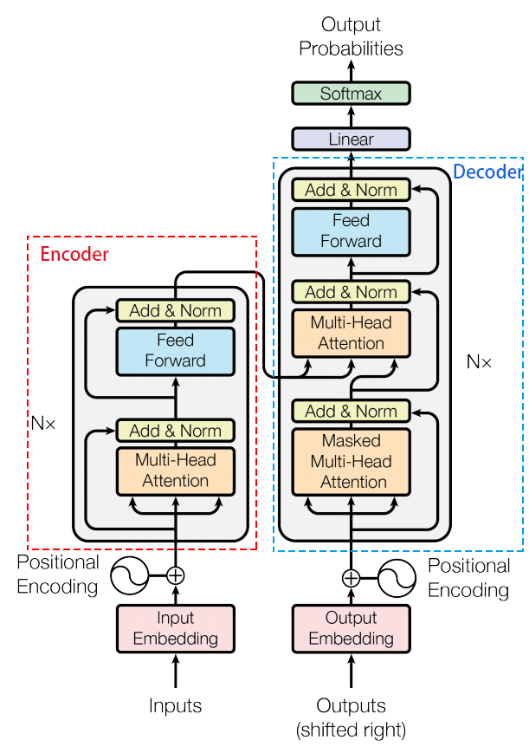
# 应用实践 基于Transformer的任务型机器人

## 实践目的

1. 了解Transformer神经网络的整体结构；
2. 了解任务型对话系统的实现流程；

## 二、实践原理

### 2.1 Transformer网络

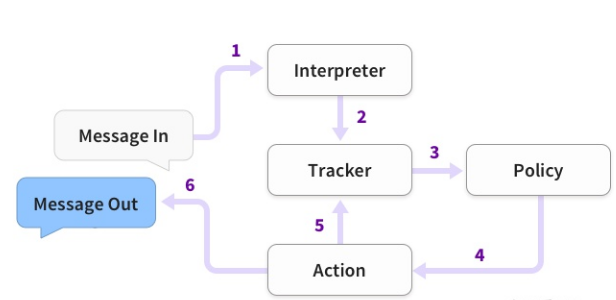


《Attention Is All You Need》是一篇Google提出的将Attention思想发挥到极致的论文。这篇论文中提出一个全新的模型，叫 Transformer，抛弃了以往深度学习任务里面使用到的 CNN 和 RNN ，该实验中使用的Bert就是基于Transformer构建的，这个模型广泛应用于NLP领域，例如机器翻译，问答系统，文本摘要和语音识别等等方向。

### 2.2 Rasa框架

Rasa是一个用于自动文本和基于语音的对话的开源机器学习框架。了解消息，保持对话以及连接到消息传递通道和API。

Rasa分为Rasa core和 Rasa nlu两部分：Rasa core用于指导会话流，而Rasa nlu用于理解和处理文本以提取信息(实体)；Rasa了解用户想说的内容（Rasa NLU - 实体和意图提取），然后根据上下文信息对其进行适当的操作谈话（Rasa Core）。



用户传入的消息先流经Interpreter部分即NLU部分，这部分主要负责对用户输入信息进行意图分析，并提取所有实体信息；然后Rasa Core会将识别出的用户intent传递给Tracker，Tracker主要负责跟踪会话的状态(conversation state)；再用Policy记录Tracker的当前状态，并对回复用户的action进行选择，同时将action记录在Tracker中；最后执行action返回的结果进行输出即可完成一次交互。

### 2.3 React框架

React是用于构建用户界面的JavaScript库，主要用于构建UI。在React里可以传递多种类型的参数，如声明代码，帮助你渲染出UI、也可以是静态的HTML DOM元素、也可以传递动态变量、甚至是可交互的应用组件。

## 三、设备

本实验所需硬件如下：

1. Ubuntu系统或windows系统PC机一台；
2. 基础Python环境及其包安装。

## 四、实践操作

4.0 环境准备

1、 安装python3.6环境

2、 安装依赖包：rasa2.2

4.1 模型训练与更新（在带GPU的电脑进行训练）

| **结构** | **作用** |
| --- | --- |
| actions/ | 自定义 rasa的actions 的代码文件 |
| data/ | 存放Rasa NLU 训练数据、Rasa stories 数据等数据 |
| piplines/ | 设置Rasa框架使用的Piplines |
| config.yml | Rasa NLU 和 Rasa Core 的配置文件 |
| credentials.yml | 定义和其他服务连接的一些细节，例如Rasa api接口 |
| domain.yml | Rasa domain 文件 |
| endpoints.yml | 和外部消息服务对接的 endpoins 细则 |

更改data中的定义意图intent，在stories中定义多轮对话的对话过程。

（具体原理可见附件：《Rasa智能机器人助手》）

在根目录下执行 rasa train

4.2 启动交互界面

1、 启动rasa服务端：

一个命令行执行：rasa run --cors “\*”

另外一个命令行执行：rasa run actions

2、启动客户端：点击index.html即可在本机打开网页端进行交互。

## 五、实践效果介绍

1. Index.html效果图（Web端）：



①初始化机器人和用户的基本信息(botAvater,userAvater)

②初始化NavBar及title

③快捷回复：(内容参考[quickReplys.js](https://github.com/Dustyposa/rasa_bot_front/blob/main/src/compoments/quickReplys.js))



## 六、结论与建议