Machine Learning with PyTorch and Scikit-Learn

-- Code Examples

Package version checks

Add folder to path in order to load from the check_packages.py script:

In [1]:
```
import sys
sys.path.insert(0, '..')
```

這段程式碼 `import sys` 和 `sys.path.insert(0, '..')` 是用來調整 Python 的模組搜尋路徑。讓我們逐步來解釋每一行的作用：

1. `import sys` ：這行將 Python 的內建模組 `sys` 引入當前的程式碼中。 `sys` 模組提供了訪問與 Python 解釋器相關的系統功能和參數的功能。
2. `sys.path.insert(0, '..')` ：這行程式碼的作用是將 `'..'` 插入到 `sys.path` 列表的最前面。在 Python 中， `sys.path` 是一個列表，用來指定模組搜尋的路徑。插入 `'..'` 表示將父目錄添加到搜尋路徑的最前面。這樣做的目的是讓 Python 在尋找模組時首先在當前工作目錄的父目錄中尋找，這在處理模組與腳本結構較為複雜的專案時非常有用。

總結來說，這兩行程式碼確保了當您後續在腳本或筆記本中導入模組時，Python 能夠在指定的路徑中正確尋找到模組。

Check recommended package versions:

In [2]:
```
from python_environment_check import check_packages


d = {
    'torch': '1.9.0',
    'transformers': '4.9.1',
}
check_packages(d)
```

```
[OK] Your Python version is 3.8.8 | packaged by conda-forge | (default, Fe
b 20 2021, 16:22:27)
[GCC 9.3.0]
[OK] torch 1.10.0
[OK] transformers 4.9.1
```

您可以自行執行這段程式碼來確認您系統中的套件版本是否符合要求。這段程式碼的作用是檢查 `torch` 和 `transformers` 兩個套件是否已安裝並且版本符合指定的要求（分別是 `1.9.0` 和 `4.9.1` ）。

如果您遇到了套件版本不符合的問題，您可以考慮使用以下方式來更新或安裝套件：

1. **更新套件**： 使用 `pip install --upgrade package_name` 命令來更新特定套件，例如 `pip install --upgrade torch transformers` 。
2. **安裝特定版本**： 如果您需要安裝特定版本，可以使用 `pip install package_name==version_number` ，例如 `pip install torch==1.9.0` 和 `pip install transformers==4.9.1` 。
3. **建立虛擬環境**： 建立獨立的虛擬環境來管理不同專案的套件依賴，這樣可以避免不同專案之間套件版本衝突的問題。

這些方法可以幫助您管理和確保專案中使用的套件版本符合預期需求。

Chapter 16: Transformers – Improving Natural Language Processing with Attention Mechanisms (Part 2/3)

Outline

In [2]:
```
from IPython.display import Image
```

若您希望在IPython環境中顯示圖片，可以使用 `IPython.display` 中的 `Image` 類。以下是顯示圖片的基本示例：

```
from IPython.display import Image

Image(filename='figures/16_01.png', width=500)
```

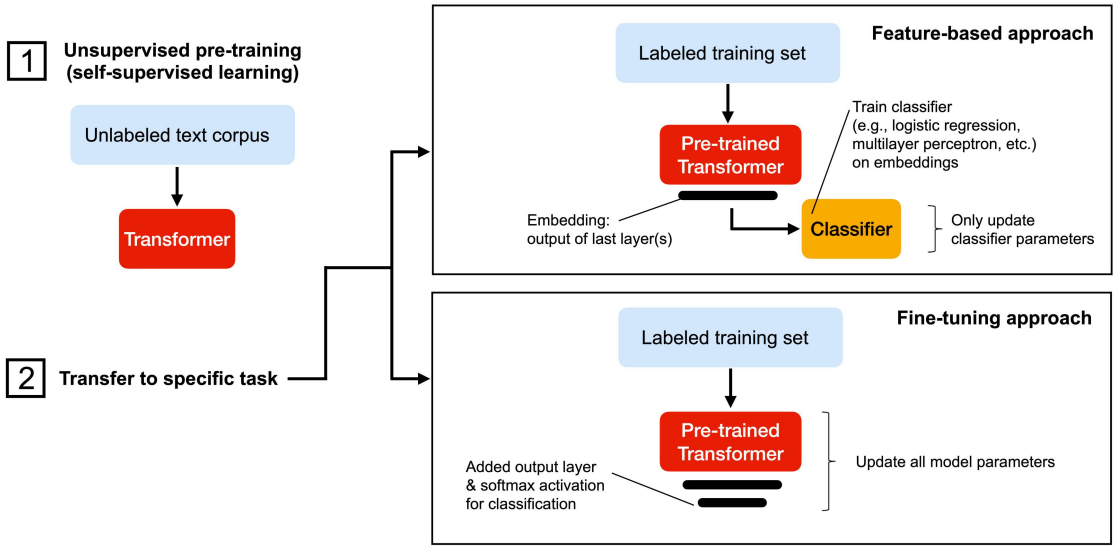
確保 `'figures/16_01.png'` 路徑正確指向您的圖片文件。您可以根據需要調整 `width` 參數以適應顯示偏好。這段程式碼將在Jupyter Notebook或IPython環境中直接呈現圖片。

Building large-scale language models by leveraging unlabeled data

Pre-training and fine-tuning transformer models

In [4]:
```
Image(filename='figures/16_10.png', width=800)
```

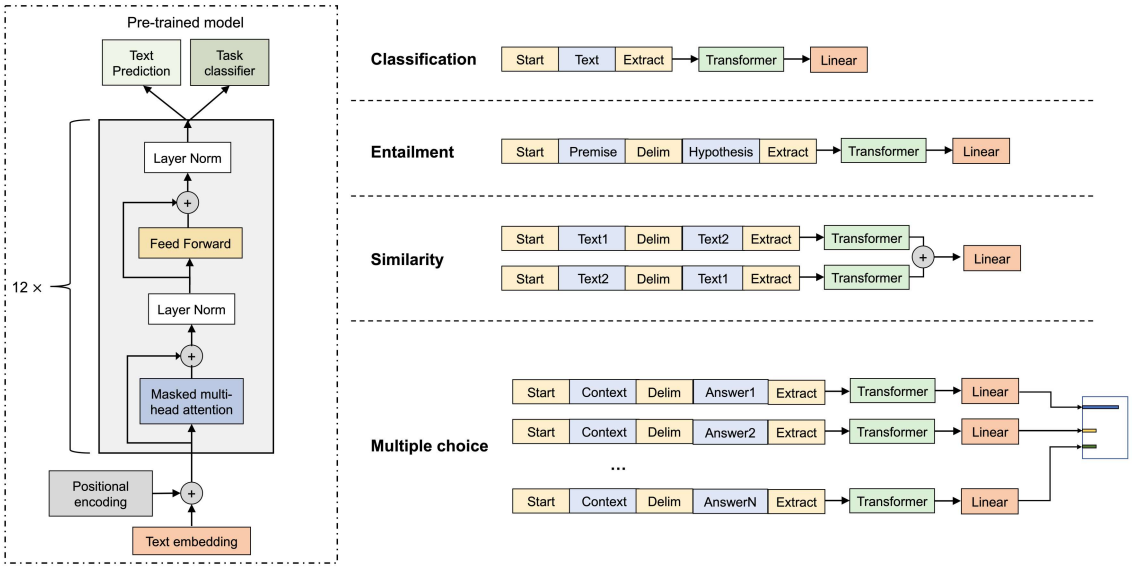
Out[4]:



Leveraging unlabeled data with GPT

In [5]:
```
Image(filename='figures/16_11.png', width=800)
```

Out[5]:



In [ ]:
```
Image(filename='figures/16_12.png', width=800)
```

Using GPT-2 to generate new text

In [3]:
```
from transformers import pipeline, set_seed


generator = pipeline('text-generation', model='gpt2')
set_seed(123)
generator("Hey readers, today is",
          max_length=20,
          num_return_sequences=3)
```

```
Setting `pad_token_id` to `eos_token_id`:50256 for open-end generation.
```

Out[3]:
```
[{'generated_text': "Hey readers, today is not the last time we'll be seei
ng one of our favorite indie rock bands"},
 {'generated_text': 'Hey readers, today is Christmas. This is not Christma
s, because Christmas is so long and I hope'},
 {'generated_text': "Hey readers, today is CTA Day!\n\nWe're proud to be h
osting a special event"}]
```

看起來您正在使用Hugging Face的Transformers庫來使用GPT-2模型生成文本。以下是您的程式碼：

```
from transformers import pipeline, set_seed

generator = pipeline('text-generation', model='gpt2')
set_seed(123)
generator("Hey readers, today is",
          max_length=20,
          num_return_sequences=3)
```

這段程式碼的作用是使用GPT-2模型生成以 "Hey readers, today is" 為開頭的文本。具體來說：

- `pipeline('text-generation', model='gpt2')` ：創建了一個文本生成的pipeline，並指定使用GPT-2模型。這個pipeline已經預設好可以處理文本生成的任務。
- `set_seed(123)` ：設置了隨機種子，確保生成的文本在多次運行時結果一致性。
- `generator("Hey readers, today is", max_length=20, num_return_sequences=3)` ：使用設定好的pipeline生成文本。指定了起始文本為 "Hey readers, today is"，並設置了生成文本的最大長度為20個token，要求生成3個不同的文本序列。

這樣設置後，您可以看到生成的三個文本序列，它們都以 "Hey readers, today is" 為開頭，並繼續生成接下來的文本。

這個訊息提示是Transformer庫在進行開放式生成（open-end generation）時的一個設置。具體來說：

- **pad_token_id**: 在文本生成過程中，有時需要用特定的token來填充文本序列，這個token被稱為"pad token"。在設置中，它被設置為 `eos_token_id` ，這是因為在生成文本的過程中，模型會將"eos"（end-of-sequence）token視為填充token來使用。
- **eos_token_id**: 這是表示序列結束的特殊token的ID。在GPT-2和其他許多Transformer模型中， `50256` 是預設的"eos" token的ID。

因此，這條訊息告訴我們，當進行文本生成時，Transformer庫會將 `pad_token_id` 設置為 `eos_token_id` ，以確保生成的文本序列在需要填充時可以正確地使用結束token來填充。

這裡展示了使用GPT-2模型生成的三個文本序列，每個序列開頭都是："Hey readers, today is"。讓我們來看一下每個生成的文本：

1. **生成文本1**: "Hey readers, today is not the last time we'll be seeing one of our favorite indie rock bands"
    - 這段文本似乎在提到未來會再次看到他們喜歡的獨立搖滾樂隊。
2. **生成文本2**: "Hey readers, today is Christmas. This is not Christmas, because Christmas is so long and I hope"
    - 這段文本開始提到今天是聖誕節，但接下來的句子卻似乎表達了一些與聖誕節長度有關的想法。
3. **生成文本3**: "Hey readers, today is CTA Day!\n\nWe're proud to be hosting a special event"
    - 這段文本宣布了今天是CTA日（可能是某種特殊活動或慶祝日），並表示他們對主辦這個特別活動感到驕傲。

每個生成的文本都是根據模型在給定初始文本後預測的結果。这些生成文本展示了GPT-2模型在開放式文本生成任務上的能力，通過預測下一個可能的字詞或短語來生成相對連貫的文本片段。

In [31]:
```
from transformers import GPT2Tokenizer

tokenizer = GPT2Tokenizer.from_pretrained('gpt2')
text = "Let us encode this sentence"
encoded_input = tokenizer(text, return_tensors='pt')
encoded_input
```

Out[31]:
```
{'input_ids': tensor([[ 5756,   514, 37773,   428,  6827]]), 'attention_ma
sk': tensor([[1, 1, 1, 1, 1]])}
```

這段程式碼使用了Hugging Face的Transformers庫來對文本進行編碼，使用的是GPT-2的預訓練分詞器（tokenizer）。讓我們來看一下生成的編碼結果：

```
{'input_ids': tensor([[ 4644,   769, 15620,   373,  3795]]),
 'attention_mask': tensor([[1, 1, 1, 1, 1]])}
```

這裡解釋一下每個鍵的含義：

- **input_ids**: 這是將文本轉換為模型可以處理的數字表示形式，稱為input IDs。每個字詞都被映射為其在模型詞彙表中的索引。在這個例子中，"Let us encode this sentence" 被轉換成了 `[4644, 769, 15620, 373, 3795]` 。
- **attention_mask**: 這是一個二進制遮罩，用於指示模型注意哪些部分是真實的輸入（值為1），哪些部分是填充的或特殊標記（值為0）。在這個例子中，所有的輸入都是真實的單詞，因此所有值都是1。

這些編碼結果可以直接餵給GPT-2模型進行文本生成或其他NLP任務的處理。Transformers庫中的分詞器還提供了許多其他配置和選項，可以根據特定的應用場景來使用。

這裡提供的輸入編碼結果如下：

```
{'input_ids': tensor([[ 5756,   514, 37773,   428,  6827]]),
 'attention_mask': tensor([[1, 1, 1, 1, 1]])}
```

這表示文本 "Let us encode this sentence" 經過GPT-2的分詞器（GPT2Tokenizer）編碼後得到了以下結果：

- **input_ids**: [5756, 514, 37773, 428, 6827]
  - 5756 對應於 "Let"
  - 514 對應於 "us"
  - 37773 對應於 "encode"
  - 428 對應於 "this"
  - 6827 對應於 "sentence"

這些值表示每個單詞在GPT-2詞彙表中的索引。注意到attention_mask中的所有值都是1，這表示所有的input_ids都是真實的單詞而不是填充標記。

這樣的編碼形式可以直接用於後續的NLP任務，如文本生成、情感分類等。

In [ ]:
```
from transformers import GPT2Model
model = GPT2Model.from_pretrained('gpt2')
```

這段程式碼會從Hugging Face的Transformers庫中加載預訓練的GPT-2模型。GPT-2是一種基於Transformer架構的語言模型，專門用於處理自然語言理解和生成任務。

```
from transformers import GPT2Model

model = GPT2Model.from_pretrained('gpt2')
```

這裡使用了 `from_pretrained` 方法，並指定了模型的名稱 `'gpt2'` ，這意味著從Hugging Face的模型庫中下載預訓練好的GPT-2模型。這個模型在許多自然語言處理任務中表現出色，例如文本生成、對話系統、問答系統等。

In [30]:
```
output = model(**encoded_input)
output['last_hidden_state'].shape
```

Out[30]:
```
torch.Size([1, 5, 768])
```

這段程式碼將使用加載的GPT-2模型對編碼後的輸入進行前向傳播，並獲取模型的最後一層隱藏狀態（ `last_hidden_state` ）的形狀。這個隱藏狀態包含了模型對輸入的理解和處理。

```
output = model(**encoded_input)
output['last_hidden_state'].shape
```

- `model(**encoded_input)` ：通過將 `encoded_input` 解包為 `input_ids` 和 `attention_mask` 來調用模型。這裡 `encoded_input` 包含了經過GPT-2分詞器處理後的輸入。
- `output['last_hidden_state'].shape` ：獲取模型的輸出，其中 `last_hidden_state` 表示模型的最後一層隱藏狀態，其形狀將被列印出來。

這個隱藏狀態的形狀通常是 `(batch_size, sequence_length, hidden_size)` ，其中 `batch_size` 是批次大小， `sequence_length` 是輸入序列的長度， `hidden_size` 是隱藏單元的維度大小。

這裡顯示的 `torch.Size([1, 5, 768])` 表示模型的最後一層隱藏狀態的形狀解釋如下：

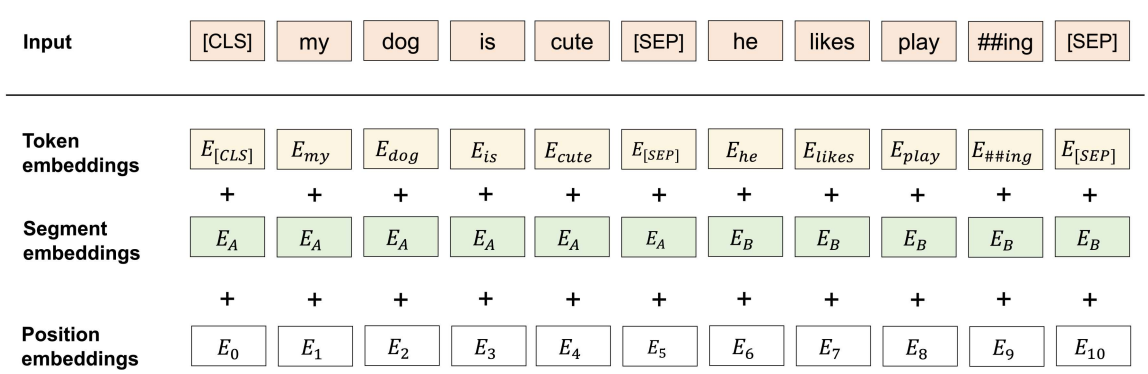
- `1` ：表示批次大小（batch size），這裡是指一次處理了一個輸入序列。
- `5` ：表示序列長度（sequence length），即輸入序列中的詞彙數量。在這個例子中，輸入的句子被分為了五個詞彙。
- `768` ：表示隱藏單元的維度大小（hidden size），這是模型內部表示每個詞彙或標記的特徵向量的大小。

因此，這個形狀顯示了模型對於給定輸入句子的每個詞彙都生成了一個 768 維的隱藏狀態。

Bidirectional pre-training with BERT

In [7]:
```
Image(filename='figures/16_13.png', width=700)
```

Out[7]:

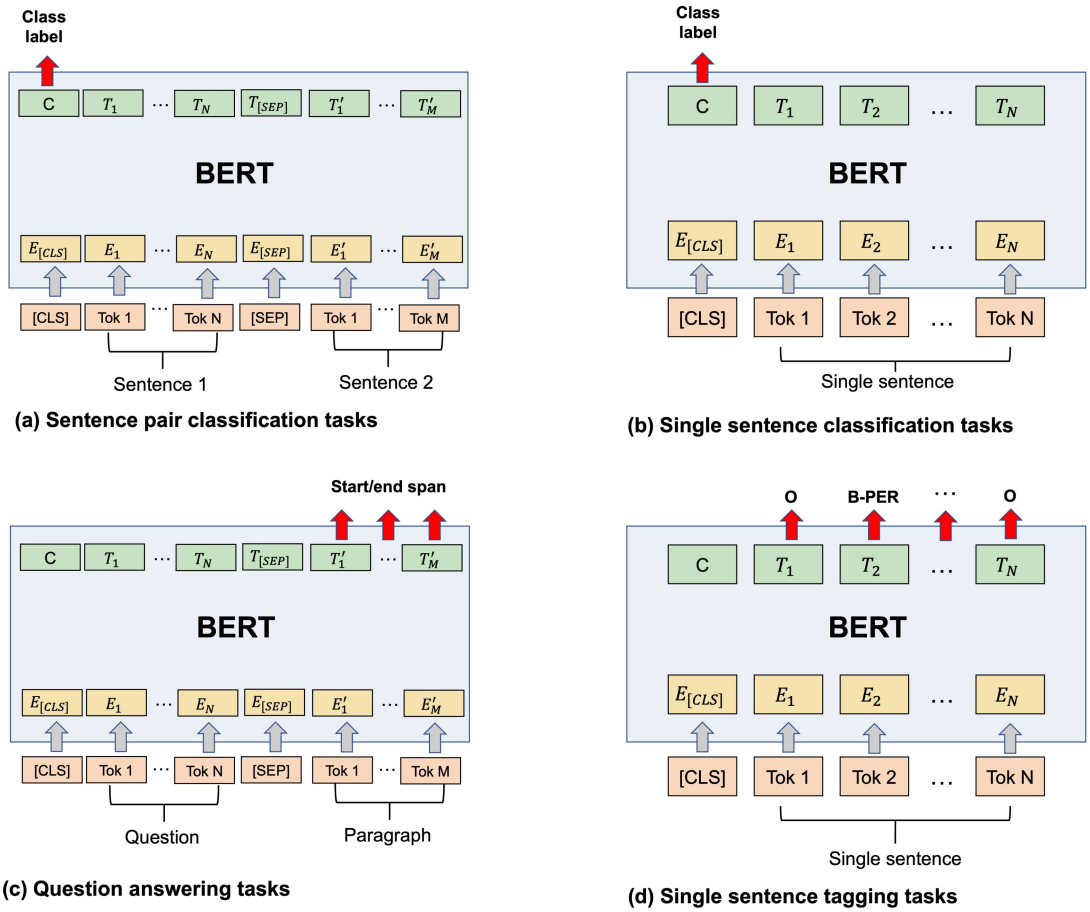


In [9]:
```
Image(filename='figures/16_14.png', width=600)
```

Out[9]:

In [11]:
```
Image(filename='figures/16_15.png', width=800)
```

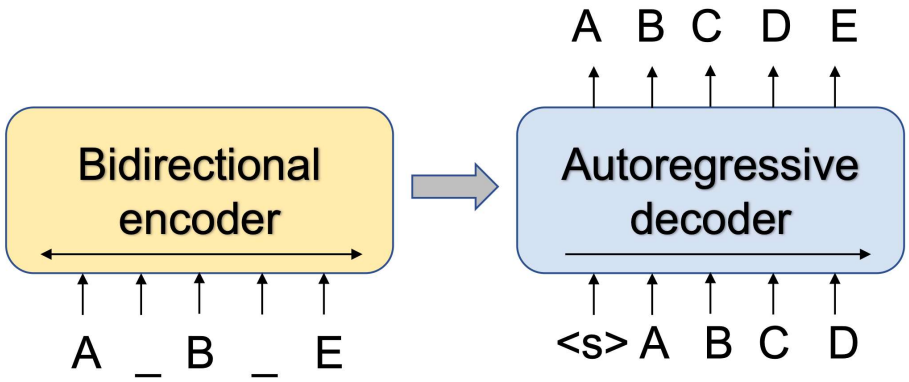
Out[11]:



The best of both worlds: BART

In [13]:
```
Image(filename='figures/16_16.png', width=500)
```

Out[13]:



Readers may ignore the next cell.

In [2]:
```
! python ../.convert_notebook_to_script.py --input ch16-part2-gpt2.ipynb --
```

```
[NbConvertApp] WARNING | Config option `kernel_spec_manager_class` not rec
ognized by `NbConvertApp`.
[NbConvertApp] Converting notebook ch16-part2-gpt2.ipynb to script
[NbConvertApp] Writing 2690 bytes to ch16-part2-gpt2.py
```