

20240321 Meeting

312707003 黃鈺婷

大型語言模型(LLM)

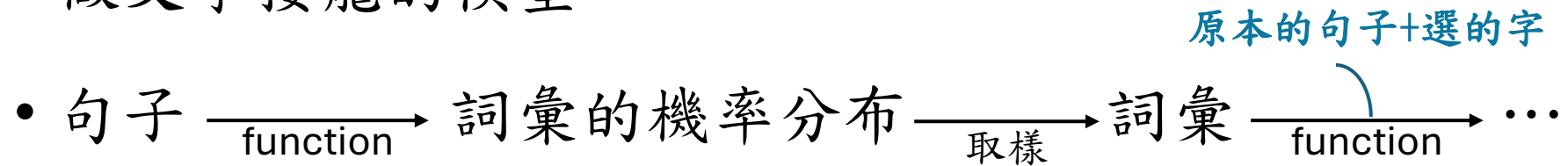
- 文字接龍 - GPT
- 文字填空 - Bert

生成式學習 (Generative learning)

- 又稱結構化學習 (Structured learning)
- ChatGPT
Generative Pre-trained Transformer
- 生成有結構的物件
Ex. 文句 (由token組成)、影像 (由像素組成)
- Autoregressive (AR) model：常用於文字
Ex. ChatGPT 文字生成是一個字接一個字
- Non-autoregressive model：常用於影像

ChatGPT

- 做文字接龍的模型



一直到取出代表end的符號

測試Testing

- 產生答案有隨機性，所以每次產生的答案都不一樣

訓練Training

- Function怎麼被找出來的？大量網路資料+訓練者提供的資料
- 如何精準提出需求？Prompting

Prompting

- Instruction Tuning

把任務改成指令，訓練時給機器不同指令，測試時期待它能知道做甚麼

- Chain of Thought (CoT) Prompting

在給機器範例時，順便給推論的過程，期待在回答時能先寫推論過程再寫答案

背後的技術

- 一般機器如何學習？

監督式學習 + 自監督式學習

人類提供的成對資料(有限)

網路上大量資料

- 網路上每段文字都能教機器做文字接龍

預訓練(Pre-trained)

- 監督式學習之前，透過網路資料學習的過程，又稱自監督式學習



- 預訓練：預先訓練的模型或預先訓練模型的過程
- 微調：使用別人訓練好的模型，換成自己的數據，調整參數再訓練一遍
- 改造(Adapter)：不動參數，插入額外模組
- 遷移學習：預訓練模型獲得的知識可遷移到目標任務上

找函數的三步驟

- 設定範圍：訂出候選函數的集合

Ex. RNN, Transformer

- 設定標準：訂出評量函數好壞的標準

Loss function，計算標準答案與f輸出結果的差異

- 達成目標：找出最好的函數→最佳化(optimization)

找出Loss最低的函數

資料的前處理

- Content filtering
- Text extraction
- Quality filtering
- Repetition removal
- Document deduplication
- Test-set filtering

去除重複資料

NLP任務

- 對整個句子進行分類
- 對句子中的每個詞進行分類
- 生成文本內容
- 從文本中提取答案
- 從輸入文本生成新句子

pipeline三個主要步驟

1. 文本被預處理為模型可以理解的格式
2. 預處理的輸入被傳遞給模型
3. 模型處理後輸出最終人類可以理解的結果

零樣本分類(zero-shot-classification)

```
from transformers import pipeline

classifier = pipeline("zero-shot-classification")
classifier(
    "This is a course about the Transformers library",
    candidate_labels=["education", "politics", "business"],
)
```

```
{'sequence': 'This is a course about the Transformers library',
 'labels': ['education', 'business', 'politics'],
 'scores': [0.8445963859558105, 0.111976258456707, 0.043427448719739914]}
```

文本生成(text-generation)

```
from transformers import pipeline

generator = pipeline("text-generation", model="distilgpt2")
generator(
    "In this course, we will teach you how to",
    max_length=30,
    num_return_sequences=2,
)
```

填充空缺(fill-mask)

```
from transformers import pipeline
```

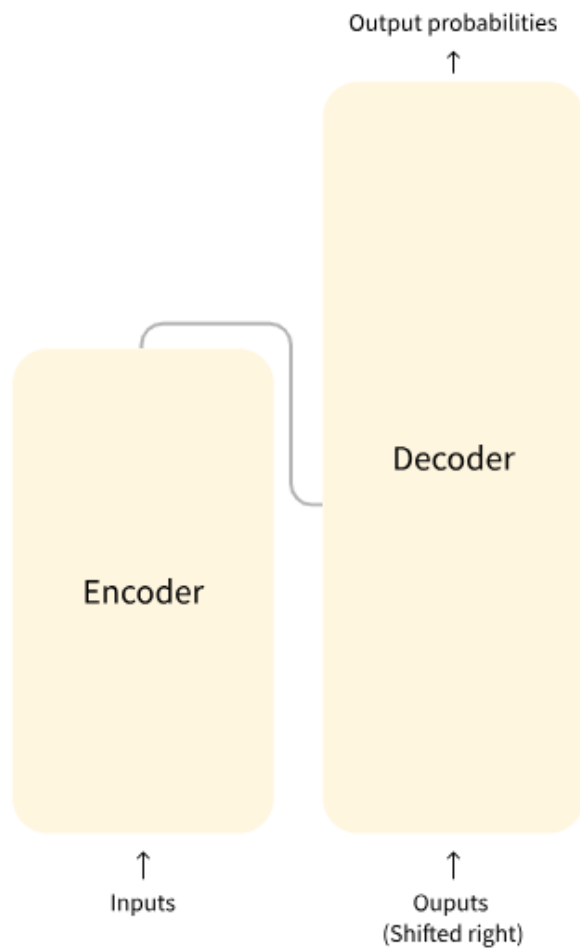


```
unmasker = pipeline("fill-mask")
```

```
unmasker("This course will teach you all about <mask> models.", top_k=2)
```

```
[{'sequence': 'This course will teach you all about mathematical models.',  
  'score': 0.19619831442832947,  
  'token': 30412,  
  'token_str': ' mathematical'},  
 {'sequence': 'This course will teach you all about computational models.',  
  'score': 0.04052725434303284,  
  'token': 38163,  
  'token_str': ' computational'}]
```

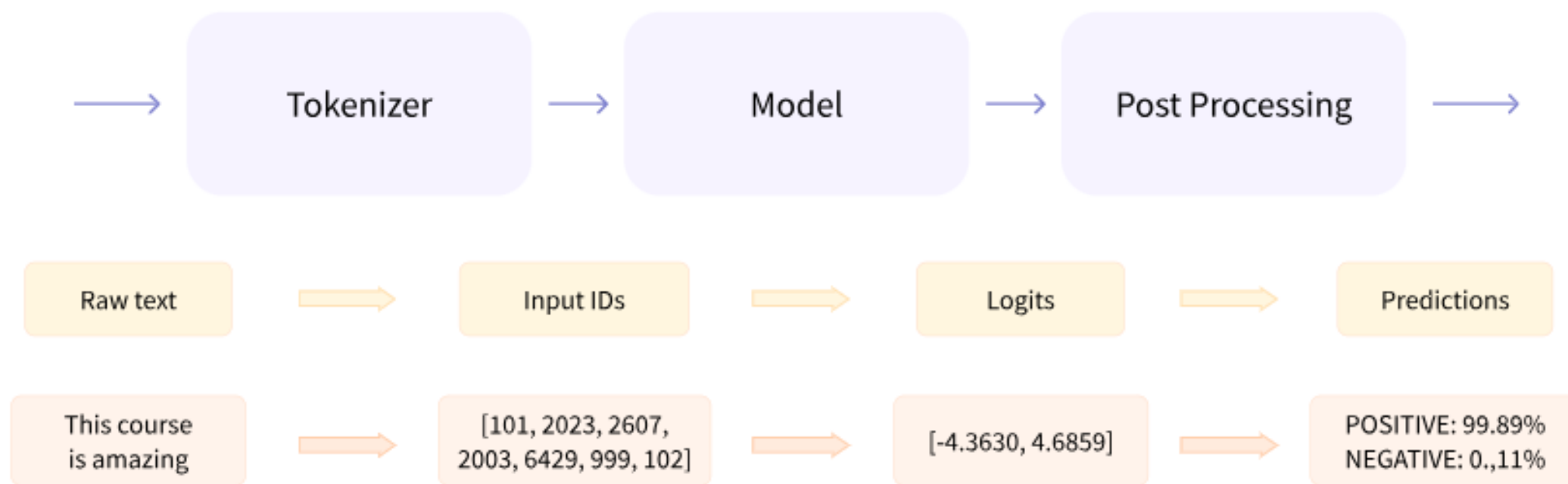
模型的架構



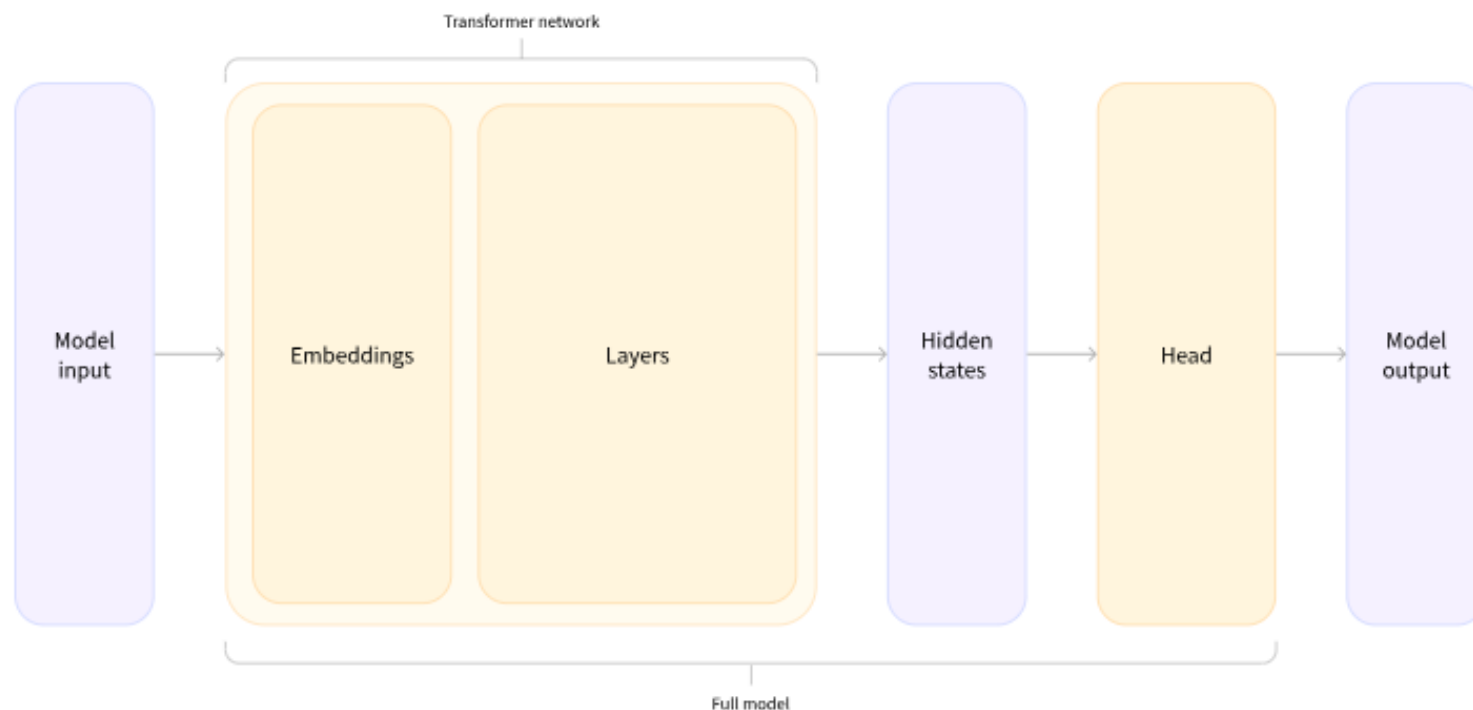
注意力層

- 告訴模型在處理每個單詞的表示時，要特別重視傳遞給它的句子中的某些單詞，並且忽略部分單詞
- 解碼器會被輸入整個目標，但不允許獲取到要翻譯的單詞
Ex. 當試圖預測第4個單詞時，注意力層只能獲取位置1到3的單詞
- 自迴歸模型
在每個階段，對於給定的單詞，注意力層只能獲取到句子中位於將要預測單詞前面的單詞

Transformers 管道內部



Transformers 管道內部



- Transformers 模型的輸出直接發送到模型頭進行處理
- 嵌入層將標記化輸入中的每個輸入ID轉換為表示token的向量
- 後續層使用注意機制操縱這些向量，以生成句子的最終表示

對輸出進行後處理

```
print(outputs.logits)
```



```
tensor([[ -1.5607,  1.6123],  
        [ 4.1692, -3.3464]], grad_fn=<AddmmBackward>)
```

- 我們的模型預測第一句為[-1.5607, 1.6123]，第二句為[4.1692, -3.3464]，這些不是概率，而是logits
- 要轉換為概率，它們需要經過SoftMax層

截斷序列

```
sequences = ["I've been waiting for a HuggingFace course my whole life👉",  
  
# Will truncate the sequences that are longer than the model max length  
# (512 for BERT or DistilBERT)  
model_inputs = tokenizer(sequences, truncation=True)  
  
# Will truncate the sequences that are longer than the specified max length  
model_inputs = tokenizer(sequences, max_length=8, truncation=True)
```