Making Your Apps Smarter: Machine Learning & AI

Shan-Hung Wu
shwu@cs.nthu.edu.tw

Department of Computer Science,
National Tsing Hua University, Taiwan

Software Design & Studio

Outline

1. Web, App, and Business Intelligence
2. What's Machine Learning & AI?
3. What's Deep Learning?
4. What's Generative AI?
5. Making Smart Apps

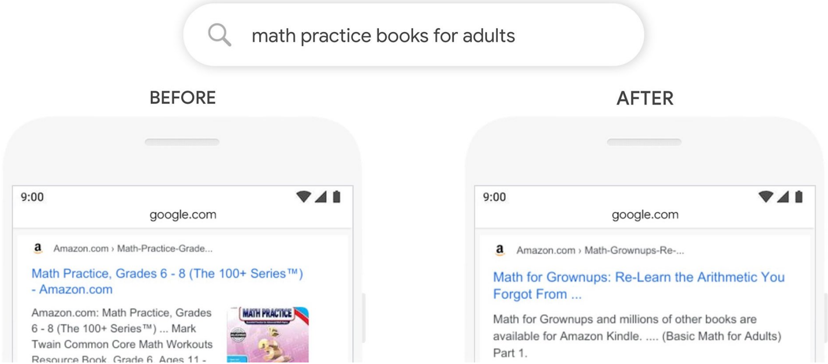
Outline

1. Web, App, and Business Intelligence
2. What's Machine Learning & AI?
3. What's Deep Learning?
4. What's Generative AI?
5. Making Smart Apps

Your Term Project

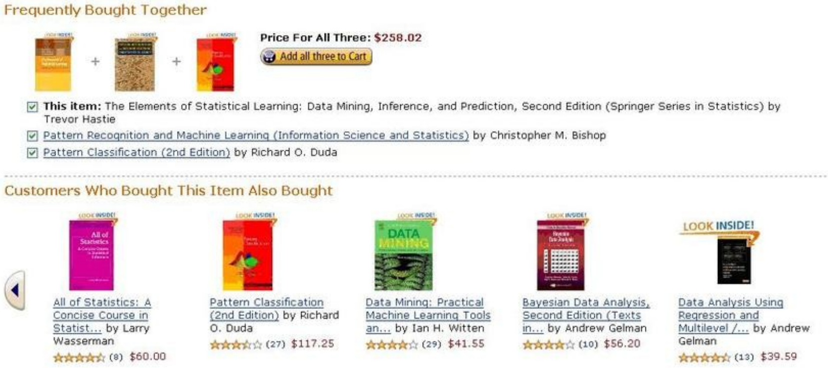
To ***design & implement*** an ***intellectual app*** that solves ***real problems***.

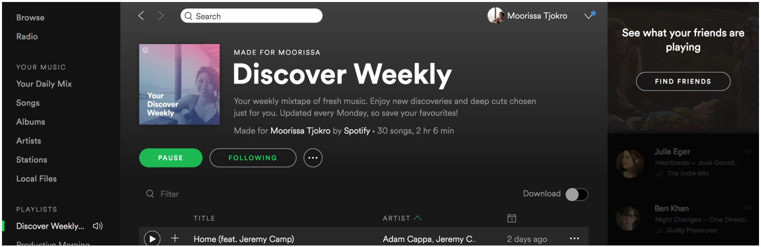
Example: Intention Identification



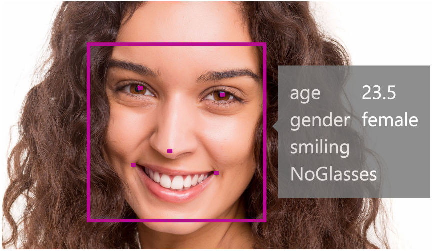
Example: Spam Detection

Example: Product Recommendation

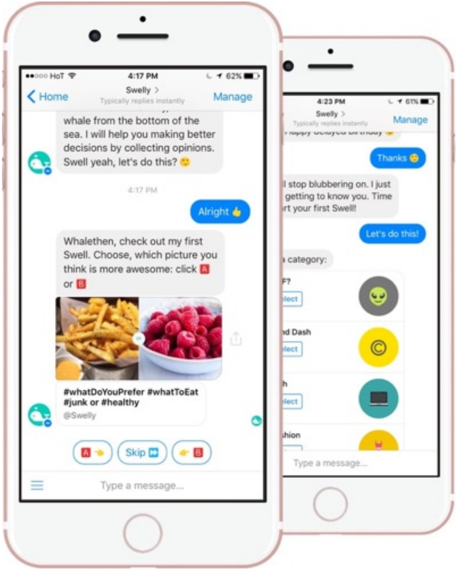


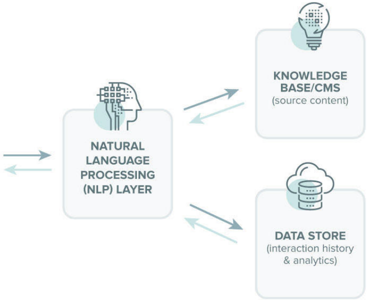


Example: Image Understanding

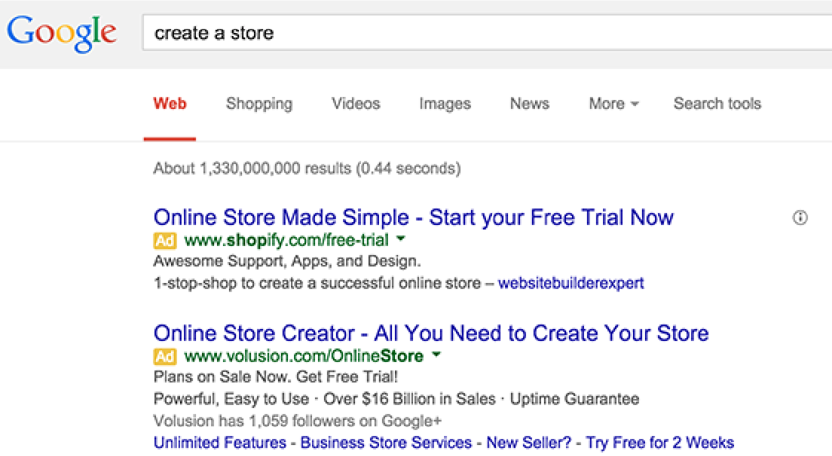


Example: Chat Bot & Service Automation





Example: Marketing & Advertisement



- All based on ***Machine Learning*** & ***AI*** technologies

Outline

AI vs. Machine Learning

- Artificial Intelligence (AI): the goal
  - Creating systems that can function intelligently and independently
  - Mirroring or surpassing human capabilities
- Machine Learning (ML): a means of achieving AI
  - Enabling machines to ***learn from data***

Prior vs. Posteriori Knowledge

- To solve a problem, we need an algorithm
  - E.g., sorting

Prior vs. Posteriori Knowledge

- To solve a problem, we need an algorithm
  - E.g., sorting
  - ***A priori knowledge*** is enough

Prior vs. Posteriori Knowledge

- To solve a problem, we need an algorithm
    - E.g., sorting
    - ***A priori knowledge*** is enough
- For some problem, however, we do not have the a priori knowledge
    - E.g., to tell if a post is toxic or not
    - The correct answer varies in time and from site to site

Prior vs. Posteriori Knowledge

- To solve a problem, we need an algorithm
  - E.g., sorting
  - ***A priori knowledge*** is enough
- For some problem, however, we do not have the a priori knowledge
  - E.g., to tell if a post is toxic or not
  - The correct answer varies in time and from site to site
- Machine learning algorithms use the ***a posteriori knowledge*** to solve problems

Prior vs. Posteriori Knowledge

- To solve a problem, we need an algorithm
  - E.g., sorting
  - ***A priori knowledge*** is enough
- For some problem, however, we do not have the a priori knowledge
  - E.g., to tell if a post is toxic or not
  - The correct answer varies in time and from site to site
- Machine learning algorithms use the ***a posteriori knowledge*** to solve problems
  - Takes ***examples*** as additional input to algorithm

General ML Step 1: Data Preparation

- Pre-process data (e.g., integration, cleaning, etc.)
- Define vector ***features*** to have a dataset:

$$\mathbb{X} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N}, \text{ where } \boldsymbol{x}^{(i)} \in \mathbb{R}^D \text{ and } \boldsymbol{y}^{(i)} \in \mathbb{R}^K,$$

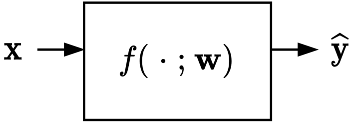
- E.g., in toxic post detection:
  - $\boldsymbol{x}^{(i)}$ represents counts of different tokens
  - $y^{(i)} \in \{0, 1\}$ indicates if the post $\boldsymbol{x}^{(i)}$ is toxic or not

General ML Step 2: Model Development

$$\mathbf{x} \longrightarrow \boxed{f(\,\cdot\,;\mathbf{w})} \longrightarrow \hat{\mathbf{y}}$$

1. Assume a ***model*** $\{f(\cdot;\boldsymbol{w})\}_{\boldsymbol{w}}$ that is a collection of candidate functions f's
   - Each f predicts label $\hat{\boldsymbol{y}}$ given an input $\boldsymbol{x}$
   - f is assumed to be parametrized by $\boldsymbol{w}$

General ML Step 2: Model Development



1. Assume a ***model*** $\{f(\cdot;\boldsymbol{w})\}_{\boldsymbol{w}}$ that is a collection of candidate functions f's
   - Each f predicts label $\hat{y}$ given an input $\boldsymbol{x}$
   - f is assumed to be parametrized by $\boldsymbol{w}$
2. Define a ***cost function***

$$C(\boldsymbol{w};\mathbb{X})$$

   that measures "how good a particular $f(\cdot;\boldsymbol{w})$ can explain the training data $\mathbb{X}$" (posteriori knowledge)

General ML Step 2: Model Development

$$\mathbf{x} \longrightarrow \boxed{f(\,\cdot\,;\mathbf{w})} \longrightarrow \widehat{\mathbf{y}}$$

1. Assume a ***model*** $\{f(\cdot;\boldsymbol{w})\}_{\boldsymbol{w}}$ that is a collection of candidate functions f's
   - Each f predicts label $\hat{\boldsymbol{y}}$ given an input $\boldsymbol{x}$
   - f is assumed to be parametrized by $\boldsymbol{w}$
2. Define a ***cost function***

   $$C(\boldsymbol{w};\mathbb{X})$$

   that measures "how good a particular $f(\cdot;\boldsymbol{w})$ can explain the training data $\mathbb{X}$" (posteriori knowledge)
3. ***Training***: employ an algorithm that solves

   $$\boldsymbol{w}^* = \arg\min_{\boldsymbol{w}} C(\boldsymbol{w};\mathbb{X})$$

   - Where "learning" happens

General ML Step 3: Testing & Deployment

1. ***Testing***: evaluate the performance of the learned $f(\cdot;\boldsymbol{w}^*)$ using another, ***unseen*** test dataset $\mathbb{X}'$
   - Examples in $\mathbb{X}'$ should have the same distribution with those in $\mathbb{X}$
   - A model minimizing $C(\boldsymbol{w};\mathbb{X})$ does ***not*** necessarily give hight test performance
2. If $f(\cdot;\boldsymbol{w}^*)$ has satisfactory test performance, deploy it to solve real-world problem

Outline

What is Deep Learning?

- ML where an $f(\cdot;\boldsymbol{w})$ has many (deep) layers

$$\hat{\boldsymbol{y}} = f^{(L)}(\cdots f^{(2)}(f^{(1)}(\boldsymbol{x};\boldsymbol{w}^{(1)});\boldsymbol{w}^{(2)})\cdots;\boldsymbol{w}^{(L)})$$

$$\mathbf{x} \rightarrow \boxed{f^{(1)}\left(\cdot\,;\,\mathbf{w}^{(1)}\right)} \rightarrow \boxed{f^{(2)}\left(\cdot\,;\,\mathbf{w}^{(2)}\right)} \rightarrow \cdots \boxed{f^{(L)}\left(\cdot\,;\,\mathbf{w}^{(L)}\right)} \rightarrow \widehat{\mathbf{y}}$$

What is Deep Learning?

- ML where an $f(\cdot;\boldsymbol{w})$ has many (deep) layers

$$\hat{\boldsymbol{y}} = f^{(L)}(\cdots f^{(2)}(f^{(1)}(\boldsymbol{x};\boldsymbol{w}^{(1)});\boldsymbol{w}^{(2)})\cdots;\boldsymbol{w}^{(L)})$$

$$\mathbf{x} \rightarrow \boxed{f^{(1)}\left(\cdot\,;\mathbf{w}^{(1)}\right)} \rightarrow \boxed{f^{(2)}\left(\cdot\,;\mathbf{w}^{(2)}\right)} \rightarrow \cdots \boxed{f^{(L)}\left(\cdot\,;\mathbf{w}^{(L)}\right)} \rightarrow \hat{\mathbf{y}}$$

- Pros:
  - Learns features from raw data automatically, called ***representation learning***
  - Learns a complex function (e.g., visual objects to labels)

What is Deep Learning?

- ML where an $f(\cdot;\boldsymbol{w})$ has many (deep) layers

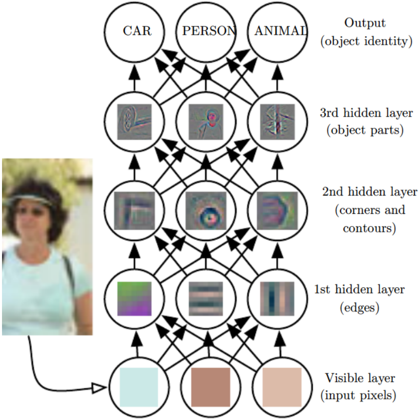
$$\hat{\boldsymbol{y}} = f^{(L)}(\cdots f^{(2)}(f^{(1)}(\boldsymbol{x};\boldsymbol{w}^{(1)});\boldsymbol{w}^{(2)})\cdots;\boldsymbol{w}^{(L)})$$

$$\mathbf{x} \rightarrow \boxed{f^{(1)}\left(\,\cdot\,;\mathbf{w}^{(1)}\right)} \rightarrow \boxed{f^{(2)}\left(\,\cdot\,;\mathbf{w}^{(2)}\right)} \rightarrow \cdots \boxed{f^{(L)}\left(\,\cdot\,;\mathbf{w}^{(L)}\right)} \rightarrow \hat{\mathbf{y}}$$

- Pros:
  - Learns features from raw data automatically, called ***representation learning***
  - Learns a complex function (e.g., visual objects to labels)
- Cons:
  - Usually needs large data to train a model well
  - High computation costs (at both training and test time); needs GPU acceleration

Representation Learning

- Automatically learned features also called ***embeddings***
- Helps understanding what's learned
- Also enable new ways of using deep models
  - To be discussed later



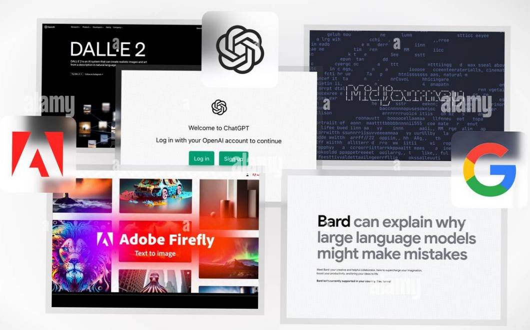
Outline

Generative AI

- The goal
- To generate ***structural*** and ***novel*** output (such as images, text, music, etc.) that cannot be deemed fake by humans

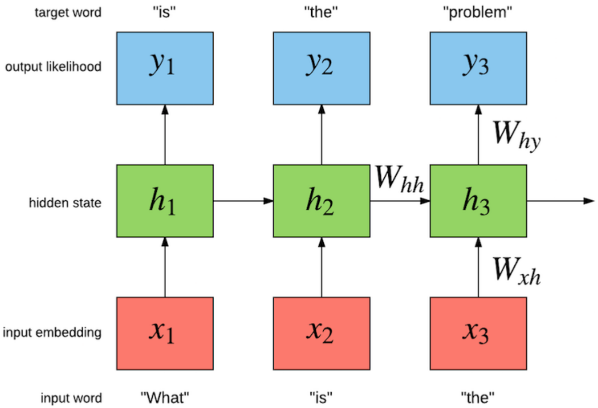
Generative Models

- The means
- Dataset: $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N}$, where $\boldsymbol{y}^{(i)}$ can be as complex as $\boldsymbol{x}^{(i)}$ and *cannot be exhausted*
- Image generation models (e.g., diffusion models)
- Text/language generation models (e.g., GPTs)
- Cross-modal generation models (e.g., GPT4 with vision)

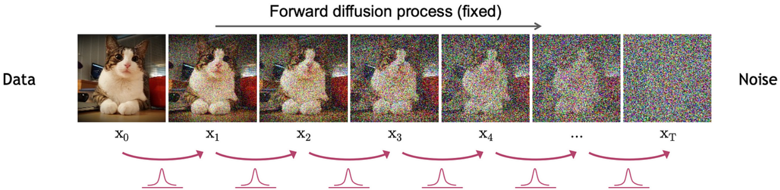


Autoregressive Models for Text Generation

- Text can be considered as 1D time-series data
- ***Autoregressive model*** takes its previous output as current input
  - Learns the conditional transition distributions of tokens
  - Rather than the join distribution of all tokens



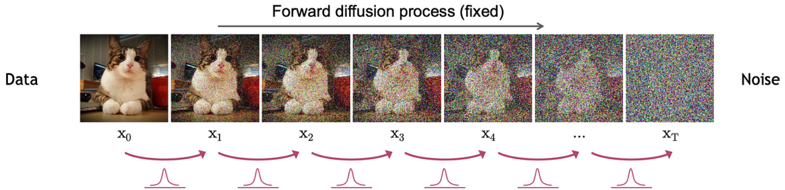
Diffusion Models for Image Generation



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \quad \text{(joint)}$$

- Note a time-series data naturally; need other strategy to simplify learning

Diffusion Models for Image Generation



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t \mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad \text{(joint)}$$

- Note a time-series data naturally; need other strategy to simplify learning
- Forward diffusion is stepwise and deterministic (no learning)
- Model learns to de-noise at each step to generate images
  - Input: noisy image $\boldsymbol{x}_t$, step t, and $\boldsymbol{y}$ (e.g., text prompt)
  - Output: less noisy image $\boldsymbol{x}_{t-1}$

Training Trick 1: Self-Supervised Pre-training

- Before training on $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N}$
- Pre-train model on $\mathbb{X}' = \{(\boldsymbol{x}^{(i,1)}, \boldsymbol{x}^{(i,2)})\}_{i=1}^{M}$, where $\boldsymbol{x}^{(i,1)}$ and $\boldsymbol{x}^{(i,2)}$ are parts of the same structural data point
  - Applicable to both text and images
- Why?

$\boldsymbol{x}^{(i)}$  $\boldsymbol{y}^{(i)}$

Training Trick 1: Self-Supervised Pre-training

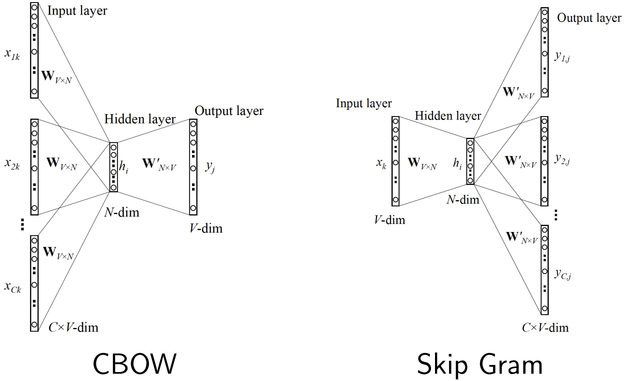
- Before training on $\mathbb{X} = \{(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})\}_{i=1}^{N}$
- Pre-train model on $\mathbb{X}' = \{(\boldsymbol{x}^{(i,1)}, \boldsymbol{x}^{(i,2)})\}_{i=1}^{M}$, where $\boldsymbol{x}^{(i,1)}$ and $\boldsymbol{x}^{(i,2)}$ are parts of the same structural data point
  - Applicable to both text and images
- Why? $M \gg N$
  - GPT-4 is trained on ~13 trillion tokens (~10 trillion words)
  - LAION has 400 million 256X256 images
- Use "common sense" to learn $\boldsymbol{y}^{(i)}$ of limited numbers

$\boldsymbol{x}^{(i)}$

$\boldsymbol{y}^{(i)}$

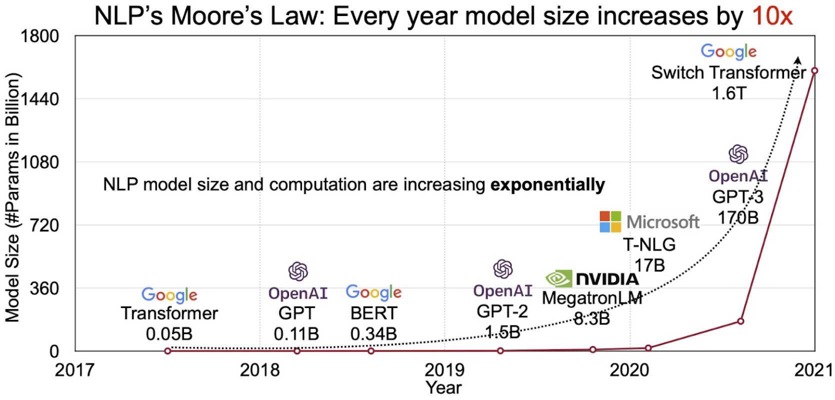
Byproduct: Semantic Embeddings

- After pre-training, embeddings of different data points have mutual distances reflecting human understanding
- E.g., word2vec [3, 2]: "... *the cat sat* ***on***..."



- Powers modern search and recommendation systems
    - Google Search, Instagram Feeds, Spotify playlists, etc.

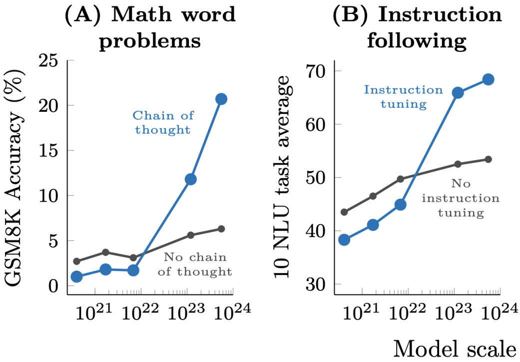
Training Trick 2: Large Models



- Training costs [4]:
  - 110M params: \$2.5k–\$50k
  - 340M params: \$10k–\$200k
  - 1.5B param: \$80k–\$1.6m

Size Does Matter!

- Emerging abilities of Large Language Models (LLMs) [5]



- A balance: 70B parameters + 1.4T training tokens [1]

Outline

1. Web, App, and Business Intelligence
2. What's Machine Learning & AI?
3. What's Deep Learning?
4. What's Generative AI?
5. **Making Smart Apps**

Using Existing ML Models

- Today, we can easily integrate the power of ML models into our apps to make impact
- Example: Flutter warpper of ML Kit from Google
  - Designed to be run ***locally*** on mobile devices
  - Supported image tasks:
    - Barcode scanning, doc scanning, face detection, image labeling, object detection, etc.
  - Supported NLP tasks:
    - Language identification, translation, entity (date/time/address/phone number) extraction, smart reply, etc.

Integrating Advanced Generative Models

- Example: OpenAI's APIs
  - Chat, image generation, embeddings, etc.
- Demo
  - Install the "http" package
  - Obtain your API key

Customizing Models (1/2)

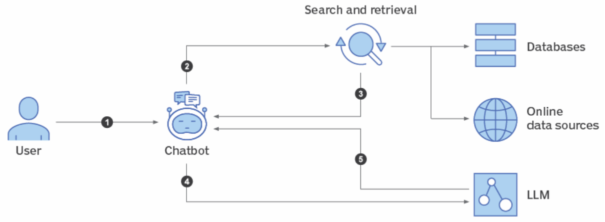
- How to customize a model for your specific tasks?

Customizing Models (1/2)

- How to customize a model for your specific tasks?
- Fine-tuning model using your own data
  - Not possible if weights are unavailable
- Write better prompts

Customizing Models (2/2)

- Enable ***Retrieval Augmented Generation*** (RAG) though Assistant API
  - Demo
  - Does not modify model's weights



- Ask model to perform "actions" defined by you via Function Calling API
- E.g., "Code Interpreter" plugin of ChatGPT

Reference I

[1] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al.
Training compute-optimal large language models.
arXiv preprint arXiv:2203.15556, 2022.

[2] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean.
Efficient estimation of word representations in vector space.
arXiv preprint arXiv:1301.3781, 2013.

[3] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean.
Distributed representations of words and phrases and their compositionality.
In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Reference II

[4] Or Sharir, Barak Peleg, and Yoav Shoham.
The cost of training nlp models: A concise overview.
arXiv preprint arXiv:2004.08900, 2020.

[5] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al.
Emergent abilities of large language models.
arXiv preprint arXiv:2206.07682, 2022.