Documentação do ChatBot RASA
FATEC Mogi Mirim - Squad 1

1. Introdução

Esse projeto consiste no desenvolvimento de um chatbot para a instituição Fatec Mogi Mirim, cujo objetivo é responder perguntas dos usuários relacionadas a vestibular. O chatbot é construído utilizando o RASA (framework utilizado para desenvolvimento de chatbots), SpaCy, NLP, e a linguagem de programação Python.

Inteligência Artificial é um campo da ciência da computação, com a intenção de aprender e realizar tarefas, pode se dizer que uma solução de inteligência artificial envolve um agrupamento de várias tecnologias, como redes neurais artificiais, algoritmos e sistemas de aprendizado. O objetivo é simular a capacidade humana ligada à inteligência para desenvolver máquinas que tenham habilidades de pensar e agir como seres humanos. A criação de algoritmos permite aprender automaticamente a partir de dados, é uma combinação de grandes volumes de dados e algoritmos inteligentes, eles permitem ao sistema ler e interpretar padrões e informações para aprender automaticamente.

Python é uma linguagem de alto nível, criada em 1990 por Guido van Rossum, no instituto nacional de pesquisa para Matemática e Ciência da Computação da Holanda. Ela possui uma sintaxe bem simples, com o uso da indentação como forma de definição de bloco de código. Seu processo de compilação, consiste na transformação código texto em bytecode, onde por sua vez, é interpretado por uma *virtual machine* (VM).

O desenvolvimento dessa linguagem foi baseado na linguagem de programação ABC, com derivação de C. Atualmente, a linguagem Python está na sua versão 3.9.5.

SpaCy é uma biblioteca de software de código aberto para Processamento Avançado de Linguagem Natural, escrita nas linguagens de programação Python e Cython. Característica linguística onde você insere texto bruto e recebe de volta um objeto, que vem com uma variedade de anotações. Quando um texto é chamado no SpaCy, primeiro ele transforma em token para produzir um doc, esse doc é processado em várias etapas diferentes conhecido como pipeline de processamento.

Arquitetura de modelo é uma função que usa apenas uma classe para quase todos tipos de camadas, não precisa criar subclasses. Usando o pip, as versões SpaCy estarão no gerenciador de pacotes de seu sistema.

NLP é uma sigla para *Natural Language Processing* (Processo de Linguagem Natural). Uma área dentro da Inteligência Artificial voltada ao entendimento da linguagem humana, servindo como intermediário entre a máquina e o ser humano. O NLP foca na criação de dispositivos tecnológicos para interação, como chatbots e assistentes virtuais. Atuando, principalmente na compreensão do que está sendo dito pelo ser humano, ele simula e estrutura a resposta apropriada. Os conceitos que englobam a NLP são aprendizado de máquina e aprendizagem profunda, pois é necessário que haja uma conexão inteligente e interativa do usuário com a máquina.

O contexto, ou intenção é essencial para a NLP para interpretar os *inputs* do usuário, os significados semânticos e sintáticos existentes dentro da conversa, análise de sentimentos e a complexidade do diálogo. Para a interação com a Inteligência Artificial, a NLP precisa dominar dois elementos: a intenção e a entidade.

RASA é um *framework* de código aberto para o desenvolvimento de chatbots e assistentes com base em Inteligência Artificial e aprendizado de máquina. A construção do chatbot em RASA é elaborada a partir da linguagem Python e a compreensão da linguagem natural (NLU, em inglês). O chatbot permite a implementação de ações customizadas, histórias e regras a serem seguidas pelo fluxo da conversa, intenções, representando a fala do usuário e as respostas do chatbot. O treinamento e a execução são realizados a partir de comandos no terminal, via Web Chat ou redes sociais como Whatsapp e Telegram.

O RASA possui dois componentes principais: o RASA NLU voltado para a classificação da intenção do usuário, capaz de entender o que foi dito para responder da forma mais apropriada. E o RASA Core, o gerenciamento do diálogo com base em aprendizado de máquina, responsável por predizer a próxima ação utilizando modelos probabilísticos como redes neurais, invés de *if/else statements*.

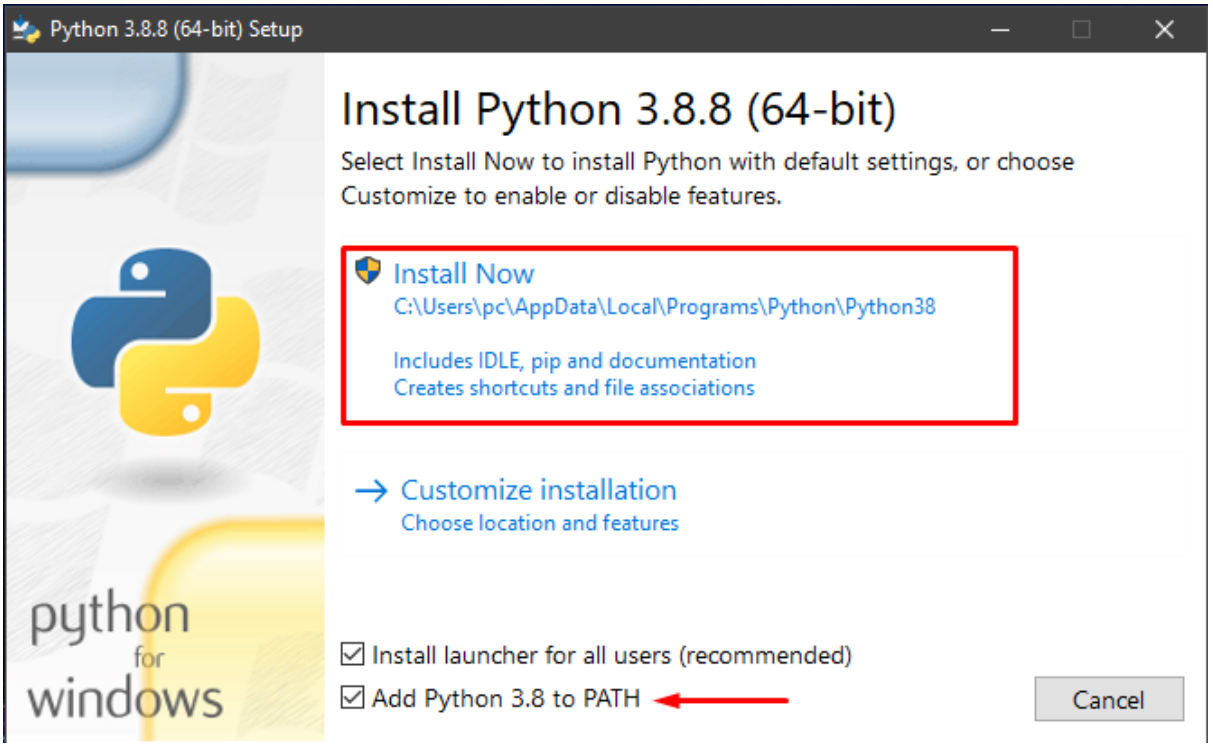
Será utilizado o framework RASA para a construção dos fluxos de conversa entre o bot e o usuário, onde serão declaradas possíveis intenções do usuário, como uma saudação de início de conversa, respostas do bot para que auxilie o usuário em algum assunto, até a despedida do usuário com o bot. As ações personalizadas, ou seja, ações que não pedem apenas uma simples resposta do bot, como a criação de um formulário, uma chamada de API ou até ações para consultar um banco de dados, será desenvolvida com a linguagem de programação Python. Será utilizado a biblioteca SpaCy, para que seja possível ter uma processamento mais avançado de Linguagem Natural, e comunicar com o bot no idioma em Português.

2. Instalação

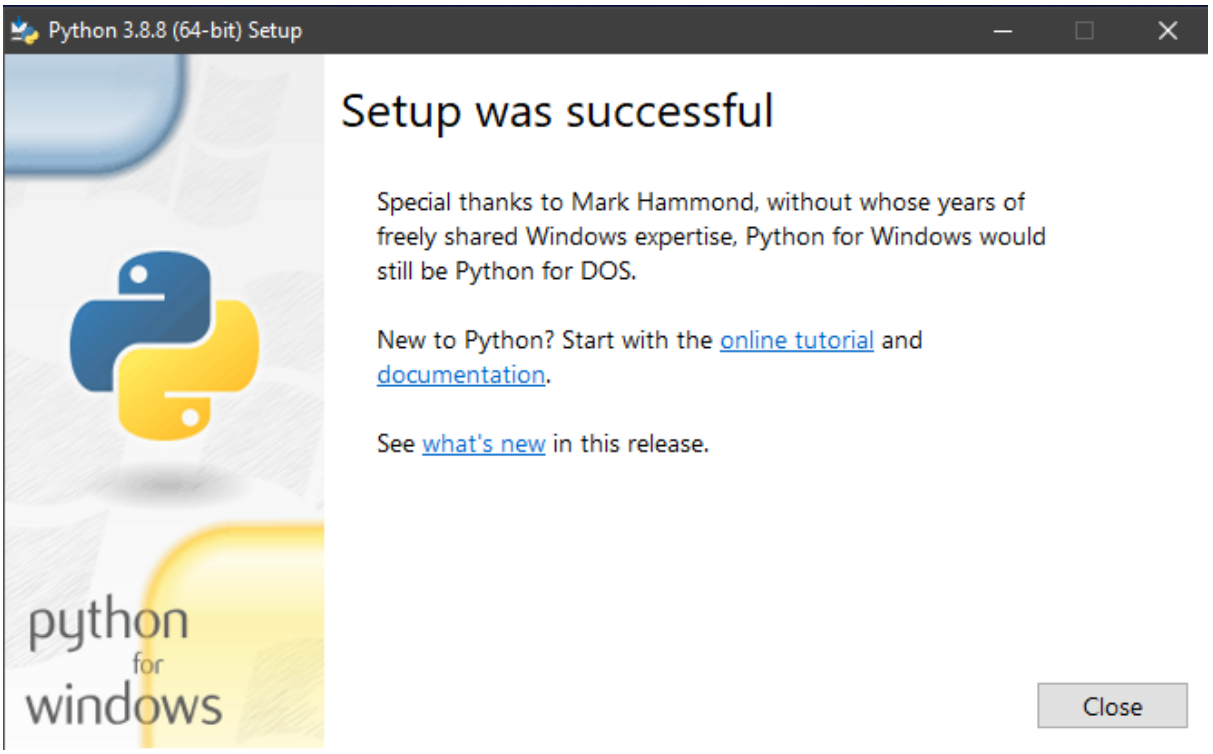
O ChatBot irá utilizar Python, RASA e SpaCy, para estes a instalação será dividida entre:

- 2.1 Windows
  - 2.1.1 Instalando o Python 3.8.8

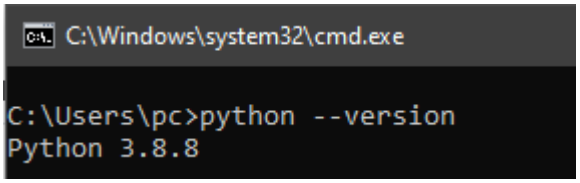
De acordo com o site oficial do Python, é possível fazer o download. Após isso, basta executar o arquivo de instalação:



A partir dessa etapa, será realizada a instalação do Python na versão 3.8.8.



A instalação foi concluída com sucesso.
Para verificar se a instalação foi realmente efetivada, basta digitar no prompt de comando:



- 2.1.2 Instalando o SpaCy 3.0.6

Logo de começo, devemos instalar a dependência:

```
> pip install -U pip setuptools wheel
```

Então, instalar o SpaCy:

```
> pip install -U spacy
```

Processo importante: para o nosso ChatBot funcionar em português - Brasil, devemos baixar o pacote do idioma. Porém, para não causar conflitos, baixaremos primeiro o pacote em inglês, posteriormente o português.

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

Inglês:

```
> python -m spacy download en_core_web_sm
```

Português:

```
> python -m spacy download pt_core_news_md
```

Também é possível analisar os pacotes instalados apenas digitando:

```
> spacy info
```

```
============================== Info about spaCy ==============================

spaCy version    3.0.6
Location         D:\PYTHON\lib\site-packages\spacy
Platform         Windows-10-10.0.19041-SP0
Python version   3.8.8
Pipelines        en_core_web_sm (3.0.0), pt_core_news_md (3.0.0)
```

- 2.1.3 Instalando o RASA 2.7

Para instalar o RASA 2.7, basta seguir a instalação direta do site ou digitar:

```
> pip3 install -U pip --user
```

```
> pip3 install rasa
```

Para verificar se a instalação houve sucesso, basta digitar:

```
> rasa --version
```

```
D:\PYTHON>rasa --version
Rasa Version      :         2.7.1
Minimum Compatible Version: 2.6.0
Rasa SDK Version  :         2.7.0
Rasa X Version    :         None
Python Version    :         3.8.8
Operating System  :         Windows-10-10.0.19041-SP0
Python Path       :         d:\python\python.exe
```

- 2.2 Linux

Para essa documentação, a versão do Linux é: Ubuntu 20.04

- 2.2.1 Instalando o Python 3.8.5

De acordo com o site oficial do Python, é possível fazer o download da versão 3.8.5. Por padrão, a maioria das distribuições Linux já provém do Python 3.8.5 em seu gerenciador de pacotes.

```
$ sudo apt-get install python
```

Logo após, devemos instalar o gerenciador de pacotes do próprio Python, o 'pip'.

```
$ sudo apt-get install python3-pip
```

Então, podemos verificar a versão do Python e se está tudo funcionando apenas digitando:

```
joao@alpha:~$ python3 --version
Python 3.8.5
```

- 2.2.2 Instalando o SpaCy 3.0.6

Logo de começo, devemos instalar a dependência:

```
$ pip3 install -U pip setuptools wheel
```

Então, instalar o SpaCy:

```
$ pip3 install -U spacy
```

Processo importante: para o nosso ChatBot funcionar em português - Brasil, devemos baixar o pacote do idioma. Porém, para não causar conflitos, baixaremos primeiro o pacote em inglês, posteriormente o português.

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

Inglês:

```
$ python3 -m spacy download en_core_web_sm
```

Português:

```
$ python3 -m spacy download pt_core_news_md
```

Também é possível analisar os pacotes instalados apenas digitando:

```
$ spacy info
```

```
============================== Info about spaCy ==============================

spaCy version    3.0.6
Location         /home/joao/.local/lib/python3.8/site-packages/spacy
Platform         Linux-5.8.0-59-generic-x86_64-with-glibc2.29
Python version   3.8.5
Pipelines        en_core_web_sm (3.0.0), pt_core_news_sm (3.0.0), pt_core_news_md (3.0.0), en_core_web_md (3.0.0)
```

- 2.2.3 Instalando o RASA 2.7

O RASA conta com uma instalação rápida e simples:

```
$ pip3 install -U pip
```

```
$ pip3 install rasa
```

Também é possível analisar a versão do RASA apenas digitando:

```
$ rasa --version
```

```
joao@alpha:~$ rasa --version
Rasa Version      :         2.7.0
Minimum Compatible Version: 2.6.0
Rasa SDK Version  :         2.7.0
Rasa X Version    :         None
Python Version    :         3.8.5
Operating System  :         Linux-5.8.0-59-generic-x86_64-with-glibc2.29
Python Path       :         /usr/bin/python3
```

3. Implementando o sistema na máquina

A implementação do Chatbot é feita a partir do download do projeto no Github, pelo link na seção Code e Download ZIP. Assim que efetuado o download, extrair o projeto para uma pasta e abri-la com o editor de código de sua preferência. Após, executar o terminal no projeto e digitar os comandos de acordo com a necessidade.

Comando para inicializar o RASA. Este é o primeiro comando a ser executado:

```
> rasa init
```

Comando para efetuar o treinamento do bot. Necessário caso exista alterações no comportamento (NLU, Domain, Rules, Stories):

```
> rasa train
```

Comando para executar as actions, referentes ao arquivo python *actions.py*:

```
> rasa run actions
```

Comando para o treinamento do bot manualmente, guiando-o. Ao salvar, as Stories são criadas automaticamente:

```
> rasa interactive
```

Ao iniciar a conversa pelo interactive, o bot exibirá a classificação feita por ele, caso esteja correta sinalize e salve.

Comando para validar os arquivos YAML na pasta *data* do projeto. São eles o NLU, Rules e Stories:

```
> rasa data validate
```

Comando para iniciar a conversa com o bot pelo terminal:

```
> rasa shell
```

Comando para iniciar a conversa com o bot pelo Webchat. Assim que o servidor do RASA rodar, abrir o *index.html* na pasta *public*:

```
> rasa run -m models --enable api --cors "*"
```

4. Explicando o projeto

4.1 FAQ

Um assistente faq Rasa leva o bot a lidar com perguntas frequentes, para um bate papo com o usuário de forma mais rápida. Após a inicialização do projeto, excluímos as intents, histórias e configurações desnecessárias do domínio. Agora, iniciaremos a criação das intenções para as perguntas, queremos fazer perguntas ao chatbot Rasa FAQ sobre a algo, mas para isso temos que inserir dados da NLU em data / nlu.md.

Response Selector

Com o Response Selector o Rasa possui uma funcionalidade que simplifica o manuseio de conversa fiada e perguntas frequentes., para isso usamos essa funcionalidade abaixo. Para este propósito, as questões nos dados NLU devem ser especialmente marcadas, ou seja, as questões devem seguir o padrão # intent: faq / ask_ <nome>.

Mostramos a pergunta sobre algo abaixo como exemplo.

Para usar o Response Selector, precisamos criar pelo menos dois intents.

 # intent: faq/ask_nome

- o que é?
- pra que serve?
- onde fica?
- por que?

Depois de criar as intents, devemos preparar as respostas, ou seja, as reações do bot à intenção do usuário. No entanto, não fazemos isso por meio de respostas dentro do domínio. Para evitar que o arquivo domain.yml se torne muito confuso e sobrecarregado com configurações, adicionamos um novo responses.md à pasta de dados.

Neste arquivo são geradas as respostas às questões previamente definidas. As respostas são criadas de forma semelhante às histórias, mas com apenas uma troca entre o usuário e o chatbot.

As vantagens do Response Selector agora são evidentes na criação da história. Não é necessário criar várias histórias, que tratem de cada questão individual. Nas histórias em data / stories.md, todas as perguntas frequentes são tratadas da mesma maneira e não há distinção entre elas. Isso pode ser uma grande vantagem, especialmente para chatbots com muitas perguntas frequentes diferentes.

Em última análise, precisamos adicionar o intent faq e a ação para responder às questões do domínio (arquivo: domain.yml).

Consequentemente, podemos treinar o chatbot Rasa FAQ com o comando rasa train e, em seguida, testá-lo via shell.

4.2 NLU

NLU (Natural Language Understanding) extrai as informações estruturadas das mensagens do usuário ou seja qual seria a intenção do usuário. Isso inclui a intenção do usuário e quaisquer entidades que sua mensagem contenha. Pode adicionar também informações como expressão regulares e tabela de pesquisa.

Exemplo desse treinamento consiste em expressões do usuário categorizado por intenção. Para facilitar o uso de seus intents, dê a eles nomes relacionados ao que o usuário deseja realizar com esse intent, mantenha-os em letras minúsculas e evite espaços e caracteres especiais.

4.3 RULES

Rules é um tipo de treinamento de dados que o assistente RASA usa para dialogar com os modelos. As Rules descrevem pequenas partes da conversação, que deverá sempre ser seguido, criando de fato uma regra de caminho.

4.4 STORIES

As Stories são um conjunto de Intents e Actions que juntas formam um fluxo de conversa, por exemplo, uma Story é composta de uma mensagem do usuário que vai ser uma palavra ou frase qual o Bot vai identificar e atribuir como parte de uma Intent feita no NLU, de acordo com essa Intent, o Bot vai retornar uma resposta denominada Action, a Action que ele vai retornar é específica para a entrada do usuário e o Bot terá conhecimento disso de acordo com as Stories.

4.5 DOMAIN

Domain é o arquivo YAML responsável por especificar as Intents, Entities, Slots, Forms e Actions do projeto RASA. Ele é o elemento essencial para a conjuntura do Bot, pois é ele que define a configuração principal e todos os processos que serão realizados. Sem ele, o Bot seria incapaz de conectar as Rules, Stories e as Actions.

4.6 ACTIONS

A cada mensagem do usuário, o modelo irá prever uma ação que o chatbot deverá executar.

Existem vários tipos de ações que poderão ser usadas no desenvolvimento de um chatbot em RASA, sendo elas: resposta, ações personalizadas e formulários.

A resposta, é a ação que irá ser utilizada com mais frequência, e poderá enviar textos, imagens e até botões para o usuário. Como o próprio nome diz, a ação de resposta é a que irá retornar uma resposta para o usuário.

Uma ação personalizada é um arquivo .py, e será desenvolvida utilizando a linguagem Python. As ações personalizadas podem ser usadas para fazer uma chamada de API, consultar um banco de dados ou qualquer outra ação que necessite da linguagem Python para desenvolver. Para que o bot possa executá-las, é necessário rodar o comando *rasa run actions*.

Os formulários são ações personalizadas, onde você pode desenvolver um design de conversação em que o assistente irá oferecer um conjunto específico de informações.

4.7 Desenvolvimento do Projeto

4.7.1 Criando um novo projeto RASA

Para iniciar um novo projeto na RASA, é necessário criar uma pasta onde vai ser feito o desenvolvimento do projeto e executar um comando chamado *rasa init* dentro dessa pasta. Esse comando vai criar todas as pastas necessárias para o desenvolvimento do projeto.

```
> rasa init
```

Enquanto o comando está sendo executado, em um determinado momento vai pedir permissão para executar o treinamento do modelo, ou negar esse treinamento. Caso você permita o treinamento do modelo durante a criação do projeto, o bot vai ser executado em inglês e será possível conversar com ele. Caso o treinamento do bot seja negado, você poderá já iniciar o desenvolvimento do projeto no mesmo instante.

4.7.2 Configurações necessárias

No arquivo config.yml, é necessário fazer algumas alterações antes de iniciar o desenvolvimento, e uma das configurações a serem feitas, é a mudança do idioma do bot.

O RASA por padrão, vem com o idioma em inglês, porém para esse projeto, é necessário usar o idioma em português. Para isso, vamos utilizar a biblioteca SpaCy, e dentro dela, baixar e utilizar o vocabulário em português.

Após efetuar a instalação do SpaCy, e o download do maior vocabulário de português, vamos fazer algumas outras configurações necessárias para o desenvolvimento do projeto.

4.7.3 Desenvolvendo as perguntas e respostas

O desenvolvimento da conversa entre o bot e o usuário em relação às perguntas do vestibular, tem a utilização do FAQ. Método que alinha o *input* do usuário diretamente com as respostas do bot, estas, por sua vez são definidas a parte em uma pasta nomeada *responses,* dentro da pasta *data*, contendo o arquivo *responses.yml* com *utters* relacionadas as *intents* possíveis do usuário e os *examples*. Para o funcionamento é necessário incrementar o arquivo *config.yml* com o código abaixo:

```yaml
- name: ResponseSelector
  epochs: 100
  retrieval_intent: faq
  model_confidence: linear_norm
  constrain_similarities: true
```

e adicionar “faq/” nas intents como prefixo e nas *utters* em *responses*. Nas *rules* adicionar o código abaixo:

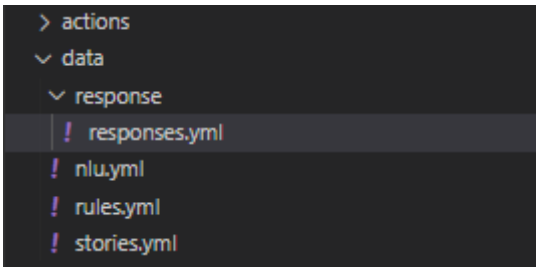
```
- rule: faq
  steps:
  - intent: faq
  - action: utter_faq
```

5. Manutenção do Chatbot

A manutenção do chatbot deve ser feita manualmente, uma vez que as informações vão se desatualizando, para isso, deve se seguir os passos abaixo.

Atualizando as respostas do Bot

Para atualizar as respostas apresentadas pelo Bot, acesse o arquivo *responses.yml* dentro da pasta *response*:



para ter acesso às *utters* que são actions que o Bot aciona sempre que precisar responder a uma intent dada pelo usuário:

```
utter_faq/ask_perdi_meu_cpf:
- text: |-
    Arquivos com os seguintes documentos que comprovem o número do CPF:
    - CPF documento exclusivo;
    - RG, desde que contenha o número do CPF;
    - Carteira Nacional de Habilitação CNH.
```

Como exemplo usaremos a *utter_faq/ask_perdi_meu_cpf* que é acionada sempre que o Bot receber do usuário uma intent relacionada a perda do CPF.

A sintaxe deve ser seguida, pois parte do padrão do RASA, por exemplo:

utter

- nome reservado pelo sistema para referir a uma resposta

faq/ask

- padrão RASA para respostas via FAQ, manter isto é uma boa prática para um Bot FAQ e vai evitar erros futuros.

text

- Essa parte indica que será retornado para o usuário como resposta um conjunto de caracteres em formato de texto.

|-

- O pipeline seguido de um hífen aplicará uma formatação adequada ao texto, como quebra de linhas automática, etc.

Abaixo contém o texto de resposta do Bot, qualquer alteração deverá ser feita exatamente nessa parte como por exemplo, troca de palavras, frases, adicionar ou remover linhas existentes e é importante se atentar sempre com a estrutura da sintaxe, manter a indentação evitará problemas maiores.

6. Integração no site

7. WebChat

Para que seja possível hospedar e conversar com o bot no website, precisa ser feito um *webchat*.

O *webchat* é um pequeno campo desenvolvido em HTML5, CSS3 e JavaScript, que o usuário irá clicar e abrir o chat para iniciar uma conversa com o bot.

Para integrar o bot com o *webchat,* é necessário descomentar o campo *socketio* no arquivo credential.yml, e chamar os eventos: user_uttered para o campo user_message_evt, e bot_uttered para o campo bot_message_evt. No campo ssesion_persistence, deve colocar o valor true.

8. RASA Lit

O RASA Lit é uma ferramenta para entender graficamente as NLU's, auxilia no processo de manipulação das frases e possíveis caminhos de escolha do bot para sua frase e contexto.

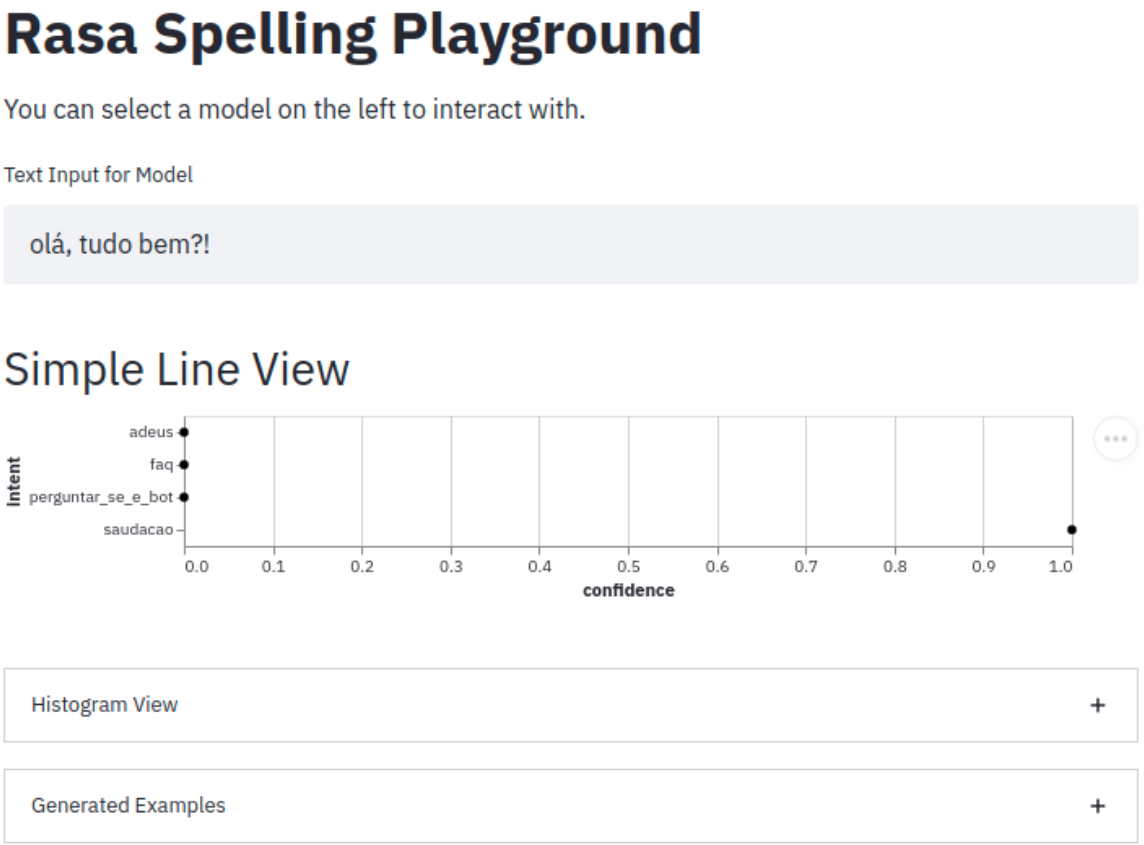
Instalação

```
> python -m pip install git+https://github.com/RasaHQ/rasalit
```

Para mais detalhes de uso, acesse o [GitHub do RASA Lit](https://github.com/RasaHQ/rasalit).

Spelling

O Spelling é um dos comandos para a utilização do RASA Lit, com ele é possível predizer a confiança de uma frase ou palavra no contexto do seu chatbot.



Live-NLU

O Live-NLU é uma ferramenta no estilo da Spelling com mais interatividade no contexto da frase. Podemos causar uma confusão na escolha da entidade apenas passando a palavra “FATEC” no contexto. Essa confusão é por causa que a palavra selecionada é muito repetida dentro do contexto das FAQ’s.

Rasa NLU Model Playground

You can select a model on the left to interact with.

Text Input for Model

olá bot da FATEC, tudo bem?

Tokens and Entities

olá bot da fatec , tudo bem ?

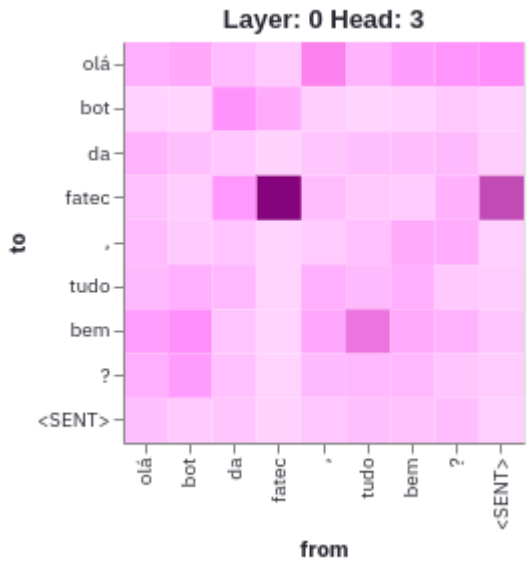
Intents

Full Model Specification +

View Diet Attention Charts. +

Com a propriedade do Live-NLU é possível analisar “View Diet Attention Charts”, uma ferramenta específica para verificar o percentual de atenção da sua frase dentro de uma camada de DIET inerente.

Última camada de treino:



Existem 5 possíveis comandos, cada um te retornará um melhor compreendimento da ferramenta:

```
> python -m rasalit nlu-cluster --port 8501
```

```
> python -m rasalit overview --folder gridresults --port 8501
```

```
> python -m rasalit spelling --port 8501
```

```
> python -m rasalit live-nlu --port 8501
```

```
> python -m rasalit diet-explorer --port 8501
```

Referências

https://www.take.net/blog/tecnologia/nlp-processamento-linguagem-natural/
https://en.everybodywiki.com/Rasa_NLU
https://www.geeksforgeeks.org/chatbots-using-python-and-rasa/

BORGES, Luiz Eduardo. **PYTHON PARA DESENVOLVEDORES**. São Paulo: Novatec, Outubro/2014.

CRUZ, Felipe. **Python**: Escreva seus primeiros programas. Vila Mariana - São Paulo: Casa do Código, 2018.