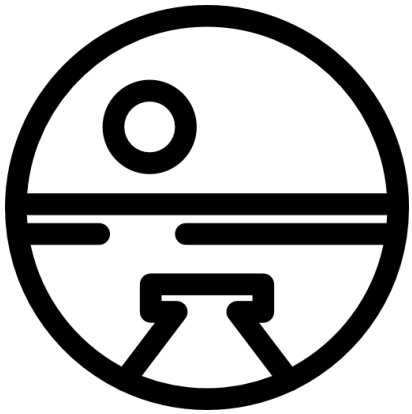

SalinasBOT

Agente Conversacional de apoio ao turista da região de Aveiro

Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática

Sistemas Inteligentes

João Alegria | 68661
Hugo Pintor | 76610
Higino Caires | 89094
Carlos Manuel | 88702

ÍNDICE

PROPÓSITO DO TRABALHO

Proposto na Unidade Curricular de Sistemas Inteligentes do Mestrado de Engenharia Informática, SalinasBOT consiste num agente conversacional de apoio ao turista da região de Aveiro.

O seu principal intuito consiste no desenvolvimento de um *chatbot* e permitir a troca de mensagens entre o utilizador e o mesmo, respondendo assim o mais assertivamente às questões lançadas pelo utilizador.

Assim, a missão deste projeto é dar resposta às principais esferas turísticas da região de Aveiro, tais como: Gastronomia, Hotelaria, Espaços de Lazer e Produtos Regionais. Também é interessante ver presente serviços complementares tais como, previsões meteorológicas, farmácias de serviço da região e serviços de partilha de transporte.

No desenrolar do presente relatório, será apresentado e explicado os primeiros detalhes idealizados para a implementação do *bot*.

CHATBOT

O termo *chatbot* têm tido uma evolução significativa nos últimos anos devido à adoção destes por parte de grandes empresas da área da tecnologia. Apple, Facebook e Google são alguns dos exemplos possíveis a apontar, onde foram adotadas novas ferramentas e *frameworks* de modo a construir assistentes inteligentes e, proporcionar assim, ao utilizador uma rápida e diferente interação com os seus produtos.

É possível definir este como um programa ou serviço de resposta automática baseado em regras, no qual pode ou não conter inteligência artificial, onde o utilizador poderá interagir e estabelecer um diálogo através da troca de mensagens de texto ou de voz.

O seu principal objetivo é responder a perguntas colocadas pelo utilizador e responder a estas o mais corretamente possível, procurando simular o comportamento de um ser humano durante a interação. [1]

É possível apontar dois tipos de *chatbots*. Os que são baseados em regras e os que são baseados em inteligência artificial. Os primeiros, unicamente baseados em regras, apresentam algumas limitações pois dependem da introdução de palavras-chave ou de comandos específicas para procederem com o seu fluxo. Desta forma, na eventualidade do *bot* não compreender a mensagem introduzida pelo utilizador, poderá conduzir a uma resposta errada ou à ausência da mesma. No entanto, um *bot* que tenha presente inteligência artificial na sua solução terá uma maior capacidade de aprender e entender a linguagem natural. Desta forma, quanto maior for a sua utilização e o número de utilizadores a beneficiar da solução, a tendência é que esta melhore, tornando-se mais credível e convincente. [2]

As principais vantagens da utilização dos *chatbots* têm como objetivo a simplificação de certas interações, especialmente as que são repetitivas, tornando possível a aplicação de respostas previamente delineadas para determinadas solicitações ao agente.

Por outro lado, não são apenas as grandes empresas a adotar este serviço. A adoção de pequenos *bots* em sistemas de troca de mensagens têm vindo a ser adotadas por pequenas e médias empresas, sobretudo com o apoio das aplicações de redes sociais já existentes. Alguns destes podem ser encontrados no Facebook Messenger, Slack e Skype, reduzindo assim a necessidade de custos acrescidos com o envio de mensagens de texto – SMS – e as limitações que estas acarretam. [3]

Figura 1 - Exemplo da utilização do chatbot da iFood no Facebook Messenger [3]

Relativamente aos modelos de suporte de criação de uma resposta, Denny Britz em *Deep Learing for Chatbots* [4] conclui que um *chatbot* pode gerar respostas com base em modelos de aprendizagem ou utilizar heurísticas para selecionar respostas predefinidas de um repositório já existente.

Desta forma é possível apontar **três tipos de modelos** [5]:

- **Modelos Geradores** – tendencialmente mais difíceis de construir e de treinar. Para estes, é necessário um número considerado de exemplos para treinar o modelo de aprendizagem, de modo a obter uma qualidade de resposta desejável. No entanto, não é possível assertivamente calcular qual a resposta que este irá gerar.
- **Modelos baseados em recuperação** – teoricamente mais fáceis de construir, fornecendo uma resposta ao utilizador mais previsível. Neste é possível definir um conjunto respostas e ter um maior controlo no retorno da mesma ao utilizador, garantindo assim que não hajam respostas inadequadas, face ao proposto pelo mesmo, onde a tomada de decisão da resposta a retornar tem como base o contexto (como por exemplo, coordenadas atuais do utilizador, *username*, todas as mensagens anteriores, etc) da conversa até então.
- **Modelos com padrões baseados em heurísticas** – as heurísticas para a seleção de uma resposta podem ser concebidas de diferentes formas, desde condições *if-else* até classificadores de *machine learning*. Uma das práticas mais simples é a adoção de regras - *rules* - com padrões como condição para as regras. A linguagem AIML [6] - *Artificial Intelligence Markup Language* - é uma das que emprega a adoção de regras para escrever padrões e modelos de resposta.

ARQUITETURA

A arquitetura exibida de seguida apresenta o *workflow* e os componentes que, nesta fase inicial, o *chatbot* irá reunir.

Após um levantamento de algumas abordagens já existentes [5] [7] conclui-se que o modo de funcionamento de um *chatbot* é semelhante em todos os analisados.

Assim, a arquitetura foi conceptualizada de modo a ser fluída e escalável. De notar que, esta solução corresponde a uma primeira versão da mesma, suscetível a posteriores alterações conforme a evolução do trabalho.

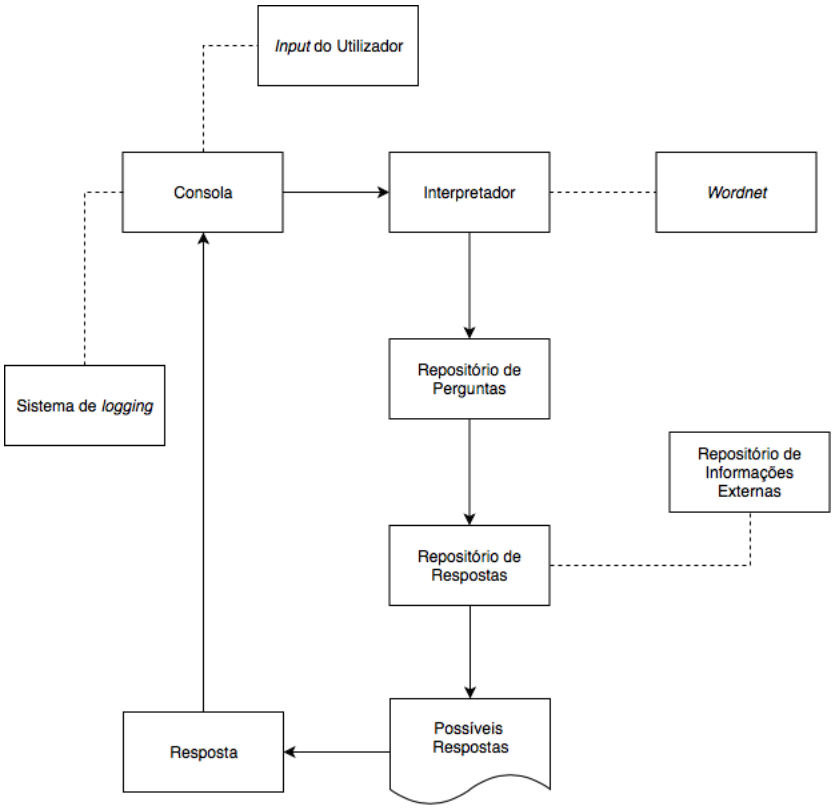

Figura 2 - Proposta de arquitetura do chatbot

Workflow

O *workflow* idealizado e apresentado anteriormente reúne as principais ações que o *chatbot* desempenhará. Uma vez que, a escalabilidade é um fator tomado em consideração, é possível adicionar novas funcionalidades no decorrer da evolução do projeto, tais como, o acesso a APIs de terceiros.

A solução iniciará com a interação entre o *chatbot* e o utilizador, onde será capturado o *input* escrito pelo mesmo e, analisado posteriormente, pelo interpretador.

Neste, e com o auxílio de ferramentas linguísticas existentes no Wordnet [8], irá tentar atribuir ao *input* do utilizador uma categoria, que estará associada a uma *tag* AIML, disponível no Repositório de Perguntas.

Após essa atribuição, o *chatbot* utilizará essa mesma categoria, mas desta vez para ir procurar uma das respostas disponíveis no Repositório de Respostas.

O *workflow* ficará concluído quando a resposta encontrada for enviada ao utilizador, repetindo-se este ciclo quando o utilizador fornecer um novo *input*.

Em paralelo estará implementado um sistema de *logging* que permitirá guardar as conversas entre o utilizador e o *chatbot*, bem como a utilização de bases de conhecimento externas (Repositório de Informações Externas) para auxiliar no enriquecimento do Repositório de Respostas.

Componentes do *Chatbot*

Dos componentes exibidos na proposta de arquitetura já apresentada, é possível dividir estes em duas categorias: componentes primários e componentes secundários.

Componentes Primários

Os componentes primários são os componentes essenciais para o funcionamento da solução, sendo que é sobre estes que incidem os principais processos contidos no *workflow* descrito anteriormente.

Assim componentes primários são:

- Consola – Ferramenta de interação entre o *chatbot* e o utilizador
- Interpretador – Módulo de interpretação dos *inputs* do utilizador e a associação a uma questão e *tag* AIML para que possa ser retornada a resposta mais assertiva.
- Repositório de Perguntas / Respostas – Sistema de base de dados do sistema. Nesta primeira iteração está estimado a implementação de *tags* definidas em AIML nas quais estão associados possíveis *skeletons* de questões colocadas pelo utilizador, bem como, possíveis respostas às mesmas.

Componentes Secundários

Relativamente aos componentes secundários estes têm como objetivo o auxílio aos componentes primários, especificados anteriormente, providenciando novas funcionalidades no *worksflow* do *chatbot*.

Assim os componentes secundários até agora idealizados são:

- Sistema de *Logging* – Módulo que permite guardar as mensagens trocadas entre o utilizador e o *bot* num ficheiro de texto.
- *Wordnet* – A utilização do *Wordnet* centrar-se-á sobretudo como suporte ao interpretador, tendo como principal função utilizar as suas valências na associação de sinónimos e de contextualização frásica, de modo a auxiliar a interpretação dos *inputs* do utilizador. Com isto, espera-se uma maior facilidade na identificação das perguntas e consecutivamente na recuperação e construção da resposta.

- Repositório de Informações Externas – Módulo encarregue de recolher informações de serviços terceiros, nomeadamente através do uso de APIs e pedidos HTTP. Será particularmente útil para recolher informações relativas aos serviços de meteorologia, farmácias e de transportes, de modo a fornecer ao utilizador informação sempre atualizada e, desta forma, aumentar o *know-how* do *bot*.

AIML

O AIML (*Artificial Intelligence Markup Language*) é um conjunto de *tags* baseadas na linguagem em XML, de forma a possibilitar a representação e criação de diálogos semelhantes à linguagem natural por meios de softwares. Por fim, este consegue simular a inteligência humana. [9]

A unidade básica da AIML consiste no tag `<category>`, onde cada categoria é formada por `<pattern>` e `<template>`, onde:

- **`<pattern>`** - Contém uma frase que o usuário pode realizar, ou seja, basicamente consiste em uma questão.
- **`<template>`** - Contém a resposta que o chatterbot irá utilizar para a resposta.

Exemplo:

```
<aiml>
  <category>
    <pattern>Olá tudo bem?</pattern>
    <template>Sim, obrigado.</template>
  </category>
</aiml>
```

O tag `<srai>` permite identificar ao chatterbot que duas questões são similares, ou seja, apesar de formuladas de um modo diferente, tem um mesmo significado.

Exemplo:

```
<aiml>
  <category>
    <pattern>Olá</pattern>
    <template>Oi, como vai você?</template>
  </category>
  <category>
    <pattern>Oi</pattern>
    <template><srai>Olá</srai></template>
  </category>
</aiml>
```

O caractere para substituição (*) pode ser usado quando se tem um grupo de instruções semelhantes que precisarão ser definidos a partir de um mesmo conjunto de questionamento.

Exemplo – *ChatBot* explicar a finalidade dos móveis de uma casa:

```
<aiml>
  <category>
    <pattern>O que é uma *</pattern>
    <template><srai>DEFINE <star/></srai></template>
  </category>
  <category>
    <pattern>Explique o que é uma *</pattern>
    <template><srai>O que é uma <star/></srai></template>
  </category>
  <category>
    <pattern>DEFINE Mesa</pattern>
    <template>Serve para servir refeições.</template>
  </category>
  <category>
    <pattern>DEFINE Cadeira</pattern>
    <template>Móvel para se sentar</template>
  </category>
</aiml>
```

Uma determinada questão pode ter várias respostas possíveis, desta forma, a utilização do tag `<random>` permite que o chatterbot escolha uma das respostas de modo aleatório.

Exemplo – *ChatBot* explicar a finalidade dos móveis de uma casa:

```
<aiml>
  <category>
    <pattern>* TUDO BEM</pattern>
    <template>
      <random>
        <li>Tudo em ordem.</li>
        <li>Muito bem, obrigado.</li>
        <li>Eu estou bem.</li>
        <li>Estou sim</li>
        <li>Muito bem e consigo?</li>
      </random>
    </template>
  </category>
</aiml>
```

WORDNET

O Wordnet [8] consiste numa extensa base de dados lexical e servirá como suporte ao interpretador, como demonstrado na arquitetura.

Sendo o Wordnet uma interface presente no módulo NLTK [10], este permite efetuar *Natural Language Processing* a partir do *input* fornecido pelo utilizador.

Desta forma, as principais funcionalidades do Wordnet são:

- Sinónimos, hipónimos, hiperónimos e definições de palavras através de *Synsets* (conjuntos de sinónimos);
- Antónimos através de *Lemmas*;
- Similaridade entre palavras;
- *Stemming* e *Lemmatization* com o uso de Morphy.

Pretende-se utilizar o WordNet para efetuar a interpretação do que é escrito no chat, pela parte do utilizador, nomeadamente efetuar descrições sobre determinadas palavras, como funcionalidade extra, e fazer uso de Natural Language Processing, para intercalar com o uso do AIML.

Guia de Instalação e Primeiros-Passos

De modo a aplicar a interface Wordnet, é necessário instalar o NLTK e para isso basta executar o comando `pip install -U nltk` na linha de comandos.

De seguida, num ficheiro Python ou usando o Python diretamente pela consola, é também necessário importar o NLTK e transferir os ficheiros de *corpora* através das seguintes instruções:

```
import nltk
nltk.download()
```

Depois dos passos anteriores terem sido elaborados e visto que os módulos necessários já estão instalados, é possível proceder com a importação do NLTK, através do:

```
from nltk.corpus import wordnet as wn
```

Assim sendo o Wordnet vai nos fornecer funcionalidades como:

- hipónimos, sinónimos, hiperónimos e definições de palavras por meio de **synonym ring** ou **synset** que é um conjunto de dados sendo eles considerados semanticamente equivalentes;
- Similaridade entre palavras;
- Através do Lemmas temos os Antónimos;
- Stemming e Lemmatization usando o Morphy

O *Stemming* vai permitir remover de uma palavra os seus prefixos e sufixos, alterando-os para a sua forma inicial. Já a *Lemmatization* é um método baseado no *WordNet's built-in morphy function*, sendo semelhante ao *Stemming* o que faz é converter a palavra na sua forma inicial, sendo que a palavra raiz também conhecida com *Lemma*, está presente no dicionário.

Em comparação, o Stemming é mais lento porque têm de verificar se o *Lemma* está presente no dicionário.

Fazendo um apanhado geral, o WordNet servirá para realizar a interpretação do que é escrito no chat, pela parte do utilizador, como especificamente efetuar descrições sobre determinadas palavras, sendo uma funcionalidade extra, e usar processamento natural da linguagem, para intercalar com o uso do AIML.

REFERÊNCIAS

[1] Wikipedia, “Chatterbot,” 2018. [Online]. Available: https://pt.wikipedia.org/wiki/Chatterbot. [Accessed: 15-Apr-2018].

[2] M. Shinmi, “ChatBots - Tendência 2017,” 2017. [Online]. Available: https://www.oxigenweb.com.br/artigos/chatbots-tendencia-2017/. [Accessed: 15-Apr-2018].

[3] L. R. Oliveira, “Você conhece os Chatbots? Descubra aqui o que são e como usá-los para otimizar o atendimento ao cliente,” 2017. [Online]. Available: https://marketingdeconteudo.com/chatbots/. [Accessed: 15-Apr-2018].

[4] D. Britz, “Deep Learning for Chatbots, Part 1 – Introduction,” 2016. [Online]. Available: http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/. [Accessed: 15-Apr-2018].

[5] Pavel Surmenok, “Chatbot Architecture,” 2016. [Online]. Available: https://medium.com/@surmenok/chatbot-architecture-496f5bf820ed. [Accessed: 15-Apr-2018].

[6] Wikipedia, “AIML,” 2017. [Online]. Available: https://pt.wikipedia.org/wiki/AIML. [Accessed: 16-Apr-2018].

[7] Quora, “What is the typical architecture of an AI chatbot?,” 2017. [Online]. Available: https://www.quora.com/What-is-the-typical-architecture-of-an-AI-chatbot. [Accessed: 16-Apr-2018].

[8] Princeton University, “WordNet,” 2018. [Online]. Available: https://wordnet.princeton.edu. [Accessed: 15-Apr-2018].

[9] Cláudio L. V. Oliveira, “Visão geral sobre Artificial Intelligence Markup Language (AIML).” [Online]. Available: http://www.profclaudio.com.br/arquivos/AIML-Visao_Geral.pdf. [Accessed: 16-Apr-2018].

[10] NLTK Project, “WordNet Interface.” [Online]. Available: http://www.nltk.org/howto/wordnet.html. [Accessed: 16-May-2018].