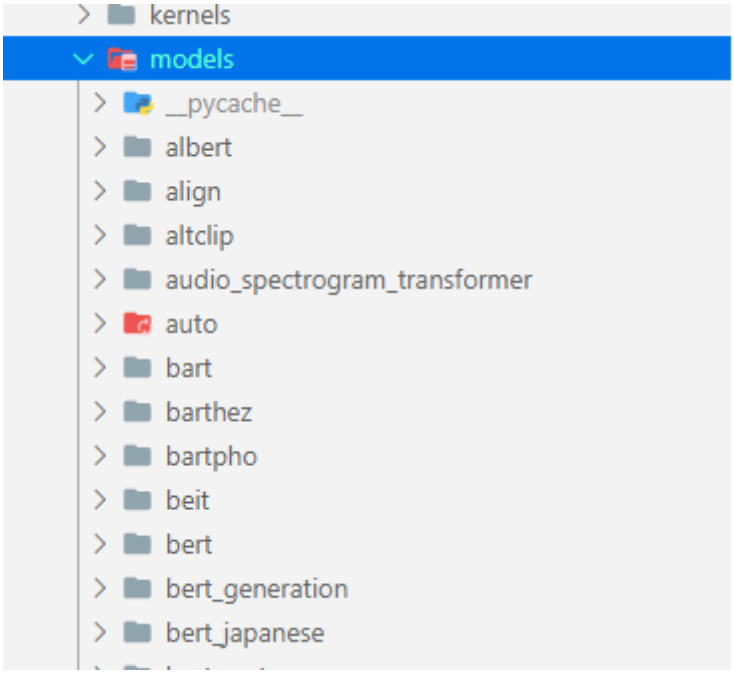
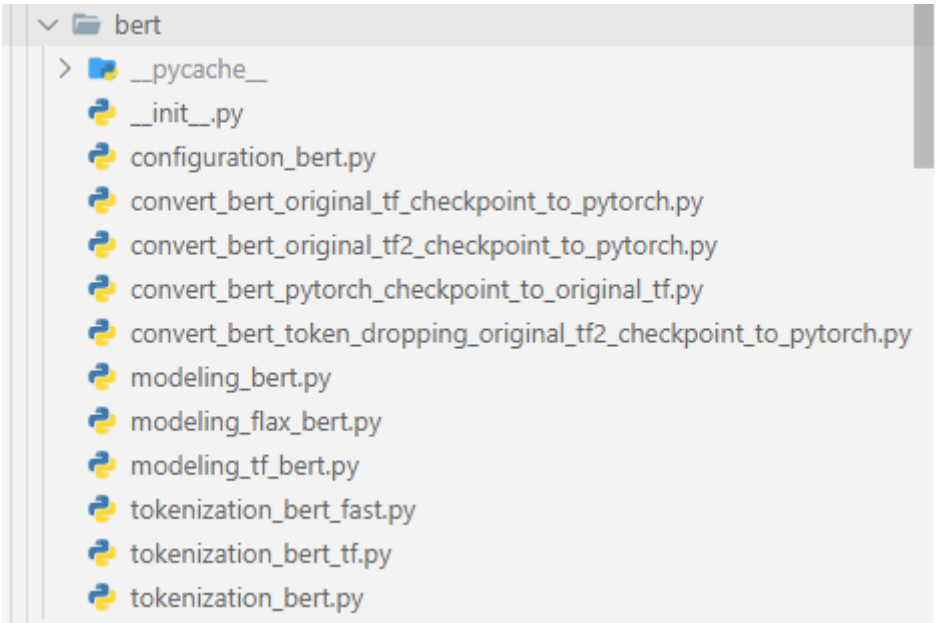
Transformers库 - Model模块

Model模块模型架构 - 以bert为例

transformers库定义了各类模型的架构，各个model定义位于：transformers\src\transformers\models



以bert为例，其文件夹下包含如下文件：



其中与模型架构相关的最重要的几个文件为：__init__.py, configuration_bert.py,modeling_bert.py三个文件；convert相关文件作用为各个深度学习框架之间的转换，tokenization相关文件实现了bert的tokenizer，此处不深入讲解。

首先着重关注一下modeling_bert.py内部定义的bert模型架构，若要关注细节推荐阅读源码

- BertModel：最终的bert类，将bert的各个模块串联起来，定义了完整的模型架构；继承于BertPreTrainedModel；主要使用了BertEncoder、BertEmbeddings两个类

- BertPreTrainedModel：继承于PreTrainedModel，而PreTrainedModel中定义了如from_pretrained，init_weights等等重要的方法，此处不予展开讲解；实际上，在transformers库中所有模型架构在定义时都继承了PreTrainedModel方法；而BertPreTrainedModel
- BertEncoder：主要使用了BertLayer类进行bert的encoder的搭建（虽然bert只有encoder），forward中hidden_states经过各个layer得到输出
- BertLayer：主要使用了BertAttention类，其实就是一层transformer encoder；而BertAttention显然实现了注意力层，但BertAttention在实现时为嵌套BertSelfAttention实现的
- BertEmbeddings类即为bert的embedding层定义

configuration_bert.py则是定义了Bert的各项参数，如维度层数等，与模型的架构定义相似，BertConfig继承于PretrainedConfig。可以参照notebook中config的相关代码进行理解。

__init__.py中我认为较为重要的两个部分为：

```
try:
    if not is_torch_available():
        raise OptionalDependencyNotAvailable()
except OptionalDependencyNotAvailable:
    pass
else:
    _import_structure["modeling_bert"] = [
        "BERT_PRETRAINED_MODEL_ARCHIVE_LIST",
        "BertForMaskedLM",
        "BertForMultipleChoice",
        "BertForNextSentencePrediction",
        "BertForPreTraining",
        "BertForQuestionAnswering",
        "BertForSequenceClassification",
        "BertForTokenClassification",
        "BertLayer",
        "BertEncoder",
        "BertSelfAttention",
        "BertLMHeadModel",
        "BertModel",
        "BertPreTrainedModel",
        "load_tf_weights_in_bert",
    ]
```

以及

```
    try:
        if not is_torch_available():
            raise OptionalDependencyNotAvailable()
    except OptionalDependencyNotAvailable:
        pass
    else:
        from .modeling_bert import (
            BERT_PRETRAINED_MODEL_ARCHIVE_LIST,
            BertForMaskedLM,
            BertForMultipleChoice,
            BertForNextSentencePrediction,
```

```
            BertForPreTraining,
            BertForQuestionAnswering,
            BertForSequenceClassification,
            BertForTokenClassification,
            BertLayer,
            BertEncoder,
            BertLMHeadModel,
            BertModel,
            BertSelfAttention,
            BertPreTrainedModel,
            load_tf_weights_in_bert,
        )
```

两部分，这两个部分相当于定义了bert文件夹下哪个文件中的类可以被import到；如若对源码不予修改的话，运行

```
from transformers import BertSelfAttention
```

是会报错的，而若将BertSelfAttention加入到以上两个模块，并使用pip install --editable /path/to/transformers 从源码构建transformers后即可成功import

如何在transformers库外利用继承修改架构实现自定义

见notebook代码

如何构建自己的完全自定义的模型

见my_model文件夹，其定义了一个最简单的模型，将其置于/src/models文件夹下，从源码构建transformers后即可使用。my_model_test.ipynb文件对自定义的简单模型进行了调用。