-cs324.github.io/winter2022/

Why Does This Course Exist?

Why do we need a separate course on LLMs? What changes with the scale of LMs?

Emergence

Content credits: https://stanford-cs324.github.io/winter2022/

Why Does This Course Exist?

Why do we need a separate course on LLMs? What changes with the scale of LMs?

Emergence

Although the technical machineries are almost similar, 'just scaling up' these models results in new **emergent** behaviors, which lead to significantly different capabilities and societal impacts.

Content credits: https://stanford-cs324.github.io/winter2022/

Why Does This Course Exist?

LLMs show emergent capabilities, not observed previously in ‘small’ LMs.

Content credits: https://stanford-cs324.github.io/winter2022/

Why Does This Course Exist?

LLMs show emergent capabilities, not observed previously in ‘small’ LMs.

- In-context learning: A **pre-trained language model** can be guided with **only prompts to perform different tasks (without separate task-specific fine-tuning)**.
  - In-context learning is an example of **emergent** behavior.

Content credits: https://stanford-cs324.github.io/winter2022/

Why Does This Course Exist?

LLMs show emergent capabilities, not observed previously in ‘small’ LMs.

- In-context learning: A **pre-trained language model** can be guided with **only prompts to perform different tasks (without separate task-specific fine-tuning)**.
  - In-context learning is an example of **emergent** behavior.

LLMs are widely adopted in real-world.

Content credits: https://stanford-cs324.github.io/winter2022/

Why Does This Course Exist?

LLMs show emergent capabilities, not observed previously in ‘small’ LMs.

- In-context learning: A **pre-trained language model** can be guided with **only prompts to perform different tasks (without separate task-specific fine-tuning)**.
  - In-context learning is an example of **emergent** behavior.

LLMs are widely adopted in real-world.

- **Research**: LLMs have transformed **NLP research** world, achieving state-of-the-art performance across a wide range of tasks such as sentiment classification, question answering, summarization, and machine translation.

Content credits: https://stanford-cs324.github.io/winter2022/

Why Does This Course Exist?

LLMs show emergent capabilities, not observed previously in 'small' LMs.

- In-context learning: A **pre-trained language model** can be guided with **only prompts to perform different tasks (without separate task-specific fine-tuning)**.
  - In-context learning is an example of **emergent** behavior.

LLMs are widely adopted in real-world.

- **Research**: LLMs have transformed **NLP research** world, achieving state-of-the-art performance across a wide range of tasks such as sentiment classification, question answering, summarization, and machine translation.
- **Industry**: Here is a very incomplete list of some high profile large language models that are being used in **production systems**:
  - Google Search (BERT)
  - Facebook content moderation (XLM)
  - Microsoft's Azure OpenAI Service (GPT-3/3.5/4)

Content credits: https://stanford-cs324.github.io/winter2022/

Why Does This Course Exist?

With tremendous capabilities, LLMs' usage also carries various **risks**.

Content credits: https://stanford-cs324.github.io/winter2022/

Why Does This Course Exist?

With tremendous capabilities, LLMs' usage also carries various **risks**.

- **Reliability & Disinformation:** LLMs often hallucinate – generate responses that *seem correct*, but are not factually correct.
  - Significant challenge for high-stakes applications like healthcare

Content credits: https://stanford-cs324.github.io/winter2022/

Why Does This Course Exist?

With tremendous capabilities, LLMs' usage also carries various **risks**.

- **Reliability & Disinformation:** LLMs often hallucinate – generate responses that *seem correct*, but are not factually correct.
  - Significant challenge for high-stakes applications like healthcare
- **Social bias**: Most LLMs show performance disparities across demographic groups, and their predictions can enforce stereotypes.
  - P(**He** is a doctor) > P(**She** is a doctor.)
  - Training data contains inherent bias

Content credits: https://stanford-cs324.github.io/winter2022/

Why Does This Course Exist?

With tremendous capabilities, LLMs' usage also carries various **risks**.

- **Reliability & Disinformation:** LLMs often hallucinate – generate responses that *seem correct*, but are not factually correct.
  - Significant challenge for high-stakes applications like healthcare
- **Social bias**: Most LLMs show performance disparities across demographic groups, and their predictions can enforce stereotypes.
  - P(**He** is a doctor) > P(**She** is a doctor.)
  - Training data contains inherent bias
- **Toxicity**: LLMs can generate toxic/hateful content.
  - Trained on a huge amount of Internet data (e.g., Reddit), which inevitably contains offensive content
  - Challenge for applications such as writing assistants or chatbots

Content credits: https://stanford-cs324.github.io/winter2022/

Why Does This Course Exist?

With tremendous capabilities, LLMs' usage also carries various **risks**.

- **Reliability & Disinformation:** LLMs often hallucinate – generate responses that *seem correct*, but are not factually correct.
  - Significant challenge for high-stakes applications like healthcare
- **Social bias**: Most LLMs show performance disparities across demographic groups, and their predictions can enforce stereotypes.
  - P(**He** is a doctor) > P(**She** is a doctor.)
  - Training data contains inherent bias
- **Toxicity**: LLMs can generate toxic/hateful content.
  - Trained on a huge amount of Internet data (e.g., Reddit), which inevitably contains offensive content
  - Challenge for applications such as writing assistants or chatbots
- **Security**: LLMs are trained on a scrape of the public Internet - anyone can put up a website that can enter the training data.
  - An attacker can perform a **data poisoning** attack.

Content credits: https://stanford-cs324.github.io/winter2022/

We Will Cover Almost All of These in 5 Modules

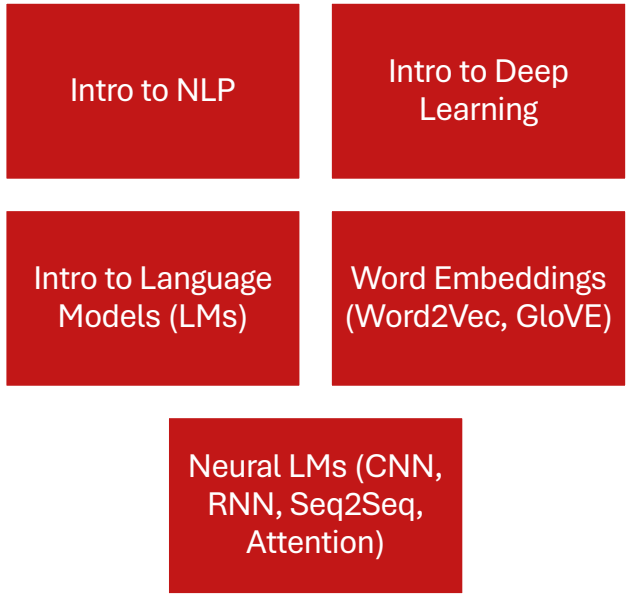
Module-1: Basics

- A **refresher on the basics of NLP and DL** required to understand and appreciate LLMs
- How did we end up in **Neural NLP**?
  - We will discuss the transition and the foundations of Neural NLP.
- The basics of **Language Modelling**
- Initial **Neural LMs**

We Will Cover Almost All of These in 5 Modules

Module-1: Basics

- A **refresher on the basics of NLP and DL** required to understand and appreciate LLMs
- How did we end up in **Neural NLP**?
  - We will discuss the transition and the foundations of Neural NLP.
- The basics of **Language Modelling**
- Initial **Neural LMs**

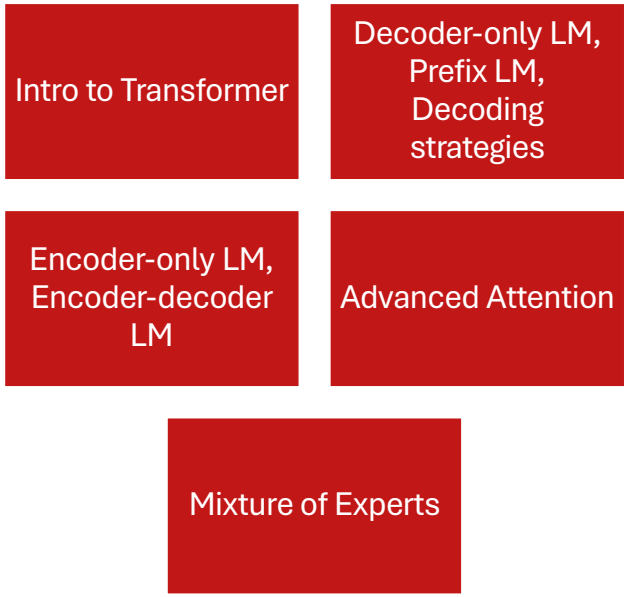


We Will Cover Almost All of These in 5 Modules

- Module-2: Architecture
  - Workings of **Vanilla Transformers**
  - Different **Transformer Variants**
    - How do their training strategies differ? How are Masked LMs (like, BERT) different from Auto-regressive LMs (like, GPT)?
  - **Response generation (Decoding) strategies**
  - What makes modern open-source LLMs like LLaMA & Mistral more effective over vanilla transformers?
    - An in-depth exploration of the **advanced attention mechanisms**
  - **Mixture-of-Experts**: an effective architectural choice in modern LLMs

We Will Cover Almost All of These in 5 Modules

- Module-2: Architecture
  - Workings of **Vanilla Transformers**
  - Different **Transformer Variants**
    - How do their training strategies differ? How are Masked LMs (like, BERT) different from Auto-regressive LMs (like, GPT)?
  - **Response generation (Decoding) strategies**
  - What makes modern open-source LLMs like LLaMA & Mistral more effective over vanilla transformers?
    - An in-depth exploration of the **advanced attention mechanisms**
  - **Mixture-of-Experts**: an effective architectural choice in modern LLMs

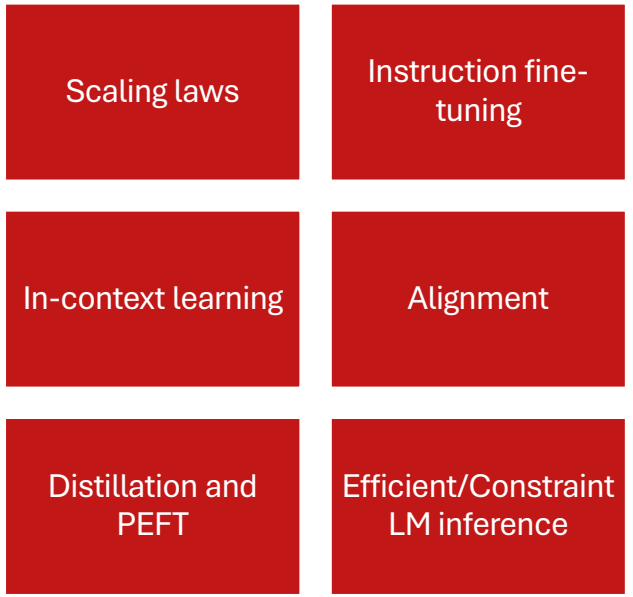


We Will Cover Almost All of These in 5 Modules

- Module-3: Learnability
  - **Scaling Laws**: how does performance vary with scale of LMs? When does ‘emergence’ kick in?
  - What makes modern LLMs so good in following user instructions?
  - What is **In-context Learning**? What are its various facets?
  - How are LLMs made to **generate responses preferred by humans**?
    - Does it remove toxicity in responses?
  - **Efficiency** is crucial in production systems.
    - How are smaller LMs made capable using pre-trained LLMs?
    - How are LLMs efficiently fine-tuned?
    - How are response generation latency of LLMs improved?

We Will Cover Almost All of These in 5 Modules

- Module-3: Learnability
  - **Scaling Laws**: how does performance vary with scale of LMs? When does 'emergence' kick in?
  - What makes modern LLMs so good in following user instructions?
  - What is **In-context Learning**? What are its various facets?
  - How are LLMs made to **generate responses preferred by humans**?
    - Does it remove toxicity in responses?
  - **Efficiency** is crucial in production systems.
    - How are smaller LMs made capable using pre-trained LLMs?
    - How are LLMs efficiently fine-tuned?
    - How are response generation latency of LLMs improved?

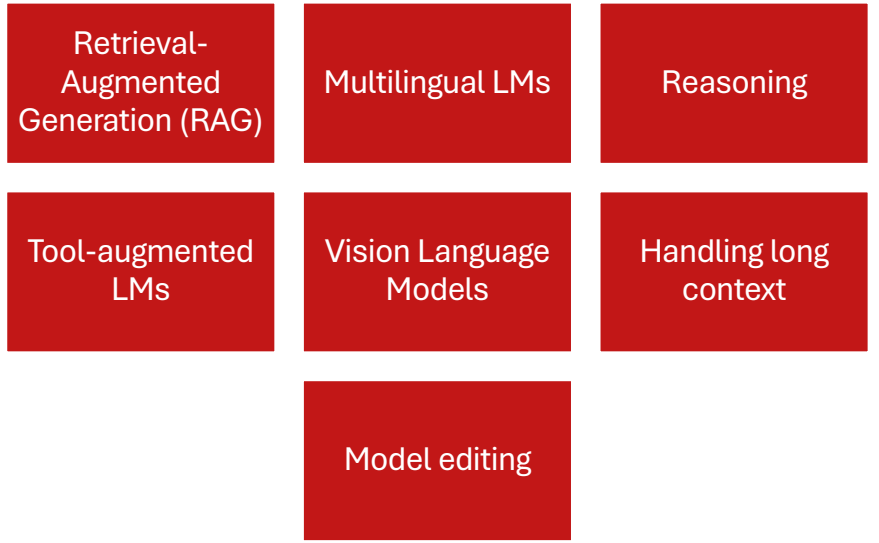


We Will Cover Almost All of These in 5 Modules

- Module-4: User Acceptability
  - How can we make LLMs aware of certain relevant facts while generation?
  - Can LLMs operate in **multiple languages**?
  - Can LLMs reason?
  - Can **usage of external tools** help LLMs perform better?
  - Can LLMs handle **multiple modalities, like image**?
    - What changes are required in their architecture to do so?
  - How much long inputs can LLMs candle?
    - How can we **increase their context length**?
  - Can we **edit model components** to mitigate certain issues in LLMs?

We Will Cover Almost All of These in 5 Modules

- Module-4: User Acceptability
  - How can we make LLMs aware of certain relevant facts while generation?
  - Can LLMs operate in **multiple languages**?
  - Can LLMs reason?
  - Can **usage of external tools** help LLMs perform better?
  - Can LLMs handle **multiple modalities, like image**?
    - What changes are required in their architecture to do so?
  - How much long inputs can LLMs candle?
    - How can we **increase their context length**?
  - Can we **edit model components** to mitigate certain issues in LLMs?

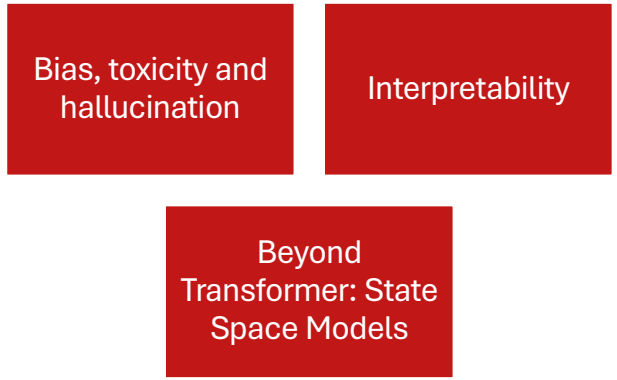


We Will Cover Almost All of These in 5 Modules

- Module-5: Ethics and Miscellaneous
  - A discussion on **ethical issues** and **risks** of LLM usage
  - How are different emergent abilities in LLMs facilitated?
    - A peep into the internal workings of LLMs to understand the source of their capabilities
  - Can LMs based on alternate architecture match Transformer-based LLMs?
    - **State-Space Models (SSMs)**

We Will Cover Almost All of These in 5 Modules

- Module-5: Ethics and Miscellaneous
  - A discussion on **ethical issues** and **risks** of LLM usage
  - How are different emergent abilities in LLMs facilitated?
    - A peep into the internal workings of LLMs to understand the source of their capabilities
  - Can LMs based on alternate architecture match Transformer-based LLMs?
    - **State-Space Models (SSMs)**



Suggestions (For Effective Learning)

- To understand the concepts clearly, experiment with the models (**Hugging Face** makes life easier).
- Smaller models (like, GPT2) can be run on **Google Colab** / **Kaggle**.
  - Even 7B models can be run with proper quantization.

Always **get your hands dirty !**

LLM Research is all about implementing and experimenting with your ideas.

Rule of thumb: Do not believe in any hypothesis until your experiments verify it !