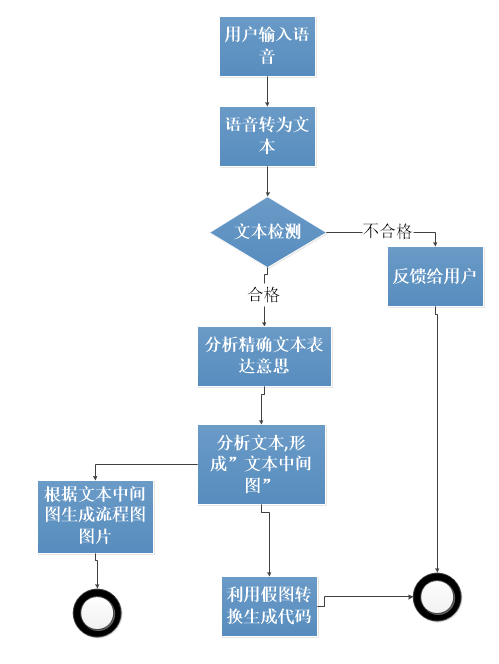
# 语音编程

## 流程图



## 方案实现关键步骤

（1）语音转文本:语音转文本(此处调用三方接口实现)

（2）文本检测:文本检测就是对文本进行分类，对于用户输入中，去掉“无关文本”，也就是不是描述代码逻辑的文本去掉；

（3）分析精确文本表达的意思: 根据用户输入判断是否需要对一些模糊的地方进行明确，

明确的部分分为两种,一种是系统根据自带规则进行模糊匹配纠正;另外一种是使用人机对话技术,反问用户直到明确内容.

（4）分析文本:这一步通过对文本进行解析,利用NLP相关技术形成中间结构(文本中间图).

（5） 利用文本中间图生成流程图或者代码.

### 检测分类

对文本检测分类，将没有逻辑的文本剔除掉-可以看出是一个二分类的问题.

(1)前期通过基于规则的方法进行剔除:通过对文本进行分句，逐句进行判断，判断的方法通过对文本进行分类（类别包括 定义类、计算类、判断类、循环类等），然后根据分类结果，判断句子是否含有各个类别所需要的成分（如变量，循环范围等），如果段落无法分类成功或者分类成功但是没有所需要的成分，就判断其是非法文本。

(2)后期可以改进使用机器学习的方法,利用大量数据进行学习得到分类器,将文本进行分类.

#### 文本分类

##### 分类类别

1. 定义语句；

例子：定义一个整形变量a；

1. 计算语句；

将a赋值为2；

1. 循环语句；

循环整个A数组

1. 判断语句if型。

如果a大于3；

1. 判断语句else型；
2. 判断语句else if 型；

（7） 排序

##### 分类器

多分类器（或者多标签分类）

##### 分类规则

1. **定义类别的规则：首先**选择外文作为变量；在**直接或者间接**修饰该变量的前面的所有定语中，选择’**词性且最近距离**‘b词性的’定语作为**类型,同时判断再利用该词语进行类别分类得到词语的类型**；得到类型与变量，还需要进一步判断是否是数组，利用生成代码即可。
2. **计算类型的规则**：**首先**对语句做符号转换处理（大于换为>，赋值转为=，等于转为==）,选择外文作为变量，如果没有就是**最近的变量**；对上文变量对比，如果是数组类型，则将离变量最近的数字‘i’提取作为索引，然后从i开始，挑取与该变量后面的‘wp’作为计算符号，对于每个符号，选择最近的’n’或者‘m’作为对象进行计算，直到遇到下一个变量或者结束语；其他类型则挑挑取与该变量后面的‘wp’作为计算符号，对于每个符号，选择最近的’n’或者‘m’作为对象进行计算，直到遇到下一个变量或者结束语；
3. **循环类型的规则：**提取外文变量，如果没有就提取最近的变量，（注意String字符串同时视为字符数组）。同时提取数值对作为范围，如果没有范围就是遍历整个数组。
4. **判断语句规则：**将语句分为if或者else，并做符号转换处理，大于换为>, 等于换为==，然后从头到尾按顺序提取外文，符号，构成表达式，及时判断语句表达式。

#### 分句原则

分句只用“。！？”作为分句， 注意没有使用“；”

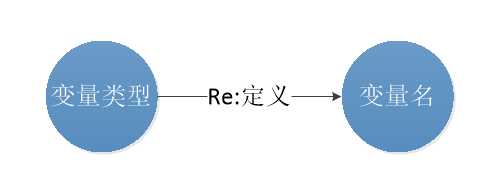
#### 用户输入规则

用户输入应该一个句子为一段完整意思，也就是说，假设是一个判断语块，应该以“。”结尾。

### 分析文本形成中间图

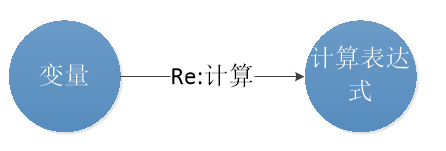
文本中间图是一段自然语言文本的”图表示”,在实现中,暂定为从文本中剥离杂乱信息,抽取出半结构化信息(信息表现形式将结合知识图谱生成的相关技术).将这些知识表示融汇成”文本中间图”,也就是图数据结构。

（1）定义关系图结构：



生成代码：变量类型 变量名；

（2）计算关系：



生成代码：变量 = 计算表达式子

（3）判断关系：



有三种判断类型分别是if、else、elif。

else的变量为NULL，表达式也为NULL

IF,生成代码是：IF(变量+表达式)

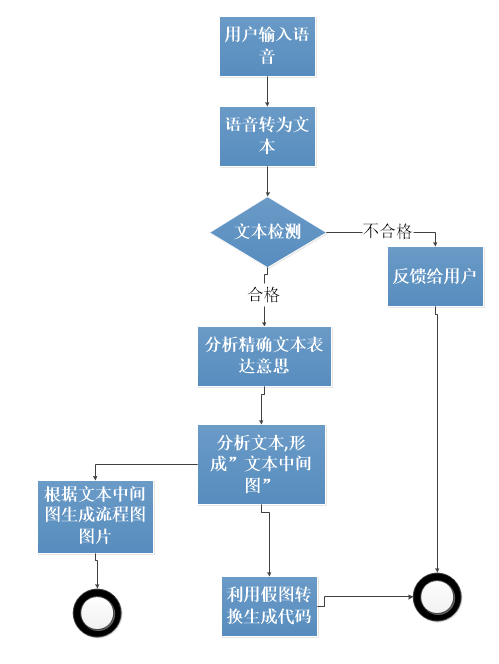
ELIF,生成代码是：IF(变量+表达式)

（4）循环关系



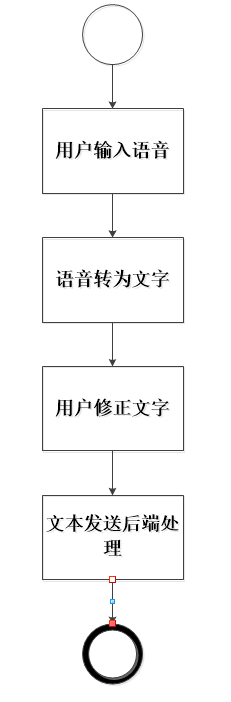
生成代码：在循环范围遍历变量

## 概要设计



## 详细设计

### 小程序与预处理模块



### 分析模块

#### 文本检测

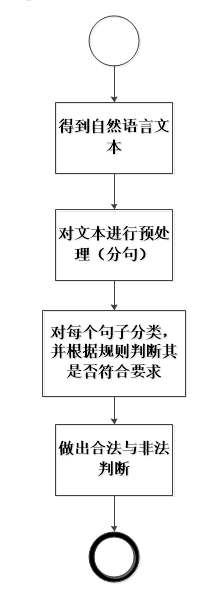


图1-4-1 文本解析流程图

#### 文本精确与分析和假图生成

对分句的自然语言文本逐渐明确意思，可以用填槽的方法来完成；假设用户说对数组进行排序，排序没有明确指示，这时候反问用户让用户“填槽”来告诉系统用哪种排序，当将整个句子明确意思后，结束。



#### 生成流程图或者代码

根据假图直接按相应转化规则转为代码或者流程图。

#### 关键变量或者数据结构（待开发过程增补）

（1）isInLoop: int 用于判断是否需要跳出循环模块 w1

规则：如果isInLoop大于0当前句子是最后一个语句或者是句号，则生成代码后同时生成#defineYKH$；否则只生成代码。如果当前句子判断为循环类型，生成#defineZKH$且isInLoop增加1。

（2）isInJudge：int 用于判断是否在判断框里，

规则：如果isInJudge大于0当前句子是最后一个语句或者是句号或是分号或者下一语句是else或者elseif类型，则生成代码后同时生成#defineYKH$。如果当前句子判断为Judge类型，生成#defineZKH$且isInJudge增加1。

# 自然语言生成用例图（未完善）

## 实现方案-思路

1. 对语句提取关系实体对
2. 提取关系实体对的关系
3. 生成SPO三元组，并使用spo三元组生成用例图

## 方案实现细节

### 关系分类类型

关联、泛化、包含、扩展



图2-1 用例图分类关系

### 分类原则

“参与者实体”：等价于图2-1的“参与者；“用例实体”：等价于图中2-1的“用例””

1. 同个句子中“参与者实体”如果不具有并列关系，则“参与者实体”之间为泛化关系。
2. 同个句子中“参与者实体”与“用例实体”之间为关联关系。
3. 同个句子中“用例实体”与“用例实体”，则是泛化、包括或者拓展关系，需要进一步分类

### 关键算法

1. **提取“用例实体”：**
2. **提取“参与者实体”；**
3. **实体之间关系提取算法;**
4. **依存关系传播算法**
5. **实体补充算法**

#### 提取“用例实体”-算法步骤

1. 参考NER的方法, 可以使用CRF, 标注训练集标注来进行训练,进而标注出：用例实体和参与者实体
2. 基于模式匹配的方法, 根据语法分析, 往往“用例实体”为v-n对，且v的依存关系为n
3. 如果句子有多个v-n对，经过“依存关系传播算法”，往往具有关系的v-n对，会依赖于同一个词语（往往是动词）

#### 提取“参与者实体”-算法步骤

（1）2.2.3.1节（1）方法

（2）往往“参与者实体”词性为n，且左边没有依赖于该参与者实体的动词；在依存关系中往往为“SBV”或者“VOB”，如果同时有“SBV”或者“VOB”的参与者实体出现在句子中，两者往往依赖于同一个动词；注意这里还需要利用‘’实体补充算法”补充‘参与者实体”的成分。

#### 实体之间的关系提取-算法步骤

（1）“参与者实体”之间：在依存关系中往往为“SBV”或者“VOB”， 参与者实体”不具有并列关系且两者往往依赖于同一个动词；注意这里还需要利用‘’实体补充算法”补充“参与者实体”的成分。**关系为：泛化关系。**

（2）“参与者实体”与“用例实体”：句子中有参与者实体与用例实体，且两者经过“依存关系传播算法”判别依赖于同一个词语（往往是动词）。**关系为：关联关系。**

**（3）**“用例实体”与“用例实体”间关系：经过“依存关系传播算法”，往往具有关系的v-n对，会依赖于同一个词语（往往是动词）。

#### 依存关系传播算法

该算法只要是根据依存关系，将输入的词语所依赖的关系链提取出来。

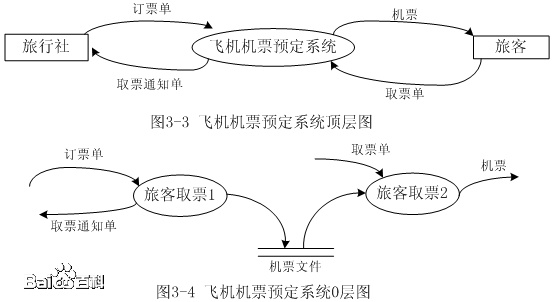
#### 实体补充算法

该算法主要依赖依存句法关系，将依赖该实体的“ATT”/”COO“/”ADV”的所有成本补充为同一个实体。

# 自然语言生成数据流图

## 实现方案-思路

1. 根据语句提取数据前后两个状态S1和S2
2. 提取S1和S2 的主处理关系R1，
3. 生成S1→R1→S2数据流情况，最终构成
4. 生成数据流图



# 基础知识

