**Rasa智能机器人助手**

**Rasa智能机器人的基本原理、应用实践与具体设计**

目 录

[一、Rasa框架 3](#_Toc21446)

[1、 Rasa的组成及原理 3](#_Toc17139)

[2、 Rasa框架的文件组成 4](#_Toc1651)

[二、 使用Rasa框架训练智能机器人助手 4](#_Toc1741)

[1、 具体实现 4](#_Toc27483)

[1.1 编写数据文件 4](#_Toc31307)

[1.2 下载Bert-chinese模型 5](#_Toc3934)

[1.3 修改配置文件 5](#_Toc25458)

[1.4 模型训练 5](#_Toc25218)

[2、 Rasa NLU Pipeline 5](#_Toc5081)

[2.1 Transformer的应用 6](#_Toc21228)

[2.2 配置 6](#_Toc16474)

[2.3 拓展（待考究） 7](#_Toc19530)

[三、使用React框架进行前端设计 7](#_Toc28477)

[1、React框架 7](#_Toc10191)

[1.1 React简介 7](#_Toc30920)

[1.2 特点 8](#_Toc14591)

[2、React项目文件构成 8](#_Toc13666)

[3、 操作细节 9](#_Toc17875)

[4、配置 11](#_Toc3490)

**一、Rasa框架**

1. **Rasa的组成及原理**

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）**

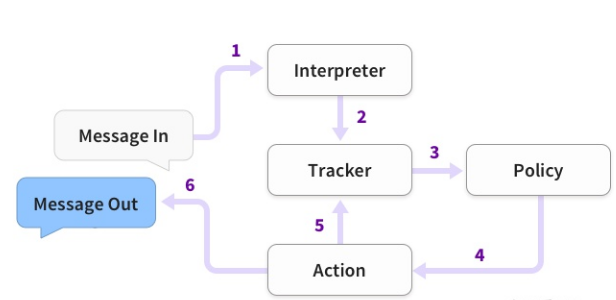


图1 基本结构

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。

1. **Rasa框架的文件组成**

| **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 |

1. **使用Rasa框架训练智能机器人助手**

本系统使用rasa框架进行智能机器人的训练，机器人被设计为实现智能问答模式的机器人助手形式，可实现一些基本的对答，例如：认识到自己是机器人，实现对天气，BTC价格，电影，动漫，动物等等的用户意图检测并进行回应。

1. **具体实现**

1.1 编写数据文件

nlu.yml：该部分主要对用户意图进行分类

stories.yml：该部分主要帮助训练AI助手，里面编写着AI助手与一个用户的多个对话过程，同时包含AI助手在应该如何回复的数据。

domain.yml：该部分包含了用户的意图，AI助手的actions，以及actions中对应的内容，即给用户的具体回复。

1.2 下载Bert-chinese模型

语言模型使用的是 bert\_chinese，所以需要下载 bert\_chinese 模型。并放到 pre\_models 文件夹中，重命名为 tf\_model.h5。

curl -L https://www.flyai.com/m/bert-base-chinese-tf\_model.h5 -o pre\_models/tf\_model.h5

1.3 修改配置文件

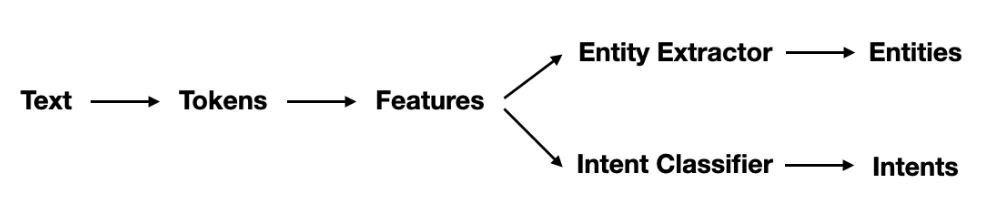
由于使用的预先训练的Bert作为语言模型，需要对配置文件model\_name,model\_weights等进行修改

1.4 模型训练

使用rasa train命令进行模型训练。

1. **Rasa NLU Pipeline**

Rasa NLU Pipeline 是在文件config.yml中进行定义，此文件描述了，Rasa通过使用管道检测意图和实体的所有步骤。它以文本作为输入开始，并一直解析，直到有实体和意图作为输出。

图 pipeline流程图

组件( Components)按顺序将用户输入处理为结构化输出。有用于实体提取( entity extraction)、意图分类( intent classification)、响应选择( response selection)、预处理( pre-processing)等的组件。

在管道中，主要有以下几种不同类型的组件：

1、语言模型(Language Models)

2、分词器(Tokenizers)

3、特征化器(Featurizers)

4、意图分类器(Intent Classifiers)

5、实体提取器(Entity Extractors)

2.1 Transformer的应用

Pipeline Components中的语言模型有MitieNLP，SpacyNLP，HFTransformersNLP这几种类型的组件。其中HuggingFace中的Transformers已经被弃用，现在LanguageModelFeaturer在实现其行为。

组件部分应用语言模型、特定的标记化和特征化来计算训练数据中每个示例的序列和句子级别表示。包括[LanguageModelTokenizer](https://rasa.com/docs/rasa/components#languagemodeltokenizer)和[LanguageModelFeaturizer](https://rasa.com/docs/rasa/components" \l "languagemodelfeaturizer)也将此组件部分的输出用于下游NLU模型。

Transformer系列模型在各大领域的应用都较为广泛，例如语音、CV、机器翻译等等都有较为出色的表现，作为语言模型也是一个较好的选择。

为使系统表现更好，我们需要在pipeline中使用预先训练过的字向量，那么我们则需要加载所需的预训练模型。本系统使用中文Bert的Bert-base-chinese预训练模型。

由于语言模型使用Bert的预训练模型，那么分词部分也需要与之对应，为实现中文分词，这部分需要自己实现一个 tokenizer即CustomBertTokenizer。

分词器的作用为将话语中的每个单词分割成一个单独的标记(token)，通常分词器的输出是一个单词列表。我们也可能会得到单独的标点符号，这取决于分词器和我们的设置。

对于英语，我们通常使用[WhiteSpaceTokenizer](https://rasa.com/docs/rasa/components/#whitespacetokenizer)，但是对于非英语，通常选择其他的。比如，对于非英语的欧洲语言，通常选择[SpaCy](https://rasa.com/docs/rasa/components/" \l "spacytokenizer)；对于汉语则通常选择[Jieba](https://rasa.com/docs/rasa/components/" \l "jiebatokenizer)。也可以采用自定义的方式进行分词。

LanguageModel Featurizer是使用经过预先训练的语言模型，如Bert、GPT-2等，为完整句子提取相似的上下文向量表示。本系统使用的则是Bert预训练模型中带的LanguageModel Featurizer。

2.2 配置

本系统使用的是本系统使用中文Bert的Bert-base-chinese预训练模型,通过参数指定要加载的语言模型。model\_name。此外，还可以通过指定参数来指定所选语言模型的体系结构变化。model\_weights。如果保留为空，则使用原始Transformers库加载的默认模型体系结构。Tokenizer这部分需要自己实现一个 Bert tokenizer,[Featurizer](https://rasa.com/docs/rasa/components#languagemodelfeaturizer)将使用Bert语言模型自带的LanguageModel Featurizer。（ps:python version 3.7+）

pipeline:

name: piplines.tokenizers.CustomBertTokenizer

name: LanguageModelFeaturizer

model\_name: bert

model\_weights: bert-base-chinese

2.3 拓展（待考究）

本系统使用的是Bert作为语言模型，我们还可以使用其他的语言模型，具体如下表所示：

| Language Model“model\_name” | Parameter “model\_weights” |
| --- | --- |
| BERT | bert |
| GPT | gpt |
| GPT-2 | gpt2 |
| XLNet | xlnet |
| DistilBERT | distilbert |
| RoBERTa | roberta |

**三、使用React框架进行前端设计**

**1、React框架**

1.1 React简介

React是用于构建[用户界面](https://baike.baidu.com/item/%E7%94%A8%E6%88%B7%E7%95%8C%E9%9D%A2/6582461" \t "https://baike.baidu.com/item/react/_blank)的[JavaScript](https://baike.baidu.com/item/JavaScript/321142)库，主要用于构建[UI](https://baike.baidu.com/item/UI/393851)。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。

1.2 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。

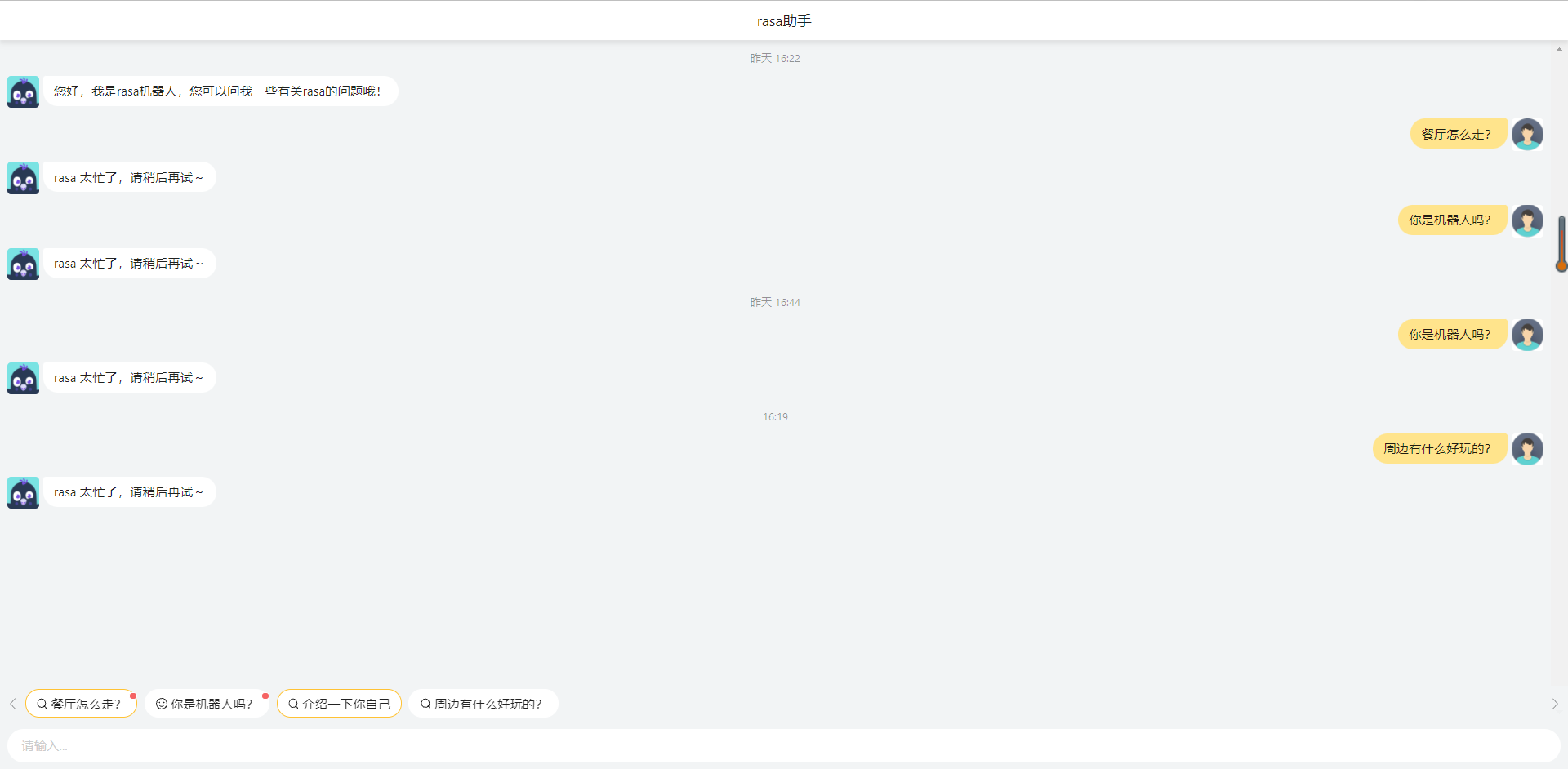
**2、React项目文件构成**

| **文件构成** |
| --- |
| chatbot\_font/ |
| REAMDE.md |
| package.json |
| .gitignore |
| public/ |
| favicon.ico |
| index.html (浏览器执行的主文件) |
| manifest.json |
| src/ |
| App.css |
| App.js |
| App.test.js |
| index.css |
| index.js(主文件调用的js文件) |
| logo.svg |

index.html文件和vue类似，都需要定义一个范围，react会在这个范围内生效，

index.js文件要将你想要显示到页面上的代码渲染到指定的范围里

1. **操作细节**
2. Index.html效果图（Web端）：



1. InitCongig.js：

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" \o "quickReplys.js))



1. msgManager.js:

Rasa bot中的response分为三种类型：

① Text



② Image



③ Card

(待定图片)

根据三种不同的response实现如图所示三种不同的BotResponse界面效果。（ps:需先对RasaResponse先进行parse）对card的选择响应详见Card.js。

1. rasaUtil.js:

前端通过rasa提供的api服务实现前后端交互。

**4、配置**

使用Facebook开源的脚手架工具webpack搭建一个完全不依赖服务器的开发环境,实现前后端的分离。