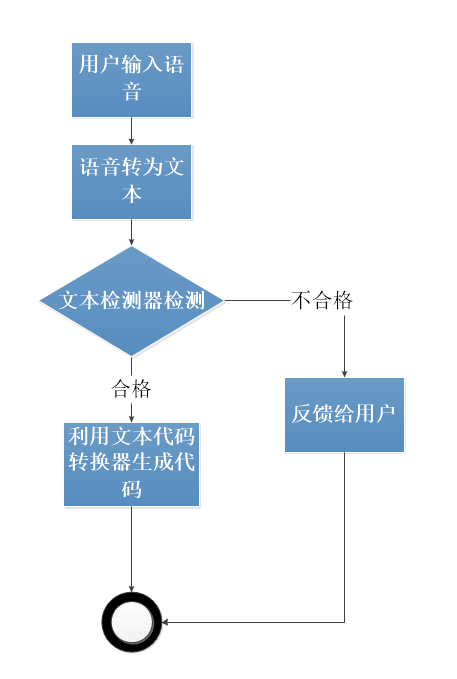
# Encorder-Decorder 方案

# 总体流程

用户的输入首先转换为自然语言文本，利用文本检测器处理获得意义明确，信息无缺漏，之后将文本输入代码转换器获得翻译后的代码。

## 流程图



# 文本代码转换器实现方案

基于编解码的方案发的架构图如下图所示，其中Block-Sentences是指由自然语言文本组成的用户描述。在经过预处理的阶段，会得到向量表示的文本作为Bert的输入。经过bert模型处理后，原先的句子向量会得到一个具有更加饱含句子上下文信息的句子向量，同时还有句子类别标记，共同作为编解码模块的Encorder输入，经过自注意力机制的编解码模块的处理后，用户的输入被解码翻译为代码，代码是属于自定义的中间代码，在中间代码的基础上，利用规则的方法转为Python、C++等。

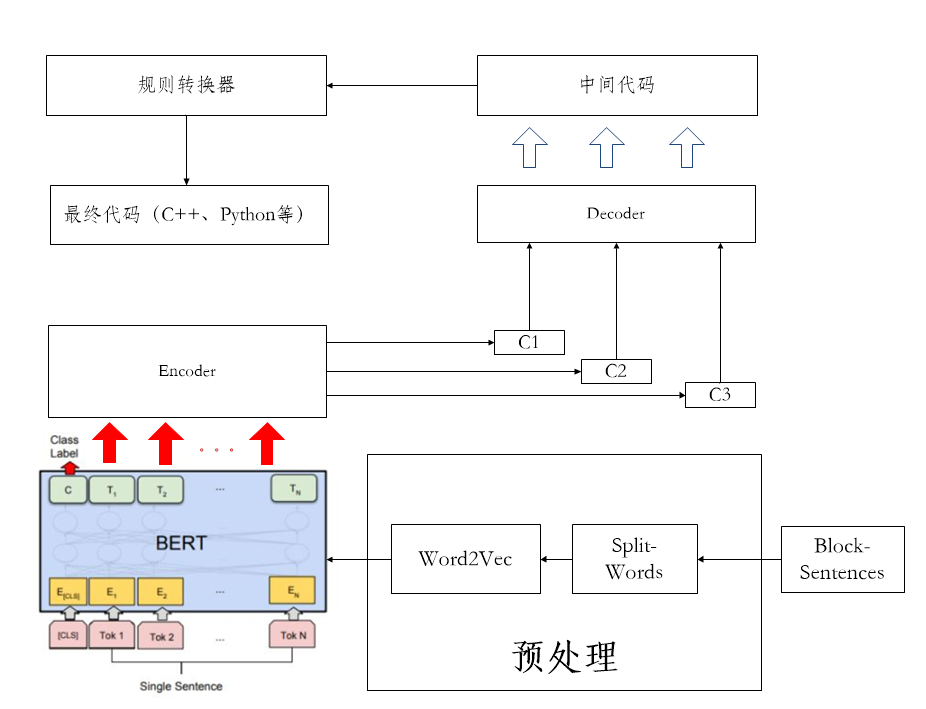


Figure 文本代码转换器

## 1 Bert模块

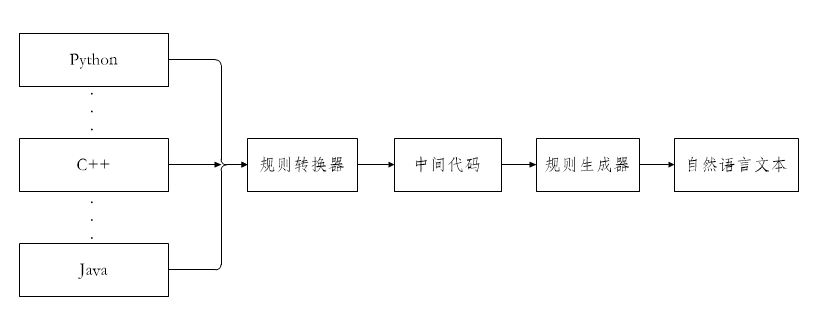
该模块在已经训练好的通用语言模型，用自制的数据集进行Fine-tunning，同时会输出句子的类别。

### 数据来源

本程序需要的数据是代码与自然语言映射文本，在现实中存在大量的代码，但是却没有其对应的自然语言文本。因此要解决数据问题就需要设计方案，使得利用已有的大量代码自然转为自然语言文本，同时要考虑到代码来源于多语言。

在本文中，我们利用规则的方法来实现造数据的过程，同时为了让模型能够有着更强的适应性，我专门设计一门中间代码，将多语言的代码来源统一转为中间代码，做到中间代码与自然语言文本的一一对应。

具体的实现流程如下图所示：



规则转换器转为将多种语言的代码转为中间代码，规则生成器为将中间代码转为自然语言文本，目前采用基于模板+规则的方法转换。

在得到大量自然语言与中间代码后，利用word2vec模型将自然语言与代码转为词向量，作为bert的训练数据。

### 训练

在通用的Bert模型上，利用1.1所造的数据进行Fine-tuning，完成模型训练。

## 2预处理模块

该模块先在1.1生成的自然语言文本做分词等预处理，利用word2vec模型训练一个词向量模型，将自然语言文本映射为向量并作为berr模块的输入。



## 3编解码模块

编解码模块应用注意力机制，通过编解码的方式将用户经过预处理模块、bert模块处理的向量，翻译成中间代码。

### 数据来源

编解码的数据与bert模块的训练数据本质来源于自然语言文本与代码对，不同在于bert模块的数据是来源于word2vec模型处理后的自然语言文本与代码，而编解码模块的数据是来源于经bert模块处理后的自然语言文本与代码。

### 训练

编解码模块会利用基于Transformer的模型进行训练，数据则如3.1所提。

## 附-实现难点与关键点

1. 数据需要兼容多语言，包括主流语言，按目前方案需要对每种语言编码一个规则转换器转为中间代码。
2. 中间代码的形式需要足够有拓展性以支持多种语言的多种语法；同时为了训练更加准确，中间代码表现形式要尽量简单。
3. 第一二步的数据质量将决定训练效果，另外数据的数量也将决定训练质量，因此在训练模型时，要考虑数据的可持续制造，模型的可增量训练。

# 文本检测器方案

文本检测器对用户的文本，首先输入文本代码器训练得到的word2vec模型获得词向量，在输入bert获得类别，根据类别及其类别对应的规则，对文本进行信息检测，如果合格输入代码转换器生成代码，不合格让用户补全。

## 流程图

