基础模型搭建实现

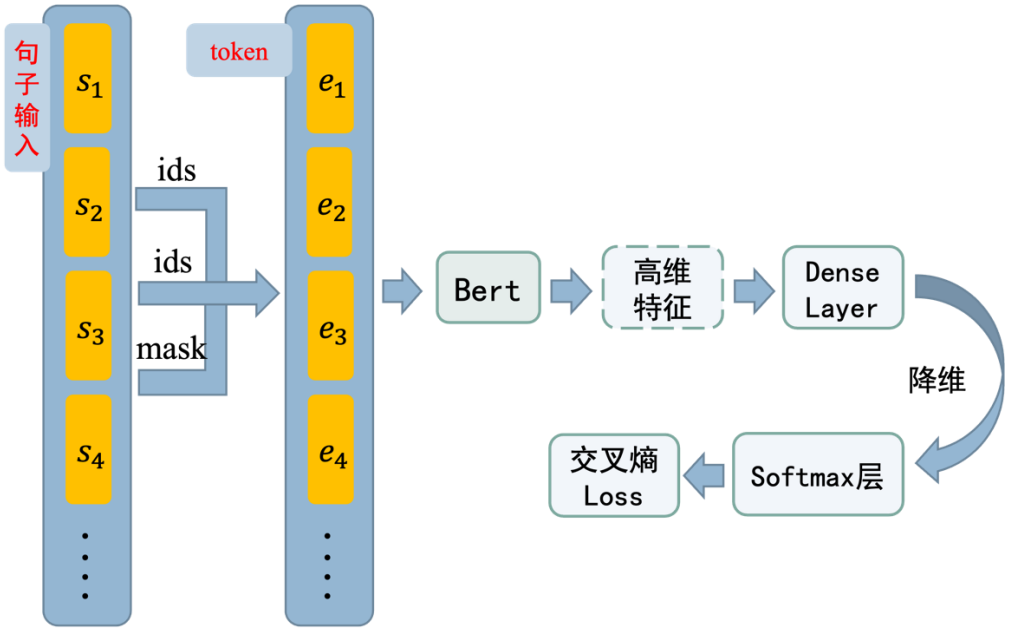

图 2 基础模型运行流程图

在系统中，我们需要首先搭建一个教师网络以在半监督学习的场景下使用。具体来说，这个模型被用来生成预测，进行模型训练，并对后续伪标签数据集的构造进行监督。基础模型运行流程图如图 2 所示。

- **模块整体框架**

```
def construct_teacher(TFModel, Config, pt_teacher_checkpoint, max_seq_length, classes, dense_dropout=0.5, attention_probs_dropout_prob=0.2, hidden_dropout_prob=0.2):

    print(Config,pt_teacher_checkpoint)
    config = Config.from_pretrained(pt_teacher_checkpoint, num_labels=classes)
    config.attention_probs_dropout_prob = attention_probs_dropout_prob
    config.hidden_dropout_prob = hidden_dropout_prob
    print(TFModel)
    print(config)
    encoder = TFModel.from_pretrained(pt_teacher_checkpoint, from_pt=True, config=config,name="teacher")

    input_ids = Input(shape=(max_seq_length,), dtype=tf.int32, name="input_ids")
    attention_mask = Input(shape=(max_seq_length,), dtype=tf.int32, name="attention_mask")
    token_type_ids = Input(shape=(max_seq_length,), dtype=tf.int32, name="token_type_ids")

```

```
    output = encoder(input_ids, token_type_ids=token_type_ids,  attention_mask=attention_mask)
    output = Dropout(dense_dropout)(output[0][:,0])
    output = Dense(classes, kernel_initializer=tf.keras.initializers.TruncatedNormal(stddev=config.initializer_range))(output)
    model = tf.keras.Model(inputs=[input_ids, token_type_ids, attention_mask], outputs=output)
    return model
```

- **参数解释**
  - **TFModel**: TensorFlow 的模型类，通常这是一个预训练的模型（如 BERT）。
  - **Config**: 相关模型的配置类，用来设置模型的一些超参数（如隐藏层大小、dropout 率等）。
  - **pt_teacher_checkpoint**: 存放预训练模型的权重以供加载的文件路径或 URL。
  - **max_seq_length**: 输入序列的最大长度。
  - **classes**: 模型输出的类别数目，即分类任务的类别数量。
  - **dense_dropout**: 全连接层中使用的 dropout 率。
  - **attention_probs_dropout_prob**: 注意力机制中的 dropout 率。
  - **hidden_dropout_prob**: 隐藏层中的 dropout 率。

1. 在这个函数中，首先我们打印配置信息和检查点路径

```
print(Config, pt_teacher_checkpoint)
```

这行代码用于调试，打印出配置类和预训练权重检查点路径。确保在后续处理中，我们知道正在使用的配置类和权重存放位置。这对于排查路径错误或配置错误，尤其在我们这个较为复杂的系统集成中非常重要。

2. 之后，我们从预训练检查点加载配置并修改 dropout 率

```
config = Config.from_pretrained(pt_teacher_checkpoint, num_labels=classes)
```

使用 Config.from_pretrained()方法加载与预训练权重相对应的配置，并为模型设定类别数量 num_labels=classes，这代表着在恶意隐写载体检测系统中，其配置将包含区分恶意载体和正常载体所需的超参数设定。

```
config.attention_probs_dropout_prob = attention_probs_dropout_prob
config.hidden_dropout_prob = hidden_dropout_prob
```

调整配置中的 dropout 参数，用于防止模型过拟合。在模型的注意力机制和隐藏层中引入 dropout，可以让模型在训练过程中随机忽略一些神经元，从而提高模型的

泛化能力。在恶意隐写载体检测中，良好的泛化能力可以让模型在处理从未见过的载体时仍然表现稳定。

3. 打印模型类和配置

```
print(TFModel)
print(config)
```

再次调用打印语句以调试，确保正在使用的模型类和配置已正确加载。通过这种方式，可以确保配置修改已被准确反映和应用，可视化调试对开发远程或迭代开发过程中识别问题非常有帮助。

4. 实例化预训练教师模型

```
encoder = TFModel.from_pretrained(pt_teacher_checkpoint, from_pt=True, config=config,
 name="teacher")
```

这一步骤使用预训练的模型权重和刚刚设定的配置实例化一个教师模型。from_pt = True 表示模型权重是从 PyTorch 格式加载的。在这一步中，教师模型从预训练权重中继承丰富的语言表征能力，并利用这些能力对输入文本进行深入处理。对于恶意隐写载体检测来说，教师模型将能够高效捕捉文本中的隐性特征，以区分恶意和正常文本。

5. 定义输入层

```
input_ids = Input(shape=(max_seq_length,), dtype=tf.int32, name="input_ids")
attention_mask = Input(shape=(max_seq_length,), dtype=tf.int32, name="attention_mask"
)
token_type_ids = Input(shape=(max_seq_length,), dtype=tf.int32, name="token_type_ids"
)
```

这里定义了模型的输入层，包括 input_ids、attention_mask 和 token_type_ids：

- **input_ids**: 代表输入文本中的每一个词的唯一 ID。
- **attention_mask**: 允许模型忽略填充的部分。对于变长的文本序列，填充常用于对齐序列长度。
- **token_type_ids**: 有助于区分属于不同语义单元的部分，如区分不同段落或对话的不同角色。

在恶意隐写载体检测中，这些输入层能够帮助模型理解和处理文本的结构和内容，保证输入序列的正确处理。

6. 通过教师模型获取输出

```
output = encoder(input_ids, token_type_ids=token_type_ids, attention_mask=attention_m
ask)
```

这一行代码将输入层的信息传递给教师模型的编码器，获取模型输出，这里的模型输出是通过一系列复杂的深度学习操作得到的特征表示，这一步骤将文本序列转化为多维特征向量。对于恶意隐写载体检测，特征表示使模型能够区分正常文本和隐藏恶意内容的文本。通过编码器，模型发现文本中的隐性特征，为分类器提供有力支持依据。

7. 添加 dropout 和全连接层

```
output = Dropout(dense_dropout)(output[0][:,0])
```

对模型输出增加一个 Dropout 层，使用 dense_dropout 指定的比例随机丢弃一些神经元，以防止过拟合。通过这种方式，增强模型的鲁棒性，使其在面对未见过的数据时仍能做出可靠的预测。

```
output = Dense(classes, kernel_initializer=tf.keras.initializers.TruncatedNormal(stddev=config.initializer_range))(output)
```

紧接着，添加一个全连接层，输出维度对应于分类任务的类别数量（classes）。这里使用截断正态分布初始化权重，以稳定训练过程。这一层的作用是依据输入特征进行分类，将基于编码器提供的特征最终判断文本是恶意隐写载体还是正常载体。

8. 构建和返回模型

```
model = tf.keras.Model(inputs=[input_ids, token_type_ids, attention_mask], outputs=output)
return model
```

构建并返回一个完整的 Keras 模型实例，该模型将输入层和输出层结合起来。具体来说：

- **inputs**: 模型的输入是 input_ids、attention_mask 和 token_type_ids。
- **outputs**: 模型的输出是通过 Dropout 和全连接层处理后的分类结果。

通过上述代码，我们实现了基础模型的搭建，利用预训练模型的深度特征提取能力，通过调整超参数和实现 Dropout 防止过拟合，最终实现对输入文本的高效分类。在恶意隐写载体检测的任务中，教师模型的强大语言理解能力确保了能够捕捉到文本中细微的差异，从而准确地区分恶意和正常载体。这种细致详尽的逐行解释帮助我们了解了每一步骤的具体功能和应用，通过这样的分析，可以为具体的任务实现提供强有力的技术保障和改进方向。