第 5 章 使用管道工具

5.1 管道工具介绍

HuggingFace 有一个巨大的模型库，其中一些是已经非常成熟的经典模型，这些模型即使不进行任何训练也能直接得出比较好的预测结果，也就是常说的 Zero Shot Learning。

使用管道工具时，调用者需要做的只是告诉管道工具要进行的任务类型，管道工具会自动分配合适的模型，直接给出预测结果，如果这个预测结果对于调用者已经可以满足需求，则不再需要再训练。

管道工具的 API 非常简洁，隐藏了大量复杂的底层代码，即使是非专业人员也能轻松使用。

5.2 使用管道工具

5.2.1 常见任务演示

1. 文本分类

使用管道工具处理文本分类任务，代码如下：

```
#第 5 章/文本分类
from transformers import pipeline
classifier = pipeline("sentiment-analysis")
result = classifier("I hate you")[0]
print(result)
result = classifier("I love you")[0]
print(result)
```

可以看到，使用管道工具的代码非常简洁，把任务类型输入 pipeline()函数中，返回值即为能执行具体预测任务的 classifier 对象，如果向具体的句子输入该对象，则会返回具体的预测结果。示例代码中预测了 I hate you 和 I love you 两句话的情感分类，运行结果如下：

```
{'label': 'NEGATIVE', 'score': 0.9991129040718079}
```

```
    {'label': 'POSITIVE', 'score': 0.9998656511306763}
```

从运行结果可以看到，I hate you 和 I love you 两句话的情感分类结果分别为 NEGATIVE 和 POSITIVE，并且分数都高于 0.99，可见模型对预测结果的信心很强。

2. 阅读理解

使用管道工具处理阅读理解任务，代码如下：

```
    #第 5 章/阅读理解
    from transformers import pipeline
    question_answerer=pipeline("question-answering")
    context=r"""
    Extractive Question Answering is the task of extracting an answer from a text
given a question. An example of a
    question answering dataset is the SQuAD dataset, which is entirely based on
that task. If you would like to fine-tune
    a model on a SQuAD task, you may leverage the examples/PyTorch/question-
answering/run_squad.py script.
    """
    result=question_answerer(
       question="What is extractive question answering?",
       context=context,
    )
    print(result)
    result=question_answerer(
       question="What is a good example of a question answering dataset?",
       context=context,
    )
    print(result)
```

在这段代码中，首先以 question-answering 为参数调用了 pipeline()函数，得到了 question_answerer 对象。context 是一段文本，也是模型需要阅读理解的目标，把 context 和关于 context 的一个问题同时输入 question_answerer 对象中，即可得到相应的答案。

注意：问题的答案必须在 context 中出现过，因为模型的计算过程是从 context 中找出问题的答案，所以如果问题的答案不在 context 中，则模型不可能找到答案。

运行结果如下：

```
    {'score': 0.6177279949188232, 'start': 34, 'end': 95, 'answer': 'the task of
extracting an answer from a text given a question'}
    {'score': 0.5152303576469421, 'start': 148, 'end': 161, 'answer': 'SQuAD
dataset'}
```

在示例代码中问了关于 context 的两个问题，所以此处得到了两个答案。

第 1 个问题翻译成中文是“什么是抽取式问答？”，模型给出的答案翻译成中文是“从

给定文本中提取答案的任务”。

第 2 个问题翻译成中文是“问答数据集的一个好例子是什么？”，模型给出的答案翻译成中文是“SQuAD 数据集”。

3. 完形填空

使用管道工具处理完形填空任务，代码如下：

```
    #第 5 章/完形填空
    from transformers import pipeline
    unmasker=pipeline("fill-mask")
    from pprint import pprint
    sentence='HuggingFace is creating a <mask> that the community uses to solve
NLP tasks.'
    unmasker(sentence)
```

在这段代码中，sentence 是一个句子，其中某些词被<mask>符号替代了，表明这是需要让模型填空的空位，运行结果如下：

```
    [{'score': 0.17927466332912445,
      'token': 3944,
      'token_str': ' tool',
      'sequence': 'HuggingFace is creating a tool that the community uses to solve
NLP tasks.'},
     {'score': 0.11349395662546158,
      'token': 7208,
      'token_str': ' framework',
      'sequence': 'HuggingFace is creating a framework that the community uses
to solve NLP tasks.'},
     {'score': 0.05243551731109619,
      'token': 5560,
      'token_str': ' library',
      'sequence': 'HuggingFace is creating a library that the community uses to
solve NLP tasks.'},
     {'score': 0.034935347735881805,
      'token': 8503,
      'token_str': ' database',
      'sequence': 'HuggingFace is creating a database that the community uses to
solve NLP tasks.'},
     {'score': 0.02860259637236595,
      'token': 17715,
      'token_str': ' prototype',
      'sequence': 'HuggingFace is creating a prototype that the community uses
to solve NLP tasks.'}]
```

原问题翻译成中文是“HuggingFace 正在创建一个社区用户，用于解决 NLP 任务的____。”，

模型按照信心从高到低给出了 5 个答案，翻译成中文分别是“工具”“框架”“资料库”“数据库”“原型”。

4. 文本生成

使用管道工具处理文本生成任务，代码如下：

```
#第 5 章/文本生成
from transformers import pipeline
text_generator=pipeline("text-generation")
text_generator("As far as I am concerned, I will",
               max_length=50,
               do_sample=False)
```

在这段代码中，得到了 text_generator 对象后，直接调用 text_generator 对象，入参为一个句子的开头，让 text_generator 接着往下续写，参数 max_length=50 表明要续写的长度，运行结果如下：

```
[{'generated_text': 'As far as I am concerned, I will be the first to admit
that I am not a fan of the idea of a "free market." I think that the idea of a
free market is a bit of a stretch. I think that the idea'}]
```

这段文本翻译成中文后为就我而言，我将是第 1 个承认我不支持“自由市场”理念的人，我认为自由市场的想法有点牵强。我认为这个想法……

5. 命名实体识别

命名实体识别任务为找出一段文本中的人名、地名、组织机构名等。使用管道工具处理命名实体识别任务，代码如下：

```
#第 5 章/命名实体识别
from transformers import pipeline
ner_pipe=pipeline("ner")
sequence = """Hugging Face Inc. is a company based in New York City. Its
headquarters are in DUMBO,
therefore very close to the Manhattan Bridge which is visible from the
window."""
for entity in ner_pipe(sequence):
    print(entity)
```

运行结果如下：

```
{'entity': 'I-ORG', 'score': 0.99957865, 'index': 1, 'word': 'Hu', 'start':
0, 'end': 2}
{'entity': 'I-ORG', 'score': 0.9909764, 'index': 2, 'word': '##gging',
'start': 2, 'end': 7}
{'entity': 'I-ORG', 'score': 0.9982224, 'index': 3, 'word': 'Face', 'start':
8, 'end': 12}
```

```
    {'entity': 'I-ORG', 'score': 0.9994879, 'index': 4, 'word': 'Inc', 'start':
13, 'end': 16}
    {'entity': 'I-LOC', 'score': 0.9994344, 'index': 11, 'word': 'New', 'start':
40, 'end': 43}
    {'entity': 'I-LOC', 'score': 0.99931955, 'index': 12, 'word': 'York', 'start':
44, 'end': 48}
    {'entity': 'I-LOC', 'score': 0.9993794, 'index': 13, 'word': 'City', 'start':
49, 'end': 53}
    {'entity': 'I-LOC', 'score': 0.98625815, 'index': 19, 'word': 'D', 'start':
79, 'end': 80}
    {'entity': 'I-LOC', 'score': 0.95142674, 'index': 20, 'word': '##UM', 'start':
80, 'end': 82}
    {'entity': 'I-LOC', 'score': 0.93365884, 'index': 21, 'word': '##BO', 'start':
82, 'end': 84}
    {'entity': 'I-LOC', 'score': 0.9761654, 'index': 28, 'word': 'Manhattan',
'start': 114, 'end': 123}
    {'entity': 'I-LOC', 'score': 0.9914629, 'index': 29, 'word': 'Bridge',
'start': 124, 'end': 130}
```

可以看到，模型识别中的原文中的组织机构名为 Hugging Face Inc，地名为 New York City、DUMBO、Manhattan Bridge。

6. 文本摘要

使用管道工具处理文本摘要任务，代码如下：

```
    #第 5 章/文本摘要
    from transformers import pipeline
    summarizer = pipeline("summarization")
    ARTICLE = """ New York (CNN)When Liana Barrientos was 23 years old, she got
married in Westchester County, New York.
    A year later, she got married again in Westchester County, but to a different
man and without divorcing her first husband.
    Only 18 days after that marriage, she got hitched yet again. Then, Barrientos
declared "I do" five more times, sometimes only within two weeks of each other.
    In 2010, she married once more, this time in the Bronx. In an application for
a marriage license, she stated it was her "first and only" marriage.
    Barrientos, now 39, is facing two criminal counts of "offering a false
instrument for filing in the first degree," referring to her false statements on
the
    2010 marriage license application, according to court documents.
    Prosecutors said the marriages were part of an immigration scam.
    On Friday, she pleaded not guilty at State Supreme Court in the Bronx, according
to her attorney, Christopher Wright, who declined to comment further.
    After leaving court, Barrientos was arrested and charged with theft of service
```

```
and criminal trespass for allegedly sneaking into the New York subway through an
emergency exit, said Detective
    Annette Markowski, a police spokeswoman. In total, Barrientos has been married
10 times, with nine of her marriages occurring between 1999 and 2002.
    All occurred either in Westchester County, Long Island, New Jersey or the Bronx.
She is believed to still be married to four men, and at one time, she was married
to eight men at once, prosecutors say.
    Prosecutors said the immigration scam involved some of her husbands, who filed
for permanent residence status shortly after the marriages.
    Any divorces happened only after such filings were approved. It was unclear
whether any of the men will be prosecuted.
    The case was referred to the Bronx District Attorney\'s Office by Immigration
and Customs Enforcement and the Department of Homeland Security\'s
    Investigation Division. Seven of the men are from so-called "red-flagged"
countries, including Egypt, Turkey, Georgia, Pakistan and Mali.
    Her eighth husband, Rashid Rajput, was deported in 2006 to his native Pakistan
after an investigation by the Joint Terrorism Task Force.
    If convicted, Barrientos faces up to four years in prison.  Her next court
appearance is scheduled for May 18.
    """
    summarizer(ARTICLE, max_length=130, min_length=30, do_sample=False)
```

示例代码中的 ARTICLE 是一个很长的文本，使用文本总结工具对这段长文本进行摘要，并设定摘要内容的长度为 30～130 个词，运行结果如下：

```
    [{'summary_text': ' Liana Barrientos, 39, is charged with two counts of
"offering a false instrument for filing in the first degree" In total, she has
been married 10 times, with nine of her marriages occurring between 1999 and 2002 .
At one time, she was married to eight men at once, prosecutors say .'}]
```

摘要翻译成中文为现年 39 岁的莉安娜 • 巴连托斯被控两项“提供虚假文书申请一级学位”的罪名。她共结过 10 次婚，其中 9 次发生在 1999—2002 年。检察官表示，她曾一度与 8 名男性同时结婚。

由于原文太长，这里不便于给出中文翻译，读者可以自行检查该摘要和原文的内容是否契合。

7. 翻译

使用管道工具处理翻译任务，代码如下：

```
    #第 5 章/翻译
    from transformers import pipeline
    translator=pipeline("translation_en_to_de")
    sentence="Hugging Face is a technology company based in New York and Paris"
    translator(sentence, max_length=40)
```

在这段代码中，首先以参数translation_en_to_de调用了pipeline()函数，得到了translator。从该参数可以看出，这是一个从英文翻译到德文的管道工具。

注意：由于默认的翻译任务底层调用的是t5-base模型，该模型只支持由英文翻译为德文、法文、罗马尼亚文，如果需要支持其他语言，则需要替换模型，具体可参见本章“替换模型执行中译英任务”和“替换模型执行英译中任务”两节。

运行结果如下：

```
    [{'translation_text': 'Hugging Face ist ein Technologieunternehmen mit Sitz
in New York und Paris.'}]
```

模型给出的德文翻译成中文是“Hugging Face是一家总部位于纽约和巴黎的科技公司。”这和英文原文的意思基本一致。

5.2.2 替换模型执行任务

1. 替换模型执行中译英任务

管理工具会根据不同的任务自动分配一个模型，如果该模型不是调用者想使用的，则可以指定管道工具使用的模型。此处以翻译任务为例，代码如下：

```
    #第5章/替换模型执行中译英任务
    from transformers import pipeline, AutoTokenizer, AutoModelForSeq2SeqLM
    #要使用该模型，需要安装sentencepiece
    !pip install sentencepiece
    tokenizer=AutoTokenizer.from_pretrained("Helsinki-NLP/opus-mt-zh-en")
    model=AutoModelForSeq2SeqLM.from_pretrained("Helsinki-NLP/opus-mt-zh-en")
    translator=pipeline(task="translation_zh_to_en",
                        model=model,
                        tokenizer=tokenizer)
    sentence="我叫萨拉，我住在伦敦。"
    translator(sentence, max_length=20)
```

在这段代码中，同样执行翻译任务，不过执行了默认的翻译任务工具不支持的中译英任务，为了支持中译英这个任务，需要替换默认的模型，代码中加载了一个模型和其对应的编码工具，再把模型和编码工具作为参数输入pipeline()函数中，得到替换了模型的翻译管道工具。最后执行一个中译英任务，运行结果如下：

```
    [{'translation_text': 'My name is Sarah, and I live in London.'}]
```

从运行结果来看，翻译的效果还是比较理想的。

2. 替换模型执行英译中任务

根据上述中译英管道工具的例子，此处再举一例英译中任务，代码如下：

```
    #第5章/替换模型执行英译中任务
    from transformers import pipeline, AutoTokenizer, AutoModelForSeq2SeqLM
```

```
#要使用该模型，需要安装 sentencepiece
!pip install sentencepiece
tokenizer=AutoTokenizer.from_pretrained("Helsinki-NLP/opus-mt-en-zh")
model=AutoModelForSeq2SeqLM.from_pretrained("Helsinki-NLP/opus-mt-en-zh")
translator=pipeline(task="translation_en_to_zh",
                    model=model,
                    tokenizer=tokenizer)
sentence="My name is Sarah and I live in London"
translator(sentence, max_length=20)
```

代码内容和中译英任务大同小异，只是替换了模型的名字，以及管道工具的翻译方向，运行结果如下：

```
[{'translation_text': '我叫萨拉,我住伦敦'}]
```

从运行结果来看，翻译的效果还是比较理想的。

5.3　小结

本章讲解了 HuggingFace 管道工具的使用，管道工具的使用非常简单，同时也能实现非常强大的功能，如果对预测的结果要求不高，则可以免于再训练的烦琐步骤。管道工具是 HuggingFace 提供的非常实用的工具。