Documentação do ChatBot RASA IFRS-RG

1. Introdução

O presente projeto se refere ao desenvolvimento de um Chatbot para a secretária do Instituto Federal do Rio Grande do Sul campus Rio Grande (IFRS-RG). O objetivo do chatbot é devolver ao usuário, seja ele aluno ou colaborador, informações acerca de suas dúvidas relacionadas ao uso de determinados serviços disponibilizados pela instituição, como:

- Calendário Acadêmico
- Comprovante de matrícula
- Contato dos professores
- Cursos disponíveis
- Grade de horários
- Informações sobre inscrição/matrícula
- Informações sobre rematrícula
- Requerimentos ou formulários
- Como acessar os sistemas acadêmicos

Para a construção do chatbot foi utilizado o RASA (framework para desenvolvimento de chatbots), SpaCy, NLP, e a linguagem de programação Python.

SpaCy é uma biblioteca de software *open source* para o Processamento Avançado de Linguagem Natural. Spacy está escrita nas linguagens de programação Python e Cython. A biblioteca Spacy fornece uma análise sintática rápida e precisa, chamada de reconhecimento de entidades e acesso pronto aos vetores de palavras. Essa biblioteca também oferece tokenização, detecção de limites de sentença, vetores de palavras integrados e alinhamento na sequência original com alta precisão (Khumar, 2020).

Usando o pip, as versões SpaCy estarão no gerenciador de pacotes de seu sistema.

RASA é um *framework* de código aberto para o desenvolvimento de chatbots e assistentes com base em Inteligência Artificial e aprendizado de máquina.

O chatbot em RASA é construído tendo por base a linguagem Python e a compreensão da linguagem natural (NLU). O chatbot funciona respondendo questões ou aspectos solicitados pelos usuários, através de suas ações customizadas, histórias e regras, estas que por sua vez, seguem o fluxo da conversa e intenções.

O treinamento e a execução são realizados a partir de comandos no terminal, ou ainda via redes sociais como Whatsapp, Telegram, entre outras, ou também via web chat.

A construção dos fluxos de conversa entre o bot e o usuário é realizada através do framework RASA. Ainda, a biblioteca SpaCy é utilizada para comunicar com o bot no idioma em Português, além de auxiliar no processamento de Linguagem Natural.

2. **Escopo**

A secretária do IFRS RG recebe através das suas mídias sociais, por parte de alunos e colaboradores, muitas dúvidas, questionamentos, acerca de assuntos estudantis, e muitas vezes não é possível responder em um curto período de tempo as solicitações, dado o número de pessoal disponível para tal tarefa. Dessa forma, o chatbot não visa substituir funcionários, mas sim auxiliá-los com rapidez e assertividade e principalmente automatizar interações repetitivas, como por exemplo, no caso do IFRS RG; solicitações/explicações de como fazer a matrícula, ou a rematrícula, calendário acadêmico vigente, grade de disciplinas para determinado curso, os cursos e as modalidades destes que são ofertados na instituição de ensino. Assim, o chatbot desenvolvido, busca auxiliar nestes processos, sanar dúvidas e entregar ao usuário uma interação fluida e coerente.

3. **Funcionalidades**

O Chatbot tem por objetivo interpretar as mensagens enviadas a página para os seguintes fluxos de conversa:

- Calendário Acadêmico;
- Comprovante de matrícula;
- Contato dos professores;
- Cursos disponíveis;
- Grade de horários;
- Informações sobre inscrição/matrícula;
- Informações sobre rematrícula;
- Requerimentos ou formulários;
- Como acessar os sistemas acadêmicos;

De forma geral, o chatbot fornecerá as informações das perguntas do usuário, associadas aos tópicos apresentados acima, direcionando para links de acesso aos diferentes sistemas do IFRS RG, formulários ou as requisições em específico realizadas pelos usuários.

Também direcionará aos botões que contenham informações importantes.

4. Instalação

O ChatBot utiliza Python, RASA e SpaCy.

4.1 Windows

4.1.1 Instalando Python

Preferencialmente deve-se instalar python na versão 3.8, para garantir o bom funcionamento do RASA. As versões e o passo a passo da instalação do python pode ser visualizada em sua documentação.

4.1.2 Instalando o SpaCy 3.3.1

Para instalar o SpaCy 3.3.1 é preciso digitar o seguinte comando:

```
$ pip3 install rasa[spacy]
```

```
$ python3 -m spacy download pt_core_news_lg4
```

Para o ChatBot funcionar em português - Brasil, devemos baixar o pacote do idioma.

No entanto, para não termos conflitos, será baixado primeiramente o pacote em inglês e na sequência o pacote em português.

Pode-se configurar o download conforme o sistema operacional de sua máquina ou ambiente virtual (venv ou env) de trabalho.

Aviso: o projeto atual usa o pacote **pt_core_news_lg.**

Para obter informações do pacote instalado digite:

```
$ spacy info
```

4.1.3 Instalando o RASA 3.2.4

Para instalar o RASA 3.2.1, basta seguir a instalação no site oficial ou digitar:

```
pip3 install -U --user pip && pip3 install rasa
```

Para verificar se a instalação foi bem-sucedida, digite:

$ rasa --version

E para iniciar um projeto em RASA, basta dar o comando:

$ rasa init // Cria uma pasta com config iniciais
$ rasa train // Treina o modelo
$ rasa shell // Roda o último modelo treinado na CLI

4.2 Linux

A versão linux para este projeto é: Ubuntu 20.04

4.2.1 Instalando o SpaCy 3.3.1

Logo de começo, devemos instalar a dependência:

$ pip3 install rasa[spacy]

$ python3 -m spacy download pt_core_news_lg4

Para o ChatBot funcionar em português - Brasil, devemos baixar o pacote do idioma.

No entanto, para não termos conflitos, será baixado primeiramente o pacote em inglês e na sequência o pacote em português.

Pode-se configurar o download conforme o sistema operacional de sua máquina ou ambiente virtual (venv ou env) de trabalho.

Aviso: no projeto atual, utilize o pacote **pt_core_news_lg.**

Informações sobre o pacote spacy pode ser obtida pelo comando:

$ spacy info

4.2.3 Instalando o RASA 3.2.4

O RASA apresenta uma rápida instalação:

```
pip3 install -U --user pip && pip3 install rasa
```

É possível analisar a versão do RASA apenas digitando:

$ rasa --version

5. Implementando o sistema na máquina

A implementação do Chatbot é feita a partir do download ou clone do projeto no GitHub. Após este processo, basta utilizar um editor de código de sua preferência, para abrir o projeto.

Inicialmente será preciso acessar a pasta do bot em seu diretório. Logo, será possível efetuar o treinamento do bot, após alterações nos arquivos NLU, Domain, Rules ou Stories:

$ rasa train

Após baixar a aplicação é necessário passar o comando no seu terminal:

$ docker-compose up –build

Em um segundo terminal, dê o comando para executar as actions, correspondentes ao arquivo python *actions.py*:

$ rasa run actions

Para rodar localmente o chatbot é necessário, no documento endpoints.yml comentar a informação do token que conecta o chatbot ao Telegram e descomentar a informação referente ao localhost, como pode ser visualizado abaixo:

```
17  # rodando no docker
18  action_endpoint:
19    url: "http://actions:5055/webhook"
20
```

Comando para iniciar a conversa com o bot pelo terminal:

$ rasa shell

6. O projeto chatbot IFRS-RG

O chatbot Rasa leva o bot a lidar com perguntas frequentes, feitas por estudantes e colaboradores do IFRS RG, a um bate-papo de forma mais rápida. As conversas fluem através das intents, histórias e configurações do domínio.

Após a criação das intents, preparamos as respostas, às reações do bot à intenção do usuário. Fizemos esta etapa, com as respostas dentro do domínio, ou seja, no arquivo domain.yml.

Neste arquivo são geradas as respostas às questões previamente definidas.

As histórias em data / stories.md, todas as perguntas frequentes são tratadas, de acordo com a necessidade estrutural da resposta.

Através da figura 1 e suas respectivas partes (1, 2 e 3), observamos o fluxo das conversas com o chatbot.

O início (start) do assistente começa com uma interação do usuário, um oi, por meio da intent *greet* , as *utters* (*utter* - nome reservado pelo sistema para referir a uma resposta) que ocorrem na sequência (figura 1. Parte 2), buscam responder à saudação e questionar o usuário sobre suas necessidades de ajuda com os serviços fornecidos pela secretaria do IFRS- RG. Ainda, o usuário pode questionar o assistente sobre suas funções e então direcionar suas intenções para sanar as dúvidas. Os retângulos coloridos em ciano, apresentam possíveis questionamentos

dos usuários, nos demais retângulos sem coloração, é possível observar as consequentes respostas ou ações que o bot irá fornecer.

Quando o usuário buscar por requerimentos ou buscar por justificar suas faltas será requisitado o requeriment_form e a action_get_requeriments conduzirá o usuário à sua resposta. Ainda, na figura 1. Parte 1. é visualizado o caminho das solicitações email dos professores ou o email do prof, bem como quais cursos são ofertados pelo IFRS-RG (dividido em suas respectivas modalidades e posteriormente o nome do curso) e informações sobre as férias (e o direcionamento para o calendário acadêmico).

Na Figura 1. Parte 2., observamos também, as solicitações sobre matrícula ou a forma de ingresso no IFRS-RG e as *utters* e *actions* subsequentes. Uma vez que, ao sanar sua dúvida, o usuário será questionado, novamente se precisa de mais alguma informação. Se o usuário seguir com o diálogo, terá sua solicitação atendida, por exemplo,se questionar sobre um atestado de matrícula ou qual o sistema para fazer sua rematrícula, direcionando a modalidade e nome do curso, ou que seja para visualizar as disciplinas do semestre vigente (Figura 1. Parte 3) obterá uma resposta satisfatória e será novamente conduzido pelas *utters e actions* adequadas à situação. Caso o usuário encerre sua conversa, o assistente também finalizará sua conversa se despedindo devidamente.

6.1 NLU

NLU (Natural Language Understanding) extrai as informações estruturadas das mensagens do usuário. Retornando ao chatbot a intenção do usuário, e ou as entidades relacionadas a mensagem fornecida.

6.2 RULES

As Rules ou regras descrevem pequenas partes da conversação, que deverá ser seguido, criando uma regra para o caminho. Segundo a documentação do RASA, Rules é um tipo de treinamento de dados que o chatbot usa para conversar com os modelos (Figura 1).

6.3 STORIES

As Stories ou histórias são um conjunto de Intents e Actions que juntas formam um fluxo de conversa. Uma Stories é uma representação de uma conversa entre um usuário e um assistente de Inteligência Artificial, convertida em um formato específico onde as entradas do usuário são expressas como intenções (e entidades quando necessário), enquanto as respostas e Actions do assistente são expressas como nomes de Actions (Figura 1).

6.4 DOMAIN

O Domain é o arquivo responsável por conectar as rules, stories e actions. Este arquivo, é o encarregado por especificar as Intents, Entities, Slots, Forms e Actions do projeto RASA. Através dele é possível definir a configuração principal e os processos que serão realizados (Figura 1).

6.5 ACTIONS

Após cada mensagem do usuário, o modelo prevê uma ação que o assistente deve realizar posteriormente. Neste contexto, são possíveis de utilização diferentes tipos de ações (actions) para o desenvolvimento de um chatbot em RASA, sendo elas: resposta, ações personalizadas, formulários, ações básicas e ações de validação de slots.

A ação (action) utilizada mais frequentemente é a resposta, ela poderá enviar textos, imagens e até botões para o usuário. Em síntese, esta *action* retorna uma resposta ao usuário do chatbot.

A ação personalizada é uma *action* que pode executar qualquer código que for de interesse do desenvolvedor. Elas podem ser utilizadas para consultar um banco de dados ou para fazer uma chamada de API. Esta ação é um arquivo .py, e é desenvolvida utilizando a linguagem Python.

Os *forms* ou formulários são ações personalizadas, onde é possível desenvolver um formato de conversação, no qual o assistente retorna um conjunto específico de informações.

As ações padrões são ações que são incorporadas ao gerente de diálogo por padrão. De forma geral, a maioria delas é automaticamente prevista com base em certas situações de conversação. Elas também podem ser personalizadas.

Uma ação de validação de slot é um tipo especial de ação personalizada, projetada para lidar com extração personalizada e/ou validação de valores de slot. Pode ser usado para validar slots com mapeamentos predefinidos ou extrair slots com mapeamentos personalizados (Figura 1).

7. Aplicação de suporte ao chatbot

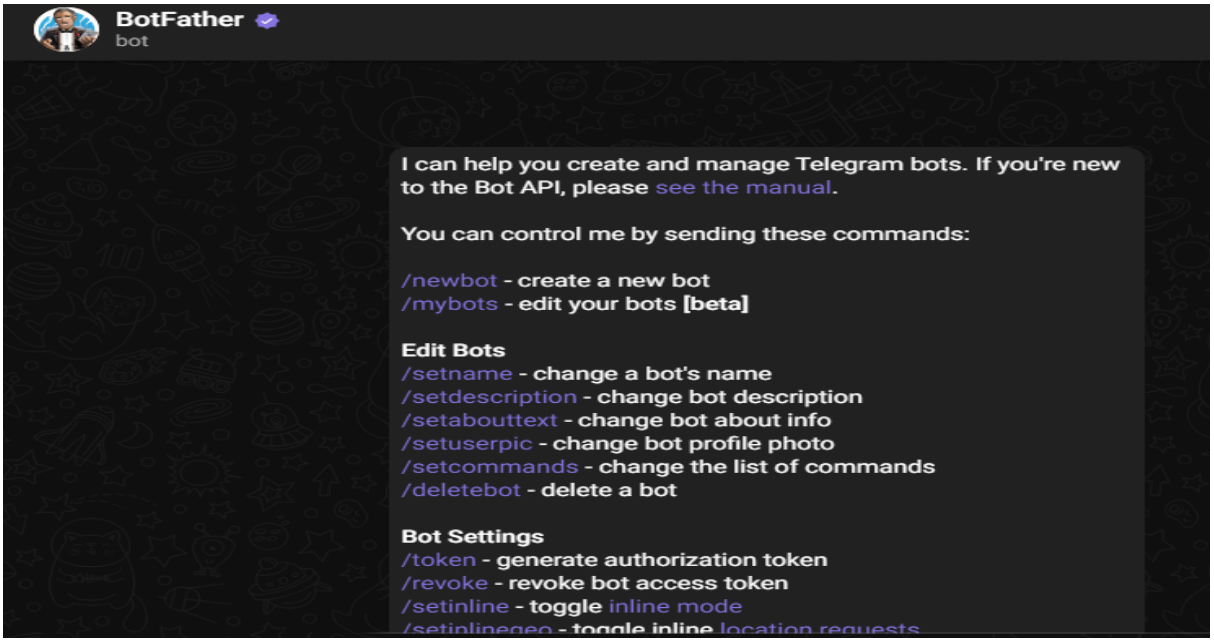
A aplicação criada foi para auxiliar as demandas de atualizações de cada semestre, ou seja, um usuário ou administrador terá que atualizar as categorias pré-definidas com informações ou documentos para que o chatbot consiga realizar suas funções de forma coesa e atualizada sem a necessidade de realizar requisições ao site, que devido n fatores pode falhar e comprometer o funcionamento do chatbot.

A imagem abaixo mostra a tela inicial da aplicação:

E no link a seguir tem se a documentação do passo a passo do desenvolvimento e a execução da aplicação. [Sistema de Controle de Informações]

8. Telegram

O telegram foi escolhido para ser o canal de comunicação com o chatbot, devido sua facilidade de criar um bot dentro da própria plataforma como mostra a imagem abaixo:

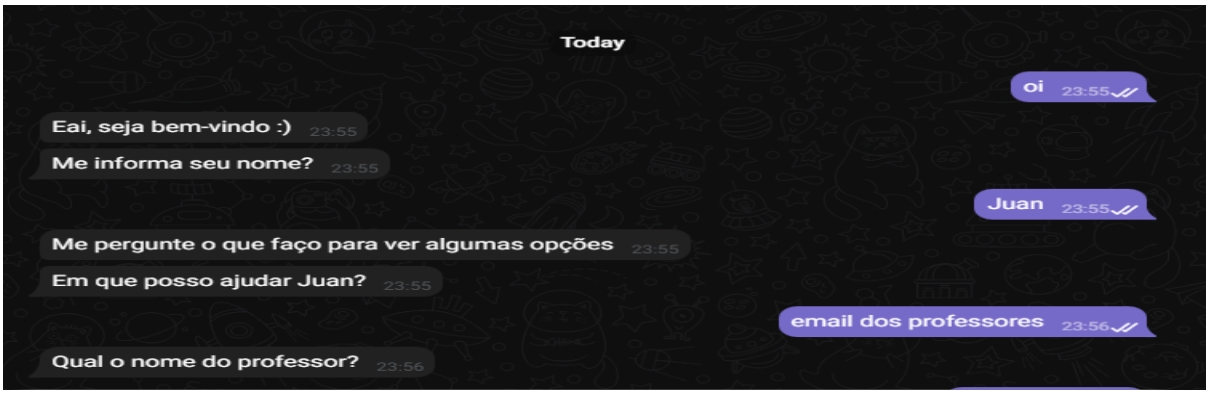


com isso foi possível criar o bot, e com as configurações de credenciais do rasa foi possível comunicar o chatbot com o telegram

```
telegram:
  access_token: "TELEGRAM_TOKEN"
  verify: "TELEGRAM_NAME"
  webhook_url: "ENDPOINT_URL/webhooks/telegram/webhook"

```

dispondo do arquivo .env, e com as variaveis telegram_token, telegram_name e endpoint_url, onde o telegram_token é o token de acesso gerado pelo botfather no próprio telegram, o telegram_name é o nome dado ao bot e o endpoint_url é a url gerada para comunicação externa.
Dispondo de todas as informações corretas será possível o uso do chatbot no telegram como na imagem abaixo:



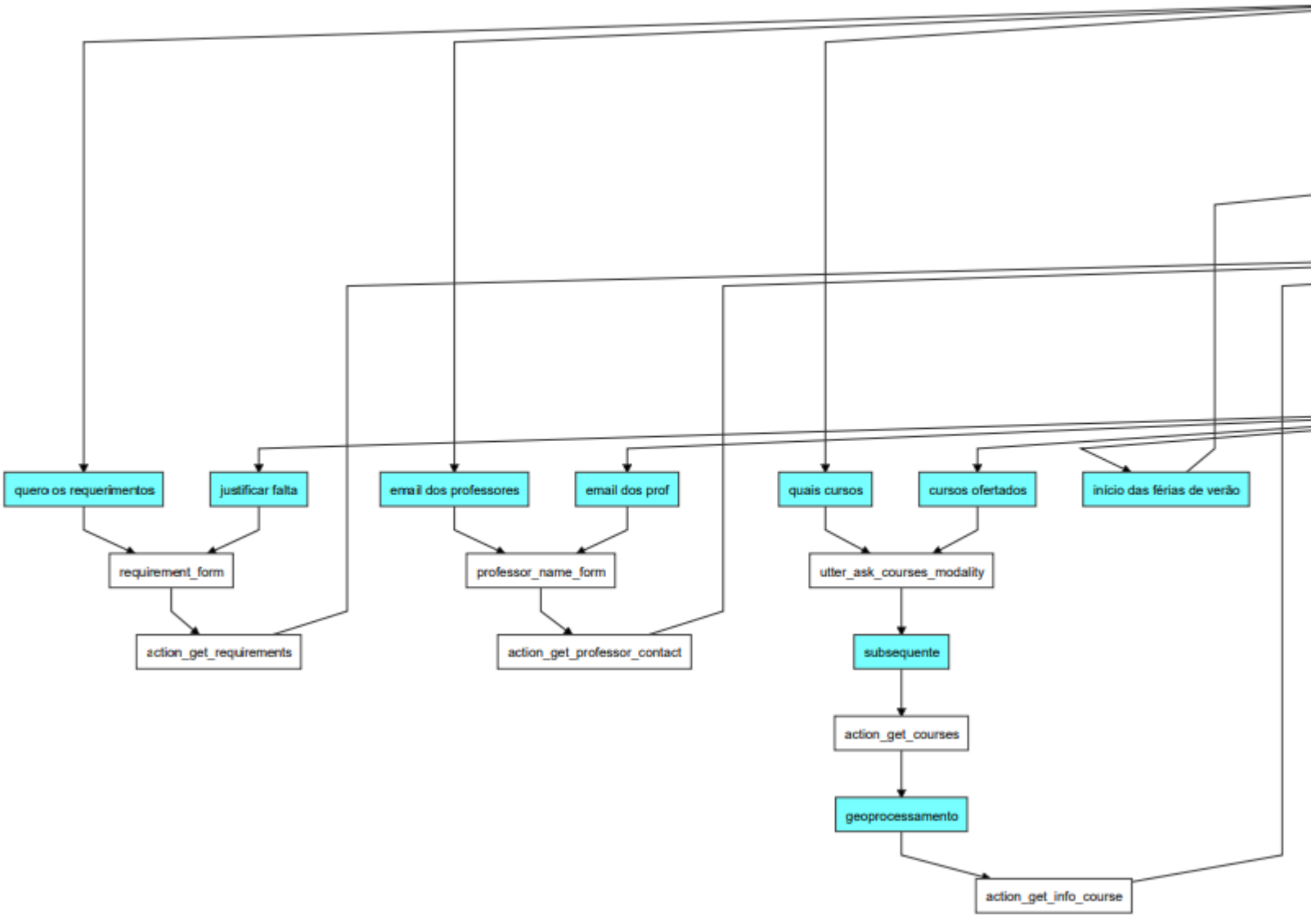


Figura 1. Parte1

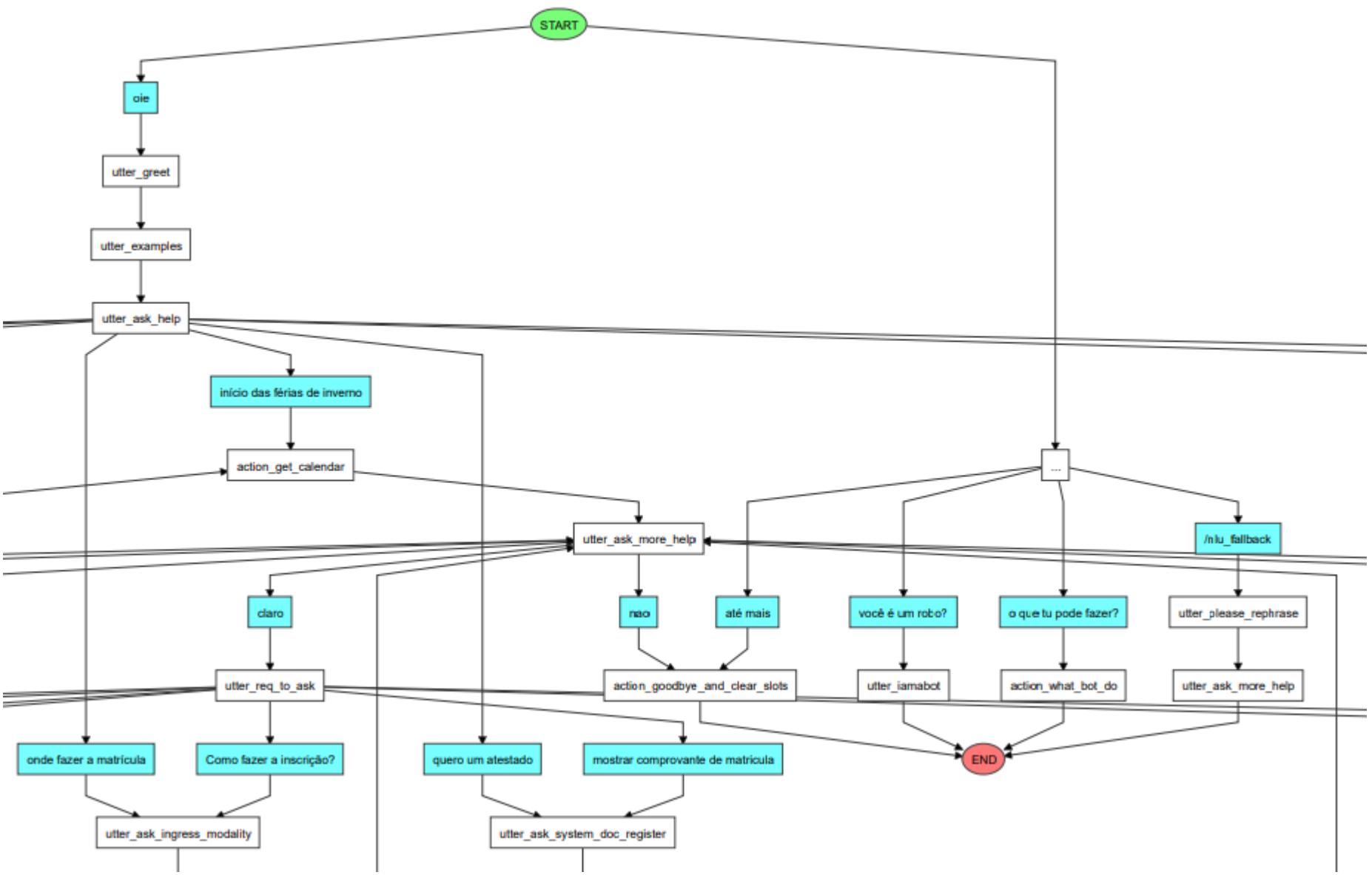

START
oie
utter_greet
utter_examples
utter_ask_help
início das férias de inverno
action_get_calendar
...
utter_ask_more_help
/nlu_fallback
claro
nao
até mais
você é um robo?
o que tu pode fazer?
utter_please_rephrase
utter_req_to_ask
action_goodbye_and_clear_slots
utter_iamabot
action_what_bot_do
utter_ask_more_help
onde fazer a matrícula
Como fazer a inscrição?
quero um atestado
mostrar comprovante de matrícula
END
utter_ask_ingress_modality
utter_ask_system_doc_register

Figura 1. Parte 2

Figura 1. Parte 3