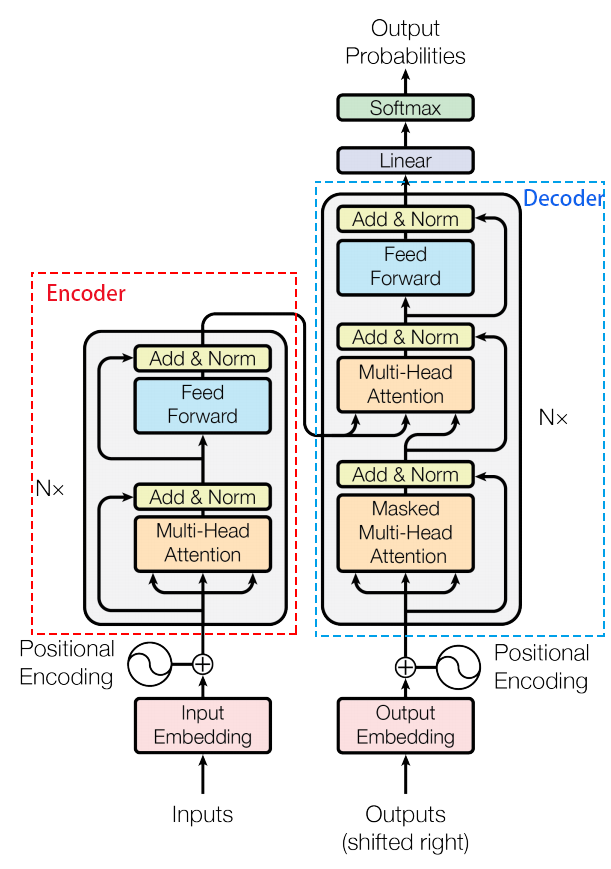
# 应用实践 基于Transformer的任务型机器人

## 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流程；
3. 通过使用深度学习模型，熟悉模型部署的流程。

## 二、实践原理

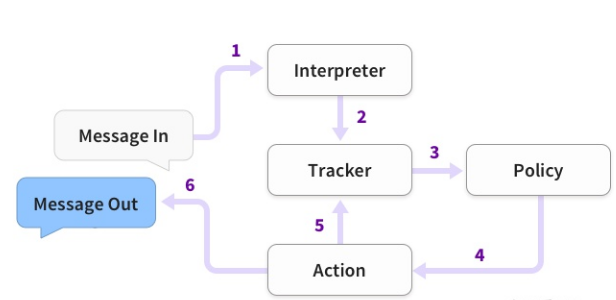
### 2.1 Transformer网络



### 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。

React具有以下特点：

（1）声明式设计：React 使创建交互式 UI 变得轻而易举。为你应用的每一个状态设计简洁的视图，当数据变动时 React 能高效更新并渲染合适的组件；

（2）组件化: 构建管理自身状态的封装组件，然后对其组合以构成复杂的 UI；

（3）高效：React通过对DOM的模拟，最大限度地减少与DOM的交互；

（4）灵活：无论你现在使用什么技术栈，在无需重写现有代码的前提下，通过引入 React 来开发新功能。

## 三、设备

本实验所需硬件如下：

1. Ubuntu系统或windows系统机器一台；
2. 基础Python环境及其包安装。

## 四、实践操作

4.1 训练模型：rasa train

4.2 启动rasa服务：

rasa run

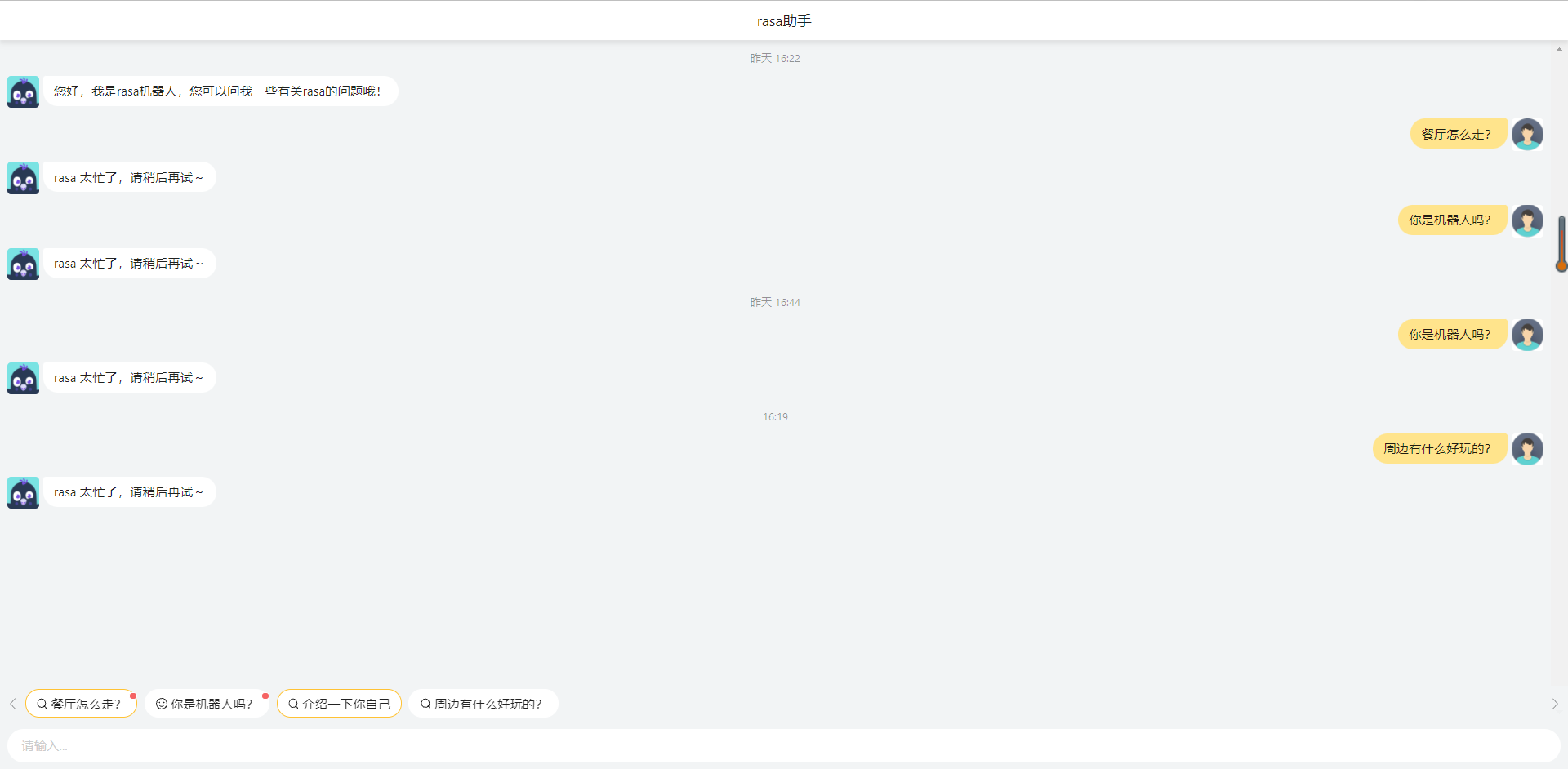
rasa run actions

4.3 启动网络端服务

点击index.html即可在本机打开网页端进行交互。

## 五、实践数据

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))



Rasa bot中的response分为两种类型：

① Text



② Image



## 六、数据分析

## 七、结论与建议