TÓPICOS DE ENGENHARIA INFORMÁTICA
CHATBOT - NATURAL LANGUAGE PROCESSING *

Luís Carlos Soares das Pazes
Aluno
Instituto Politécnico de Beja - Escola Superior de Tecnologia e Gestão
Beja
15974@ipbeja.pt

Gonçalo José Cunha Fontes
Professor
Instituto Politécnico de Beja - Escola Superior de Tecnologia e Gestão
Beja
goncalo.fontes@ipbeja.pt

***K*eywords** Chatbot · NLP · React · JSON-Server · wit.ai · pusher

Introdução

Neste relatório irá ser explorando mais a fundo o que é um chatbot, o que foi utilizado para "melhorar" uma aplicação do mesmo, e como foi utilizado NLP para extrair a intenção de um utilizador de modo a obter um flow de conversa melhor do que no projecto anterior. As tecnologias que serão utilizadas são as seguintes:

- VS Code v1.58.2
- npm v6.14.13
- node.js v14.17.2
- preact v10.5.14
- express v4.17.1
- json-server v0.16.3
- wit.ai[2]
- pusher[4]

O sistema operativo utilizado foi o Windows 10.

Este relatório divide-se em duas partes. Primeiramente irá ser explicado conceitos base, necessários para a compreensão do código, e as decisões tomadas. Na segunda parte irá ser explicado o código e as decisões tomadas no mesmo.

Foi pedido pelo docente que o tema para o chatbot fosse de um serviço de transporte de passageiro (e.g. Uber ou Cabify).

---

*Luís das Pazes. Tópicos de Engenharia Informática - Chatbot Natural Language Processing. 4 Páginas.....

Preact

"O Preact em si não se destina a ser uma reimplementação do React. Existem diferenças. Muitas dessas diferenças são triviais ou podem ser completamente removidas usando preact/compat, que é uma camada fina sobre Preact que tenta atingir 100% de compatibilidade com React. O motivo pelo qual Preact não tenta incluir todos os recursos do React é para permanecer pequeno e focado - caso contrário, faria mais sentido simplesmente enviar otimizações para o projeto React, que já é uma base de código muito complexa e bem arquitetada."[1]

Objectivo

"O objetivo deste projeto é criar um chatbot de interface dinâmica na tecnologia React, com ligação ao motor do Wit.ai, potenciando-lhe capacidade de interação e conversão. O chatbot deverá obrigatoriamente ter a temática de um serviço de transporte de passageiro (e.g. Uber ou Cabify) onde se deverão enquadrar as interações com o bot. Por exemplo, deverá ser possível fazer pedidos de transporte, informar sobre os preços, tipos de viaturas e distâncias a percorrer, entre outras. O bot deve tentar dar resposta a todos os pedidos relacionados com a temática, devendo a "conversa" aproximar-se o mais possível, à conversa que o utilizador teria com um ser humano. A interface deverá ter no seu Frontend, um Header e um Footer enquadrado na temática, e deverá ter um input de texto para potenciar as interações do utilizador. Quer as interações do utilizador, quer as repostas do bot devem ficar visíveis na interface após a submissão por parte do utilizador, e devem poder ser removidas individualmente. O chatbot deverá ser multiutilizador e deverá ser possível criar uma nova conversa sem prejuízo da anterior. Por outro lado, o bot deve poder sugerir perguntas ou respostas sempre que estas tiverem enquadramento."

Relativamente ao projeto anterior, pode verificar-se que foi removido a escolha do tema. Houve também a adição de um requisito, que são as múltiplas salas, ou instâncias, onde se pode haver várias conversas paralelas. A maior diferença é o flow da conversa, e como esta se processa.

Com a utilização de NLP e wit.ai, assim como o pusher, dá para ter conversas melhores com o bot, pois em vez de ele ir buscar a string da mensagem do user, comparar com algumas que já existem, e dar a resposta em prol disso, o bot irá procurar a intenção do utilizador, e as palavras chave que se encontram na frase.

Com esta pesquisa, o bot permite reconhecer o que o utilizador quer e envia a resposta adequada para tal. Com o método do primeiro trabalho, na maioria das situações, o que se verificava era que o bot devolvia a resposta genérica que não sabia ainda como responder.

O que se verifica agora é que o bot irá tentar perceber o que o utilizador quer, ou pelo menos adivinhar, com uma percentagem de confiança.

NLP e Wit.ai

O NLP é uma área da inteligência artificial que auxilia os computadores a entender, interpretar e manipular a linguagem humana e resulta de várias áreas cientificas, incluindo a ciência de computação e linguística computacional.

Ainda que não sendo uma ciência nova, a mesma tem tido grandes avanços nos últimos anos, devido ao interesse cada vez maior de melhorar a comunicação entre os computadores e os humanos.

Há 70 anos, os programadores usavam cartões furados para se comunicar com os primeiros computadores. Hoje, é possível dizer "Alexa, eu gosto desta música", e um equipamento responder "Ok, classificação salva" e adaptar o seu algoritmo para tocar essa música, e outras parecidas, na próxima vez que ouvir musica.[3]

Wit.ai é uma interface de NLP para aplicações, capaz de transformar frases em dados estruturados. Primeiramente, vai ser introduzida a frase pretendida, como exemplo será "Hello". De seguida cria-se uma intenção, se não houver uma já pré-definida. De seguida, serão selecionadas frases ou palavras chaves, para se adicionar essa informação como entity, para puderem ser usadas. Opcionalmente poderão ser também usados traits, que são como entities, mas com valores fixos, como os cumprimentos "Hello" e afins. Por final, carrega-se no botão para treinar o algoritmo de linguagem.

As figuras seguintes irão demonstrar visualmente os passos:

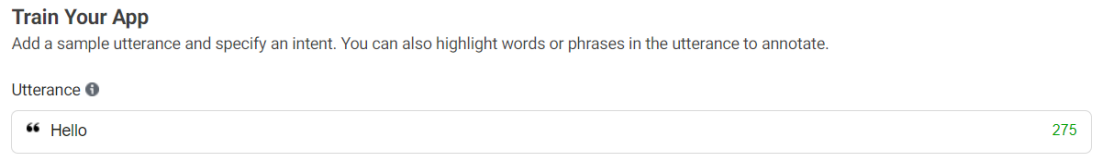

Figure 1: Passo:1

Figure 2: Passo:2

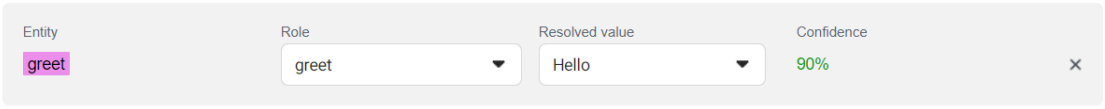

Figure 3: Passo:3

Figure 4: Passo:4

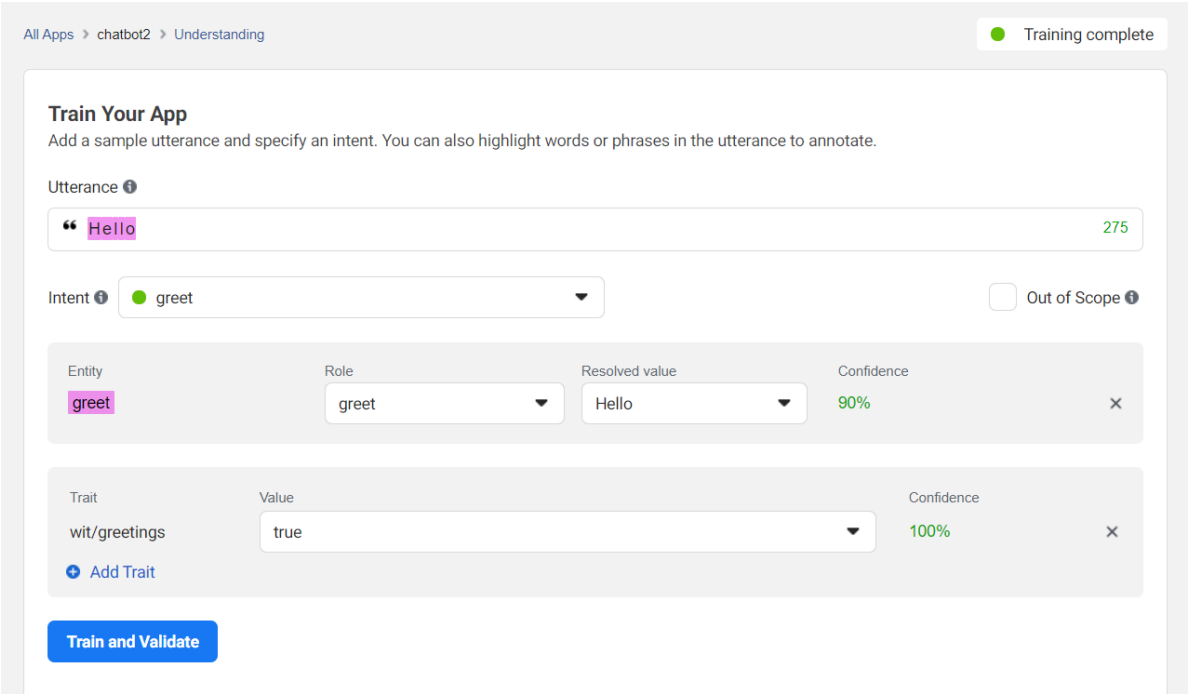

Figure 5: Passo:5

Pusher

O pusher é um API, que vai servir para empurrar as respostas do bot do backend para o frontend, através de triggers e subscriptions.

Aplicação

Na realização deste projeto foram criadas duas pastas, components, onde irão ser guardados os componentes e Images onde serão guardadas as imagens utilizadas pelo site. Foram criados os ficheiros dos componentes e também usado um ficheiro de exemplo css que foi modificado para corresponder ao tema escolhido e foi também criado o ficheiro db.json, que vai servir como uma base de dados para complementar o backend.

Como componentes, foi criado um header, que contém o título e o logo, um footer, que contém um texto a indicar o criador. Um ficheiro chamado index.js, fora da pasta dos componentes, terá a estrutura de mensagens da aplicação e as próprias mensagens. Para além destes ficheiros, foi criado também um ficheiro server.js que terá as operações todas de backend.

No ficheiro index.js, existem várias funções, que são vitais à operação do site. Neste caso, irá ser utilizado uma feature do Preact, que é o state, e é usado para carregar as mensagens escritas para o ecrã, o input do utilizador, o número da sala de chat e as salas existentes. A função handleDelete usa o método DELETE para apagar as mensagens no JSON, assim como filtra o array das mensagens para as apagar no UI também. A função handleDeleteBot apenas difere que é usada para as salas de chat, em vez das mensagens.

A função handleSubmit é a principal, que irá inicialmente desativar uma flag que verifica se o bot respondeu, isto será importante noutro código explicado mais tarde. Irá após o processo gravar a mensagem do utilizador na db, depois atualiza o array para conter também a mensagem no UI e ativa uma flag para saber que o utilizador submeteu a mensagem, que será também importante depois. Após ter gravado, irá utilizar uma route, para enviar a mensagem do utilizador para o backend que irá ser discutido mais tarde no documento.

A função handleChange atualiza o estado sempre que o utilizador escreve uma palavra.

Temos também o construtor, onde irão ser inicados o estado e as funções, assim como carregadas as mensagens, a partir do histórico gravado.

As funções change bots, handleDeleteBot e handleAddBot servem respetivamente para mudar de sala de chat, apagar a respetiva sala onde o butão for chamado e adicionar uma sala através do botão +. Apenas são permitidas 5 salas simultâneas para termos de performance. As operações de adicionar e remover salas também são feitas a nível da base de dados, para gravar as conversas quando se saír do site.

A função updateConvo serve para atualizar as mensagens quando se muda de sala.

A função render irá renderizar a página html, de acordo com os flows de react.

E por fim, no ficheiro index.js, a função componentDidMount é uma função especial que irá escutar quando o state é atualizado, e nesta função, é onde serão captadas as mensagens do bot, no canal do pusher, de acordo com as condições estabelecidas acima, que o utilizador submeteu e o bot não respondeu ainda. As mensagens do bot em si são gravadas da mesma forma que as do user.

Para o backend, temos o ficheiro server.js, que irá associar com os APIs através de chaves e afins, e irá escutar na route, pelo input do utilizador, e a resposta do wit.ai, através da função app.post, que contém as subfunções firstEntityValue e firstEntityValueLocation que verificam qual entity o bot devolveu, pondendo não haver nenhuma, e a função handleMessage, que irá ver quais as entities usadas, e responder através de condições. Por fim temos a função client que é o servidor receber a mensagem do user e responder de acordo com o estabelecido.

Conclusão

Para iniciar a aplicação, é necessário executar em 3 terminais separados, os comandos:

- npm run dev
- node server.js
- json-server –watch db.json –port 9000

References

[1] m. et al. Differences to react – preact.

[2] Facebook. Wit.ai, 2019.

[3] G. Fontes. Pln - processamento de linguagem natural, 2021.

[4] P. Limited and MessageBird. Pusher | leader in realtime technologies, 2020.