ChatBot - Versão 1.0

Campusito

João Victor de Souza 15/0132425
Patrick Beal 15/0143672

Sumário:

Introdução

O projeto do ChatBot consiste em um assistente virtual para a Faculdade de Comunicação, hospedado na plataforma Campus Online APP v3 [5], com o objetivo de suprir docentes e discentes em períodos que o jornal estiver fora do ar.

Nesta primeira versão, foi desenvolvida a estrutura do bot inteligente, que, de acordo com um diálogo previamente fornecido pela FAC, responde às perguntas de forma esperada. Também foi implementada com sucesso a integração desta plataforma com o sistema do Campus Online.

A estrutura visual do chat também foi padronizada de acordo com a plataforma em que ele está hospedado.

Estrutura do projeto

O Bot foi desenvolvido em Python utilizando a ferramenta Rasa [1]. Nesta primeira versão, a sua estrutura é a seguinte:

- CHATBOT
    - Bot
        - Data
            - nlu.md
            - stories.md
        - Models
        - actions.py
        - config.yml
        - credentials.yml
        - domain.yml
        - endpoints.yml
        - makefile
    - Modules
        - Webchat
            - assets
            - index.html
    - requirements.txt

O projeto está hospedado no GitHub [4].

Funcionamento

Inicialmente, deve-se compreender que, para conseguir rodar o bot localmente, existem três servidores para o funcionamento na plataforma do APP v3:

- Servidor do APP V3;
- Servidor global do ChatBot;
- Servidor de actions do ChatBot.

As orientações necessárias para configurar localmente os servidores podem ser encontradas nos links [4] (ChatBot) e [5] (APP v3). Ou seja, os três servidores devem estar rodando simultaneamente, então quando for configurar o APP v3, mude para a branch “chatbot”, instale os *requirements.txt* dentro de uma *venv* e rode o comando: **python manage.py runserver**.

Observe que no diretório /campus_app/templates/ da branch “chatbot” do APP v3, temos o arquivo home.html. É nesse arquivo que encontra-se a *tag script* que realiza a integração com o backend do Bot (note que a *tag* é a mesma que se encontra no repositório do ChatBot no diretório /modules/webchat/index.html, apenas mudando alguns assets).

O Bot deste projeto funciona de acordo com a estrutura definida na seção anterior. Tomando em consideração que ele é baseado na linguagem MarkDown *.md*, segue uma breve explicação de cada componente:

- O diretório **/Bot** é o responsável pela maior parte do funcionamento da ferramenta.
    - O diretório **/Data** contém 2 arquivos:
        - **nlu.md** (*Natural Language Understanding*): contém os *intents* que o usuário pode escrever enquanto interagir com o bot. Observa-se que eles são divididos em sessões, com o intuito de facilitar a estruturação das **stories**.
        - **stories.md**: Contém os possíveis *paths* que a conversa pode seguir, utilizando a seguinte estrutura básica:
            - *INTENT*
                - *UTTER*

          O *intent* foi mapeado no arquivo **nlu.md** e o *utter* será descrito a seguir no arquivo **domain.yml**. Ao perceber que o usuário ativou uma *intent* mapeada, o bot irá verificar se existe uma *story*

mapeada com essa *intent.* Caso exista, a *utter* mapeada para essa *intent* será ativada.

- **domain.yml**: Este arquivo contém todas as intents descritas no **nlu.md**, bem como os templates, os quais correspondem às possíveis respostas do bot. Os templates contém as *utters* que são descritas no **stories.md** com o seu respectivo texto. Além disso, contém as *entities*, as quais são sinônimos ou categorias de um determinado assunto. Ainda, o arquivo contém os *slots* da aplicação, os quais correspondem às "variáveis" que podem ser preenchidas de acordo com a interação com o usuário. No momento, a única *slot* é a de *feedback,* o qual recupera alguma informação que o usuário queira passar, seja de sugestão, problema ou reclamação. Por fim, existem os *forms,* que são utilizados para preencher os *slots,* e as *actions*, as quais correspondem às *utters.*
- **credentials.yml:** Aqui encontra-se a configuração inicial necessária para conseguir rodar o bot no telegram. As orientações de como fazer isso podem ser encontradas em [4].
- **endpoints.yml:** Local onde estará disponível a interação resultante do bot.
- **actions.py:** Local onde os *forms* e as possíveis *custom actions* serão tratadas.
- **Makefile:** Comandos úteis para utilizar o bot.
- O diretório **/Models** contém os arquivos resultados do treinamento. A cada mudança em algum dos arquivos acima, será necessário treinar o bot novamente com o comando: **make train**.

Toda vez que o usuário interagir com o bot, ocorre um processamento da sua mensagem em que sua resposta será atribuída de acordo com um grau de confiança alto de 90%, de acordo com o modelo de treinamento resultante do bot.

Integrações com Banco de Dados, Facebook e Telegram

Além da integração com a plataforma APP v3 e Telegram, o bot também pode ser integrado com o Facebook e outros sites da FAC, de acordo com a necessidade da faculdade.

De acordo com a estrutura da versão 1, o chatbot armazena entradas de usuários de forma estática, apenas as salvando em seu estado atual. Uma necessidade importante para o funcionamento eficiente do ChatBot seria uma integração com banco de dados utilizado pelo próprio APP, facilitando a comunicação

front - backend. Para realizar essa integração, deverá mudar o *endpoint* do ChatBot, não sendo mais local, e sim para o servidor em que ficará hospedado. Ainda, no arquivo, /campus_app/templates/home.html da branch “chatbot” do APP v3, deve-se alterar o socketURL, para efetuar a devida conexão entre o backend (ChatBot) com o front (APP v3).

Funcionalidades futuras

Dentre as funcionalidades e possibilidades futuras, temos:

- **Criação e integração com banco de dados**
  - Esta funcionalidade possibilitará a Faculdade de Comunicação analisar as sugestões, questionamentos, correções e reclamações do público alvo. Essa funcionalidade pode ser feita utilizando as *custom actions e forms* do Rasa.

- **Expansão do diálogo da plataforma**
  - O diálogo pré-definido pode ser mudado e expandido de forma simples, de acordo com as necessidades da Faculdade de Comunicação.

- **Ações customizadas de acordo com o *input* do usuário**
  - Utilizando as *custom actions e forms* da plataforma Rasa, pode-se realizar ações específicas como realizar consultas no banco de dados, trazendo a respectiva informação baseada na entrada do usuário.

Referências

Aqui temos alguns links úteis que podem ajudar os próximos desenvolvedores da plataforma.

[1]https://rasa.com/docs/rasa/
[2]https://rasa.com/docs/rasa/core/actions/
[3]https://blog.rasa.com/building-contextual-assistants-with-rasa-formaction/?_ga=2.137562038.780948946.1575480381-901910803.1570622388
[4]https://github.com/joao96/ChatBot
[5]https://github.com/VSSantana/APP-Campus-Online-v3.0