MF20006: Introduction to Computer Science

Lecture 11: Artificial Intelligence III

Hui Xu

xuh@fudan.edu.cn

Outline

1. Embedding
2. Recurrent Neural Networks
3. Transformer

1. Embedding

Problem: How to Learn from Textual Data

❑How would this news affect the stock price?

Related information

Seeking Alpha • 23 hours ago

Nvidia Stock: 5 Reasons I Bought More Ahead of Earnings and So Should You (NASDAQ: NVDA)

Yahoo Finance • 1 year ago

Nvidia's China Trouble Could Slash Billions From Revenue, Analysts Warn

MSN • 1 year ago

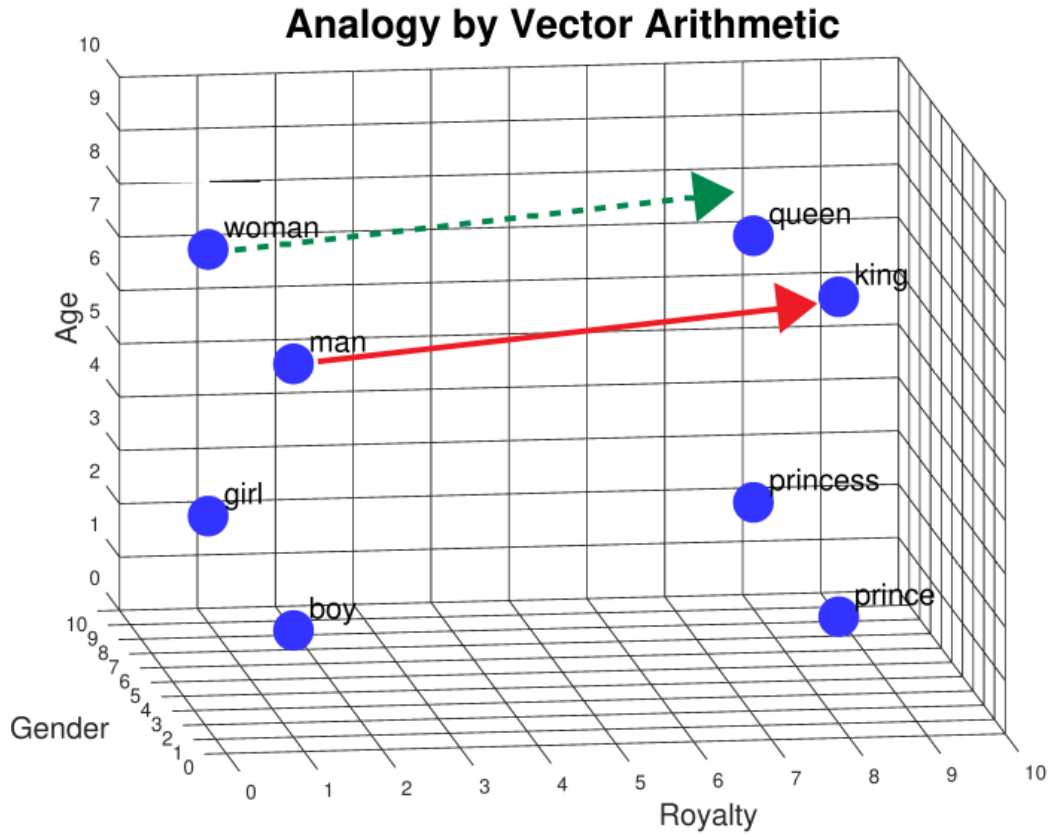
Cantor Fitzgerald Raises PT on NVIDIA Corporation (NVDA) Amid Growing Demand for AI Compute

How to Represent Each Word of a Sentence?

- **Use plain numbers: Each number represents a word, *e.g.*,**
  - "cat" → 1
  - "dog" → 2
- **One-Hot Encoding: Each bit represents a word, *e.g.*,**
  - "cat" → 10000000
  - "dog" → 01000000
- **Embedding: Each vector represents a word, *e.g.*,**
  - "cat" → [0.21, -0.56, 0.91, …]
  - "dog" → [0.20, -0.53, 0.88, …]

Intuition of Embeddings

- **Similar words have similar embeddings.**
- **Relationships can be captured, *e.g.*,**
  - king – man + woman ≈ queen
  - Paris - France + Italy ≈ Rome

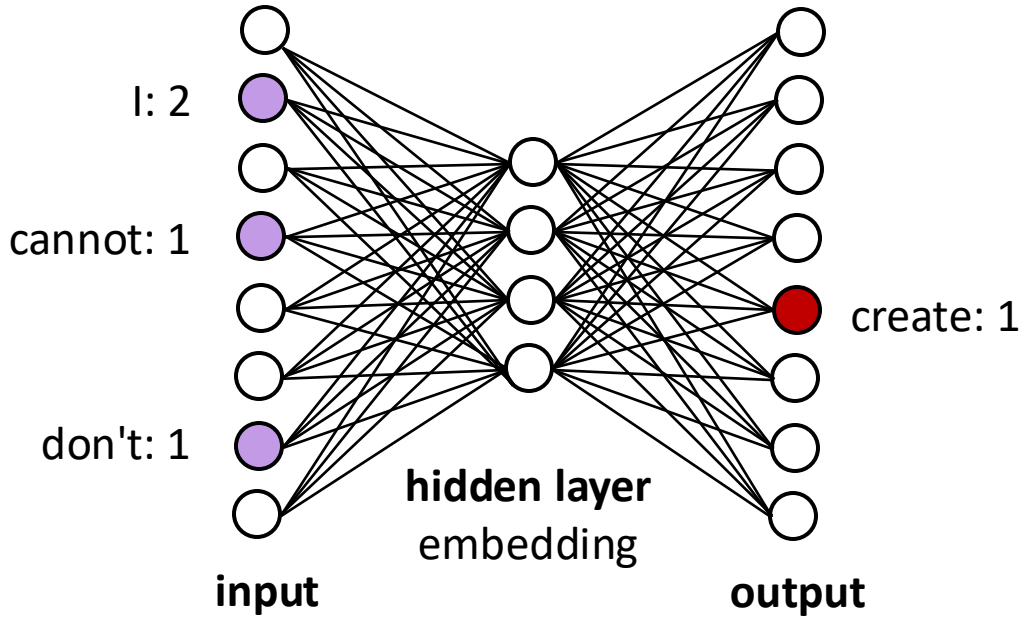


Embedding via Word2Vec

- **Design the semantic space is challenging.**
- **Use a model to learn the vector representation automatically.**
- **Two main methods:**
  - Continuous bag of words (CBOW)
  - Skip-gram

Word2Vec: Continuous Bag of Words

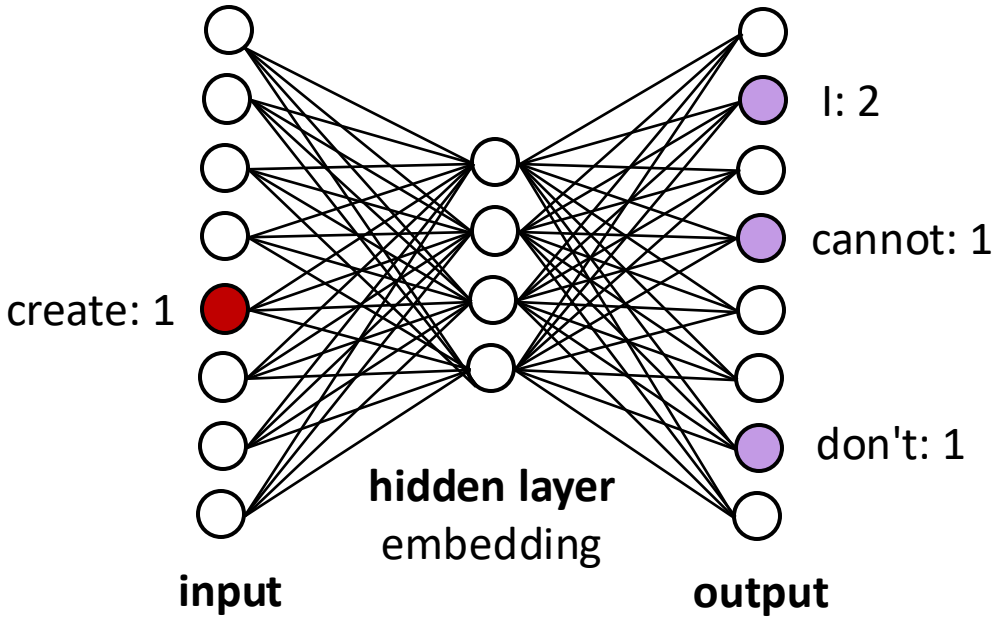
- Predict a target word given its surrounding context words.
- Use one-hot encoding for the input and output.
- Each input neuron corresponds to a word
  - The value is the frequency.



What I cannot create, I don't understand

Word2Vec: Skip-gram

- Predict the context words surrounding a given target word.



What I cannot create, I don't understand

Chinese Word2Vec

- Word2Vec works on words, not characters.
- But Chinese has no spaces between words.
- An extra text segmentation step is required.

我/ 喜欢/ 自然语言/ 处理

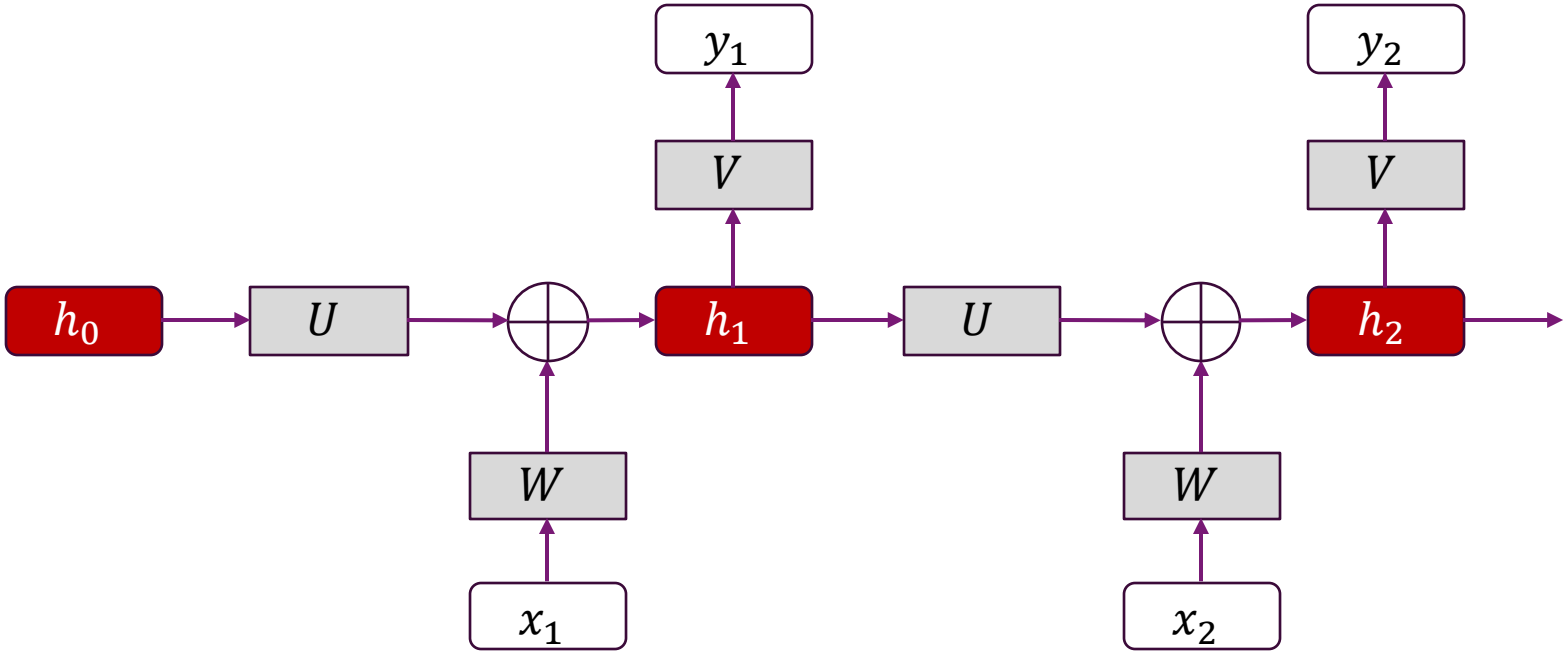
2. Recurrent Neural Networks

Recurrent Neural Networks (RNN)

- **Previous models take fixed-size input and produce fixed-size output.**
- **But many problems involve sequences of varying length:**
  - Finance: stock prices across days.
  - Text: a sentence of words.
  - Speech/music: waveform over time.
- **We need a model that can remember past inputs while processing the current one.**

Recurrent Neural Networks

- RNN introduces the concept of a hidden state or memory h_i.
- x_i is the word embedding of input; y_i is the output.
- W, U, V are the weight matrices to be trained.



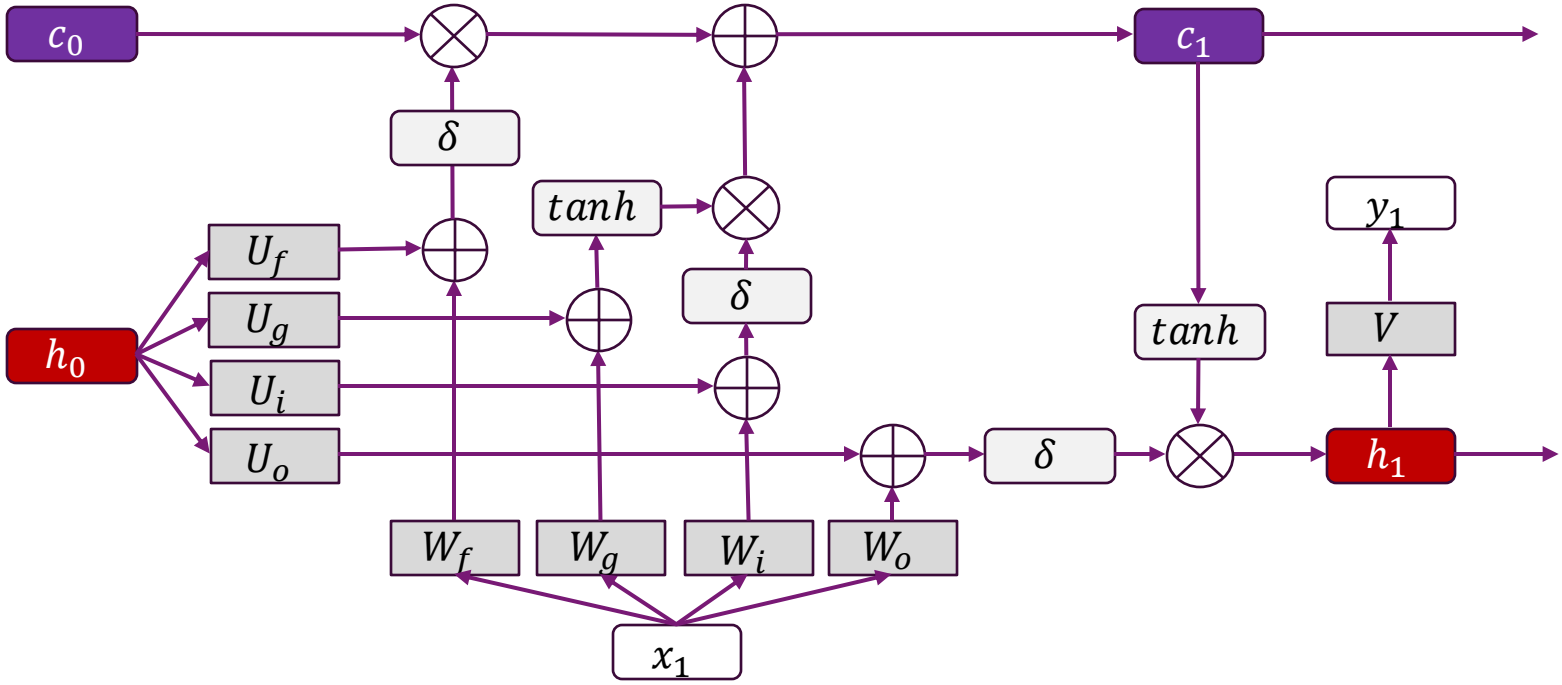
$$h_t = g(Uh_{t-1} + Wx_t) \qquad y_t = f(Vh_t)$$

Problem of RNN

- **Vanilla RNNs mainly learn short-term patterns.**
  - Because of vanishing gradients.
  - The problem motivated LSTM
- **Slow: RNNs process one step at a time.**
  - Because the hidden state depends on previous step.
  - The problem motivated transformer (LLM).

Long Short-Term Memory Network (LSTM)

- **Hidden state**: For short-term memory
- Cell state: For long-term memory



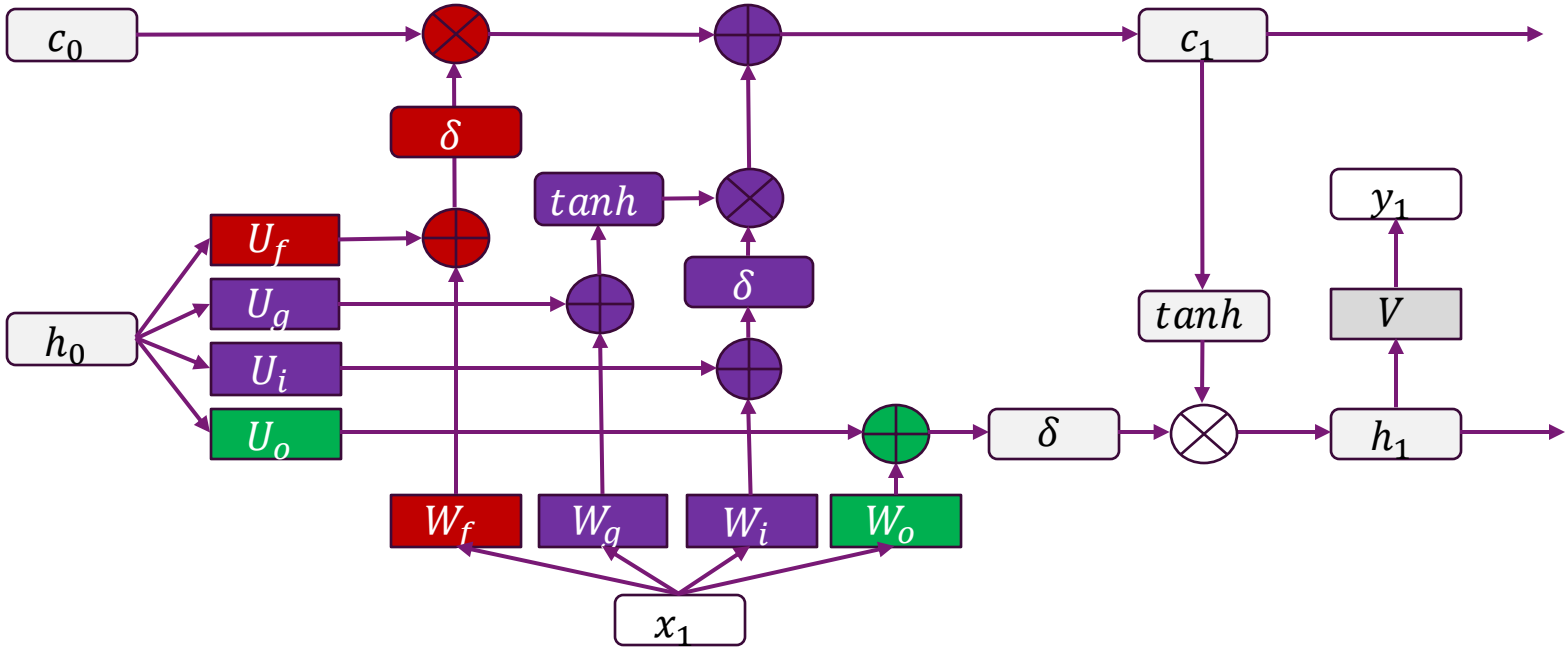
δ : sigmoid

$\otimes$: elementwise multiplication

$\oplus$: elementwise addition

LSTM

- **Forget Gate:** Control what old information to erase.
- **Input gate:** Control what new information to add.
- **Output gate:** Control what to send to hidden state.



δ : sigmoid

$\otimes$: elementwise multiplication

$\oplus$: elementwise addition

Problems that Can Be Modeled with RNN/LSTM

❑Natural Language Processing

- ➢Text Classification: sentiment analysis, topic classification
- ➢Chatbots
- ➢Translation

❑Time Series & Forecasting

- ➢Stock Price Prediction (financial time series forecasting)
- ➢Weather Forecasting (temperature, rainfall, wind patterns)

Autoregressive

- **Scenarios: In chatbots or translation tasks.**
  - Should the model consider already generated tokens?
- **The model predicts the next token based on the previous tokens.**
- **Each output is fed back as input for the next prediction.**

LSTM Tutorial: Data Preparation

❑Data in CSV format.

```
Date,Open,High,Low,Close,Volume
2015-01-02,24.26,24.73,23.82,24.72,212818400
2015-01-05,23.58,24.11,23.39,24.03,257142000
2015-01-06,23.58,23.84,23.22,23.64,263188400
2015-01-07,23.91,24.01,23.68,23.79,160423600
2015-01-08,24.83,24.89,24.12,24.24,237458000
2015-01-09,24.86,25.13,24.46,25.0,214798000
...
```

LSTM Tutorial: Data Preprocessing

```
df = pd.read_csv("AAPL.csv", index_col="Date", parse_dates=True)

prices = df[['Close']].values

# Normalize to [0,1]
scaler = MinMaxScaler()
prices_scaled = scaler.fit_transform(prices)

# Crate sequence: predict the 21st item based on the past 20 items
def create_sequences(data, seq_len=20):
    xs, ys = [], []
    for i in range(len(data) - seq_len):
        xs.append(data[i:i+seq_len])
        ys.append(data[i+seq_len])
    return np.array(xs), np.array(ys)

X, y = create_sequences(prices_scaled, 20)
```

LSTM Tutorial: Prepare Training Data and Testing Data

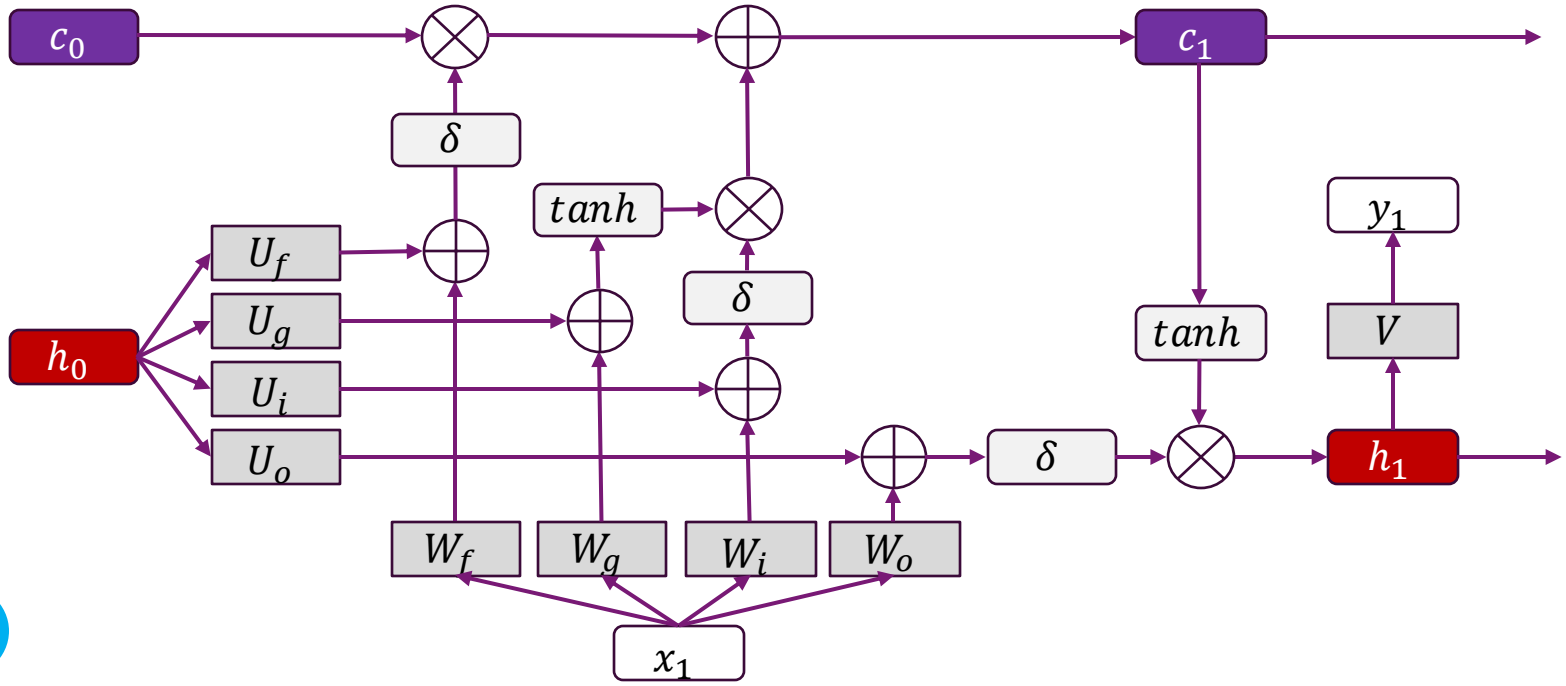
```
# Convert to torch Tensor
X = torch.tensor(X, dtype=torch.float32)
y = torch.tensor(y, dtype=torch.float32)

# Split the dataset into two parts:
# 80% for training and 20% for testing
train_size = int(len(X) * 0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

print("Training data:", X_train.shape, y_train.shape)
print("Testing data:", X_test.shape, y_test.shape)
```

LSTM Tutorial: Define LSTM Model

```
class LSTMModel(nn.Module):
    def __init__(self,
                 input_size=1, # dimension of input
                 hidden_size=64, # feature size
                 num_layers=1,
                 output_size=1):   # dimension of output
        super(LSTMModel, self).__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers, True)
        self.fc = nn.Linear(hidden_size, output_size)
```



LSTM Tutorial: Define LSTM Model

```
class LSTMModel(nn.Module):
    ...
    def forward(self, x):
        # shape of x: [batch, seq_len, hidden_size]
        # shape of out: [batch, seq_len, hidden_size]
        out, _ = self.lstm(x)
        out = out[:, -1, :] # The last item: [batch, hidden_size]
        out = self.fc(out) # fully connected
        return out
```

LSTM Tutorial: Training

```
model = LSTMModel()
optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

# DataLoader
train_loader = DataLoader(list(zip(X_train, y_train)), batch_size=64,
shuffle=True)

EPOCHS = 20
for epoch in range(EPOCHS):
    model.train()
    for xb, yb in train_loader:
        optimizer.zero_grad()
        preds = model(xb)
        loss = nn.MSELoss()(preds, yb)
        loss.backward()
        optimizer.step()
    print(f"Epoch {epoch+1}/{EPOCHS}, Loss: {loss.item():.6f}")
```

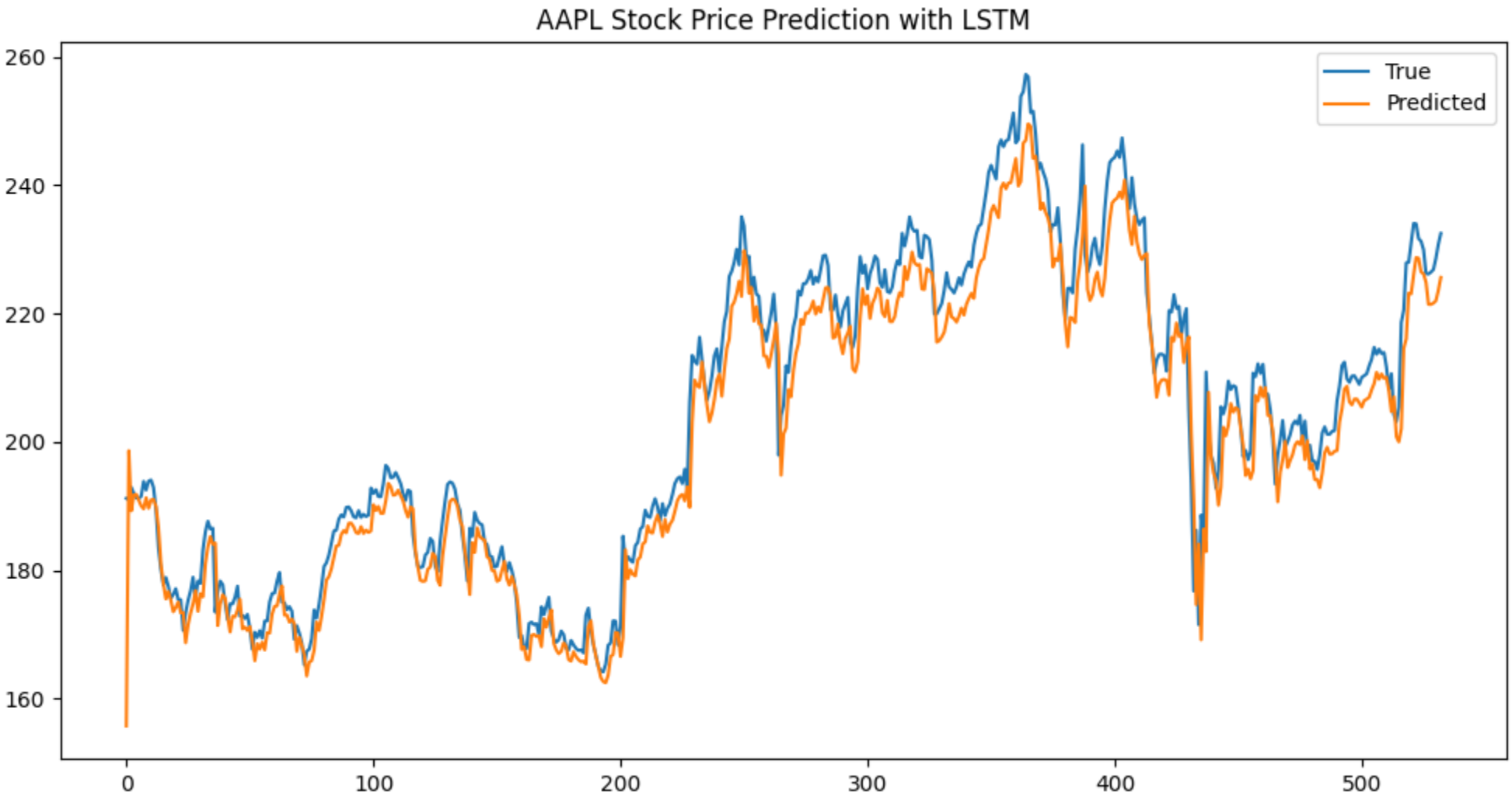
LSTM Tutorial: Evaluation

```
# Switch the model to evaluation mode
model.eval()
with torch.no_grad():
    preds = model(X_test)

# Convert the data back to the original price scale
preds_rescaled = scaler.inverse_transform(preds.numpy())
y_test_rescaled = scaler.inverse_transform(y_test.numpy())

# Plot the true prices vs. predicted prices
plt.figure(figsize=(12,6))
plt.plot(y_test_rescaled, label="True")
plt.plot(preds_rescaled, label="Predicted")
plt.legend()
plt.title("AAPL Stock Price Prediction with LSTM")
plt.show()
```

Example Result



The predictions are good in trend, but they respond one day late.

3. Transformer

Why Transformer?

❑Before Transformers:

- ➢RNNs/LSTMs: good at sequences, but slow (process one step at a time), hard with long-term dependencies.
- ➢CNNs: can parallelize, but not great at long sequential context.

❑We need a model that:

- ➢Handles long sequences well.
- ➢Works in parallel (fast training).
- ➢Captures relationships between tokens, no matter how far apart.

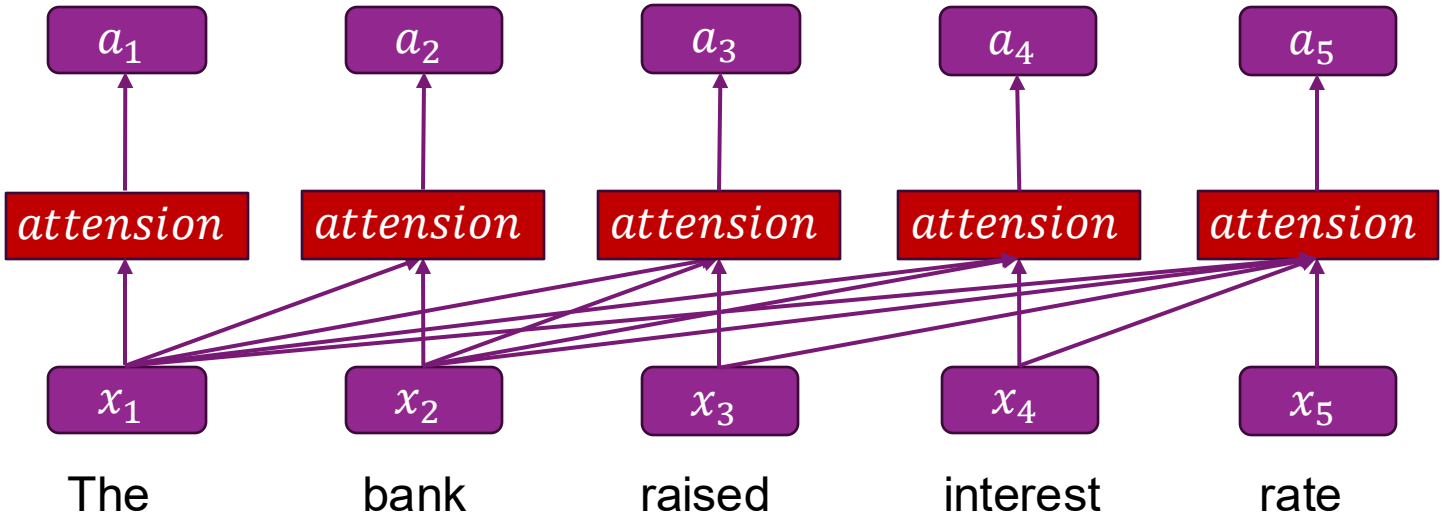
❑That's the Transformer (Vaswani et al., 2017, "Attention Is All You Need").

Core Idea: Attention

- **At the heart of a Transformer is self-attention.**
- **When reading a sentence, a word's meaning depends on other words in context.**
  - "The bank raised interest rates". Here, "bank" refers to a financial institution.
  - But in "He sat on the bank of the river", "bank" means riverside.
- **We need a mechanism where each word looks at other words and decides how much attention to give them.**

Simplified Version

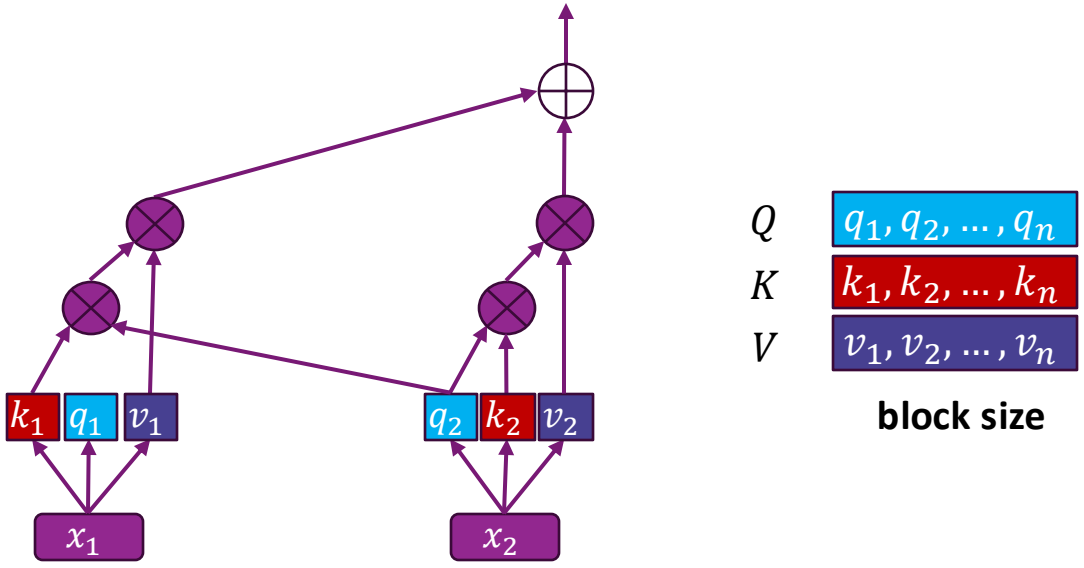
- **The context of previous tokens are captured by the attention matrix.**
  - $\{a_1, \ldots, a_n\}$ are similar to the hidden states of LSTM.
- **The calculation of $\{a_1, \ldots, a_n\}$ are parallel.**



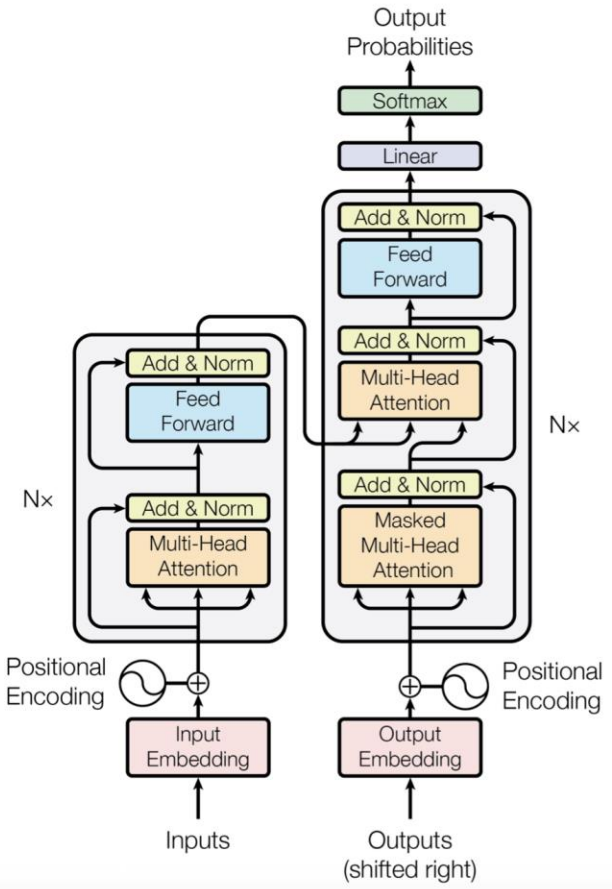
$$a_t = \sum_{i \le t} w_{it} \times x_i$$

Detailed Structure of Attention: Q, K, V

- **Think of each token embedding going into three linear projections:**
  - **Q (Query)** → "What am I looking for?"
  - **K (Key)** → "What do I contain?"
  - **V (Value)** → "What information do I provide?"
- **Learn different k, v, q for different positions.**



Transformer



Encoder: The encoder is composed of a stack of $N = 6$ identical layers. Each layer has two sub-layers. The first is a multi-head self-attention mechanism, and the second is a simple, position-wise fully connected feed-forward network. We employ a residual connection [11] around each of the two sub-layers, followed by layer normalization [1]. That is, the output of each sub-layer is $\mathrm{LayerNorm}(x + \mathrm{Sublayer}(x))$, where $\mathrm{Sublayer}(x)$ is the function implemented by the sub-layer itself. To facilitate these residual connections, all sub-layers in the model, as well as the embedding layers, produce outputs of dimension $d_{\text{model}} = 512$.

Decoder: The decoder is also composed of a stack of $N = 6$ identical layers. In addition to the two sub-layers in each encoder layer, the decoder inserts a third sub-layer, which performs multi-head attention over the output of the encoder stack. Similar to the encoder, we employ residual connections around each of the sub-layers, followed by layer normalization. We also modify the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, combined with fact that the output embeddings are offset by one position, ensures that the predictions for position i can depend only on the known outputs at positions less than i.

Ashish Vaswani, *et al.* "Attention Is All You Need." NIPS 2017.

Masking Mechanism

- The attention scores are computed as: $\frac{QK^T}{\sqrt{d_k}}$
- To mask future positions, we add a large negative number $-\infty$ to invalid positions.

$$Score_{i,j} = \begin{cases} \dfrac{\boldsymbol{QK^T}}{\sqrt{\boldsymbol{d_k}}}, & \text{if } j < i \\ -\infty, & otherwise \end{cases}$$

$$\text{mask matrix} = \begin{bmatrix} 0 & -\infty & -\infty \\ 0 & 0 & -\infty \\ 0 & 0 & 0 \end{bmatrix}$$

Token 1 can see it self.
Token 2 can see both token 1 and 2.
Token 3 can see token 1, 2, and 3.

Types of Models

- **Encoder Only: Compress and use the representation directly.**
  - Applications: Text classification, price prediction
- **Encoder-Decoder: Separates “understand input” and “generate output.”.**
- **Decoder only: Tokens cannot see future tokens.**
  - Applications: Translation, Chatbot

Artificial General Intelligence

Sparks of Artificial General Intelligence: Early experiments with GPT-4

Sébastien Bubeck Varun Chandrasekaran Ronen Eldan Johannes Gehrke
Eric Horvitz Ece Kamar Peter Lee Yin Tat Lee Yuanzhi Li Scott Lundberg
Harsha Nori Hamid Palangi Marco Tulio Ribeiro Yi Zhang

Microsoft Research

"Given the breadth and depth of GPT-4's capabilities, we believe that it could reasonably be viewed as an early (yet still incomplete) version of an artificial general intelligence (AGI) system."

weak /narrow AI ⟹ strong AI/AGI

application-specific

Sébastien Bubeck, *et al*. "Sparks of artificial general intelligence: Early experiments with gpt-4." *arXiv preprint arXiv:2303.12712 (2023)*.

What can GPT-4 Do? Reasoning?

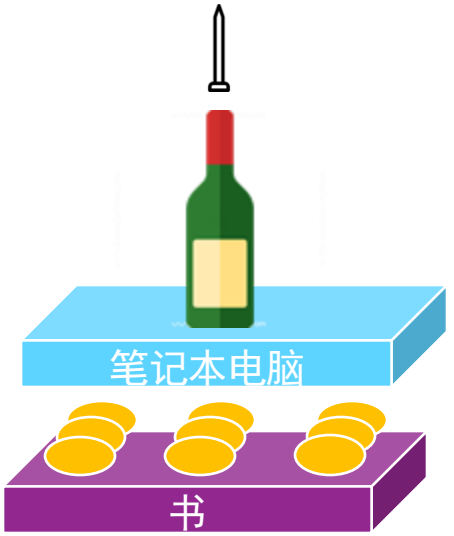
Prompt: Here we have a book, 9 eggs, a laptop, a bottle and a nail. Please tell me how to stack them onto each other in a stable manner.

1本书，9个鸡蛋，1台笔记本电脑，1个瓶子，1个钉子，如何叠在一起？

GPT-4:

One possible way to stack the objects onto each other in a stable manner is:

- Place the book flat on a level surface, such as a table or a floor. The book will serve as the base of the stack and provide a large and sturdy support.
- Arrange the 9 eggs in a 3 by 3 square on top of the book, leaving some space between them. The eggs will
form a second layer and distribute the weight evenly. Make sure the eggs are not
handle them gently to avoid dropping them.
- Place the laptop on top of the eggs, with the screen facing down and the keyb
laptop will fit snugly within the boundaries of the book and the eggs, and its
will provide a stable platform for the next layer. Make sure the laptop is clos
not press any keys or buttons.
- Place the bottle on top of the laptop, with the cap facing up and the bottom
will add some height and balance to the stack, and its cylindrical shape will r
with the laptop. Make sure the bottle is empty and closed, and do not shake or
- Place the nail on top of the bottle cap, with the pointy end facing up and th
The nail will be the final and smallest object in the stack, and its sharp tip
rolling or sliding off the bottle. Make sure the nail is clean and straight, an
it.



Sébastien Bubeck, *et al*. "Sparks of artificial general intelligence: Early experiments with gpt-4." *arXiv preprint arXiv:2303.12712 (2023)*.

Capability of GPT-4 in Taking Exams

Exam	GPT-4	GPT-4 (no vision)	GPT-3.5
Uniform Bar Exam (MBE+MEE+MPT)	298 / 400 (~90th)	298 / 400 (~90th)	213 / 400 (~10th)
LSAT	163 (~88th)	161 (~83rd)	149 (~40th)
SAT Evidence-Based Reading & Writing	710 / 800 (~93rd)	710 / 800 (~93rd)	670 / 800 (~87th)
SAT Math	700 / 800 (~89th)	690 / 800 (~89th)	590 / 800 (~70th)
Graduate Record Examination (GRE) Quantitative	163 / 170 (~80th)	157 / 170 (~62nd)	147 / 170 (~25th)
Graduate Record Examination (GRE) Verbal	169 / 170 (~99th)	165 / 170 (~96th)	154 / 170 (~63rd)
Graduate Record Examination (GRE) Writing	4 / 6 (~54th)	4 / 6 (~54th)	4 / 6 (~54th)
USABO Semifinal Exam 2020	87 / 150 (99th - 100th)	87 / 150 (99th - 100th)	43 / 150 (31st - 33rd)
USNCO Local Section Exam 2022	36 / 60	38 / 60	24 / 60
Medical Knowledge Self-Assessment Program	75 %	75 %	53 %
Codeforces Rating	392 (below 5th)	392 (below 5th)	260 (below 5th)
AP Art History	5 (86th - 100th)	5 (86th - 100th)	5 (86th - 100th)
AP Biology	5 (85th - 100th)	5 (85th - 100th)	4 (62nd - 85th)
AP Calculus BC	4 (43rd - 59th)	4 (43rd - 59th)	1 (0th - 7th)
AP Chemistry	4 (71st - 88th)	4 (71st - 88th)	2 (22nd - 46th)
Introductory Sommelier (theory knowledge)	92 %	92 %	80 %
Certified Sommelier (theory knowledge)	86 %	86 %	58 %
Advanced Sommelier (theory knowledge)	77 %	77 %	46 %
Leetcode (easy)	31 / 41	31 / 41	12 / 41
Leetcode (medium)	21 / 80	21 / 80	8 / 80
Leetcode (hard)	3 / 45	3 / 45	0 / 45

GPT-4 Technical Report, 2023-03-16

GPT is not Trustworthy

百分位排名的含义

百分位排名（Percentile Rank）是统计学中用来衡量一个值在数据分布中的相对位置的指标。具体来说，它表示一个值在数据集中高于或等于该值的观测数据所占的百分比。

定义

百分位排名是从 0 到 100 的值，表示一个观测值相对于整个数据集的位置。如果某个值的百分位排名是 p，这意味着这个值高于或等于数据集中 $p\%$ 的其他值。

公式

百分位排名可以通过以下公式计算：

$$P = \frac{\text{小于等于目标值的数量}}{\text{总样本数量}} \times 100$$

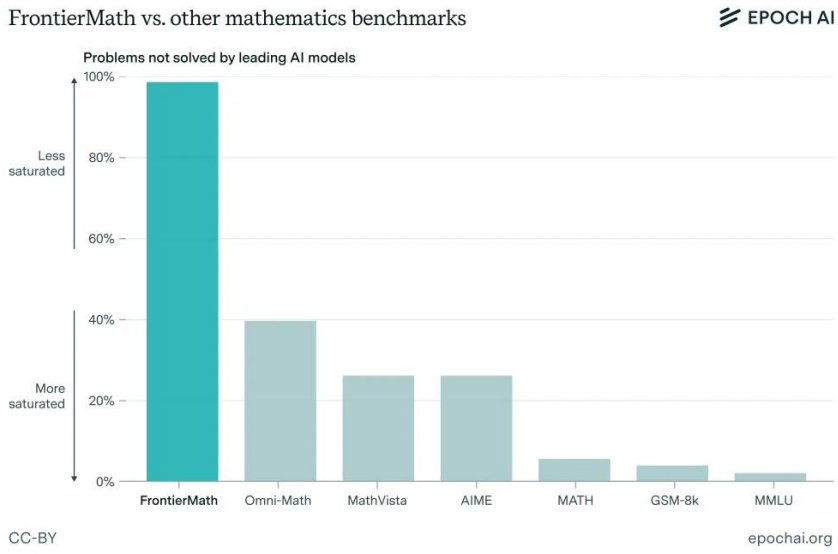
示例

假设某次考试有 1000 名考生，某考生的成绩排名第 850 名（从高到低）。

- **百分位排名**：

$$P = \frac{850}{1000} \times 100 = 85\%$$

这表明该考生的成绩高于或等于 85% 的考生。



Debate: Is AGI a Threat?

"The alarm bell I'm ringing has to do with the existential threat of them taking control. I used to think it was a long way off, but I now think it's serious and fairly close."

Geoffrey Hinton @ EmTech Digital 2023

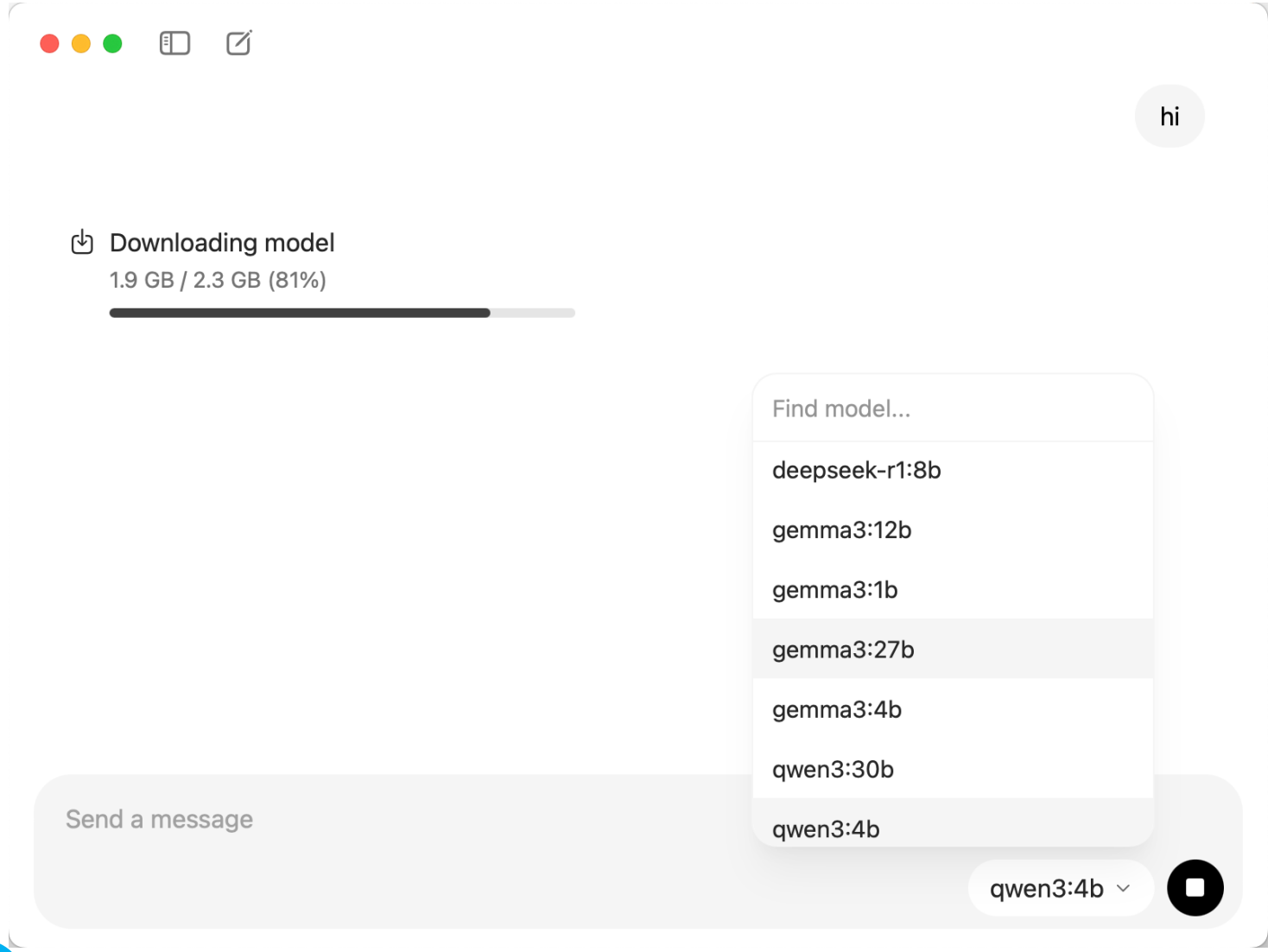
Paperclip Maximiser Problem by Nick Bostrom

- Suppose we have an AI whose only goal is to make as many paper clips as possible.
- The AI will realize quickly that it would be much better if there were no humans because humans might decide to switch it off.
- Also, human bodies contain a lot of atoms that could be made into paper clips.
- ...

Use LLMs

- **Use online services**
- **Install local models**
  - Ollama: https://ollama.com
  - LM Studio: https://lmstudio.ai
- **Develop agents**

Install Local Models via ollama



Agent Development: Find Your API Key

- LLMs generally provide services via API Keys
- QWen: https://bailian.console.aliyun.com/

Agent Development: One-Round Chat

```
import os
from dashscope import Generation

API_KEY = "sk-XXXXXXXXXXXXXXXXXXXX"

resp = Generation.call(
    model = "qwen-plus",
    prompt = "Hello! ",
    api_key = API_KEY,
)


print(resp["output"]["text"])
```

Agent Development: Multi-Round Chat

```
messages = []

def chat_with_qwen(user_input: str):
    messages.append({"role": "user", "content": user_input})
    resp = Generation.call(
        model="qwen-plus",
        messages=messages,
        result_format="message",
        api_key=API_KEY,
    )

    choices = resp.output.choices
    if choices and len(choices) > 0:
        reply = choices[0].message.content
        messages.append({"role": "assistant", "content": reply})
        return reply
    else:
        return "[No Response]"
```

"user" means the input is from user input

"assistant" indicates the text is from llm output

Agent Development: Multi-Round Chat

```
if __name__ == "__main__":
    print("=== QWen ===")
    while True:
        user_input = input("You: ")
        if user_input.strip().lower() in {"exit", "quit"}:
            break
        reply = chat_with_qwen(user_input)
        print("QWen:", reply)
```