Universidade de Aveiro

Departamento de Electrónica, Telecomunicações e Informática

MEI

Agente Conversacional

André Rodrigues Nº 60042

João Amaral Nº 65772

João Pires Nº 83283

Mafalda Rodrigues Nº 72089

Mariana Pereira Nº 83319

Trabalho realizado no âmbito da disciplina

Sistemas Inteligentes

Índice

1. Introdução

No âmbito da cadeira Sistemas Inteligentes, foi solicitado a implementação de uma tecnologia de agente conversacional, tendo como intuito o desenvolvimento de um *chatbot* através da inteligência artificial.

A motivação do grupo para a escolha e implementação deste tema, foi de encontro à importância que este pode revelar atualmente na implementação de um *chat* numa loja online, dando ajuda ao utilizador em qualquer assunto. Por consequência, sabendo das deficiências que estes programas podem ter na sua comunicação foi uma motivação extra a melhorar esta vertente, de modo a maximizar a eficiência do *chatbot* (agente conversacional) que fosse criado.

Por fim, o principal objetivo deste projeto será a criação de um agente conversacional sobre um determinado tema.

2. O que é um *Chatbot*?

O termo *chatbot* refere-se a um programa ou serviço de resposta automática baseado em regras, podendo ou não utilizar inteligência artificial, com o qual o utilizador poderá interagir e estabelecer um diálogo através de mensagens de texto ou de voz.

O principal objetivo de um *chatbot* é responder ao utilizador de forma convincente, procurando simular da melhor forma possível o comportamento de um ser humano durante o diálogo, sem que o utilizador se aperceba de que não está a dialogar com uma pessoa real.

Um *chatbot* que seja unicamente baseado em regras apresenta algumas limitações, pois obedece a fluxos de navegação bem definidos e funciona através de *keywords* ou de comandos específicos. Isto implica que, caso o *chatbot* não compreenda a mensagem inserida pelo utilizador, não saiba como lhe responder.

Por outro lado, um *chatbot* que tenha presente inteligência artificial na sua implementação terá capacidade para aprender e entender linguagem natural. Desta forma, o *chatbot* irá ter uma tendência a melhorar com o tempo e com a utilização, tornando-se mais convincente e mais credível quanto maior for a sua frequência de utilização e o número de pessoas que o utilizar.

Uma vantagem da utilização dos *chatbots* é a simplificação de certas interações – especialmente as que são repetitivas – tornando possível a aplicação de respostas automáticas para determinadas perguntas ou afirmações introduzidas.

Alguns exemplos da aplicação de *chatbots* podem ser encontrados, por exemplo, em alguns serviços oferecidos por SMS, no atendimento automático por telefone, em algumas aplicações de *messaging* que permitem comunicar com *chatbots* existentes ou a criação de novos *bots* (*Facebook, Telegram, Kik*) por forma a automatizar certas mensagens, entre outros.

3. Arquitetura

Nesta fase inicial do trabalho o grupo teve em consideração dois elementos ao nível da arquitetura, sendo estes o *workflow* e os componentes que constituem o nosso *chatbot*.

Assim, foi desenhada uma arquitetura relativamente fluída ao nível de processos, mas flexível do ponto de vista da escalabilidade. Para atingir esta escalabilidade, os componentes foram divididos em componentes primários e secundários.

Assim o grupo conceptualizou a arquitetura que se pode ver na seguinte imagem:

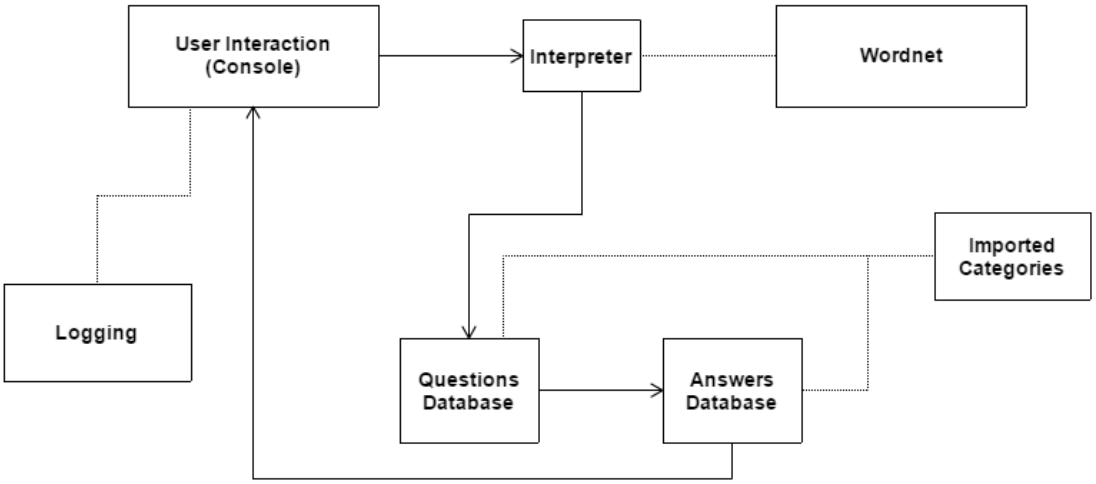

Figura 1 - Arquitetura do *Chatbot*

Antes de passar à descrição sucinta do fluxo e dos vários componentes, o grupo realça que esta arquitetura corresponde a um design inicial e que pode sofrer modificações durante o processo de desenvolvimento.

3.1. *Workflow* do *Chatbot*

Após alguma pesquisa, o grupo percebeu que o modo de funcionamento de um *chatbot* era semelhante em todos os *chatbots* analisados.

Como tal, o grupo utilizou estes exemplos e criou um *workflow* que contém um processo central sólido e que é bastante escalável, podendo ser adicionadas novas funcionalidades à medida que o desenvolvimento avance.

Assim, começou por ser definido como ponto inicial uma interação entre o *chatbot* e o utilizador. Aqui serão capturados os inputs do utilizador, na forma de frases, que irão ser analisadas por um interpretador.

Esse interpretador, com auxílio de ferramentas linguísticas existentes no Wordnet, irá tentar atribuir ao input do utilizador uma categoria, que estará associada a uma *tag* AIML, que estarão disponíveis na Questions Database.

Após essa atribuição o *chatbot* utilizará essa mesma categoria, mas desta vez para ir buscar uma resposta ao input do utilizador, estando estas respostas disponíveis na sua Answers Database.

O *workflow* ficará concluído quando a resposta encontrada for enviada ao utilizador no ecrã de interação, 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 (Imported Categories) para auxiliar no preenchimento da Questions/Answers Databases.

3.2. Componentes do chatbot

O grupo faz uma distinção clara dos vários componentes do *chatbot*, dividindo-os em componentes primários ou componentes secundários, consoante a sua função na arquitetura.

3.2.1. Componentes primários

Os componentes primários são os componentes considerados essenciais na estrutura principal do *chatbot*, sendo que é sobre estes que incidem os principais processos contidos no *workflow* descrito anteriormente.

Assim os componentes primários são:

- User Interaction – Sistema que permite a conversa entre o utilizador e o *chatbot*, estando neste momento projetado que tome a forma de uma consola de comandos.
- Interpreter – Um interpretador que tem uma das funções mais importantes de todo o processo e que consiste na interpretação dos inputs do utilizador e a sua associação a uma questão e *tag* AIML para que possa ser dada uma resposta adequada.
- Questions/Answers Database – São o sistema de base de dados do *chatbot*. Atualmente o grupo pensa em ter *tags* definidas por AIML às quais são associadas questões típicas colocadas pelo utilizador, bem como respostas que devem ser dadas a essas mesmas questões. Estas bases de dados são, no fundo, o conhecimento do *chatbot*.

3.2.2. Componentes secundários

Os componentes secundários são componentes de suporte que têm como principal objetivo o apoio aos componentes principais. Este apoio toma a forma de novas funcionalidades que modificam ou facilitam os processos existentes no *workflow* principal do *chatbot*.

Os componentes secundários atualmente previstos são:

- Logging – Sistema de apoio ao sistema de interação com o utilizador e que guarda o historial das conversas com o *chatbot* que podem posteriormente ser utilizados de várias formas.
- Wordnet – O Wordnet será primariamente utilizado como suporte ao interpretador, tendo como principal função utilizar as suas capacidades a nível de sinónimos e contextualização frásica para auxiliar na gestão dos inputs do utilizador e assim facilitar todo o processo pergunta/resposta associado às databases.
- Imported Categories – São utilizadas para construir a base de conhecimento do *chatbot*. O grupo tem neste momento várias *categorias* que poderão ser utilizadas quando for decidido em concreto a especialização do chatbot. Estas coleções poderão preencher as databases pergunta/resposta com dados já existentes, facilitando assim o processo de "aprendizagem" do mesmo.

4. O que é o 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.

4.1. Estrutura de um Projeto

Para a implementação de um projeto AIML, é necessário conter a estrutura descrita na Figura 2.

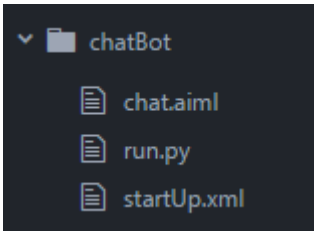

Figura 2 - Estrutura de um projeto AIML

4.1.1. startUp.xml

O ficheiro denominado startUp.xml, tem como intuito carregar os ficheiros AIML pretendidos, pois estes irão corresponder aos ficheiros de aprendizagem do *chatbot*. Em suma, os ficheiros AIML contêm os padrões de pergunta e resposta que o *chat* irá ter.

```
<aiml version="1.0.1" encoding="UTF-8">
    <!-- std-startup.xml -->

    <!-- Category is an atomic AIML unit -->
    <category>

        <!-- Pattern to match in user input -->
        <!-- If user enters "LOAD AIML B" -->
        <pattern>LOAD AIML B</pattern>

        <!-- Template is the response to the pattern -->
        <!-- This learn an aiml file -->
        <template>
            <learn>chat.aiml</learn>
            <!-- You can add more aiml files here -->
            <!--<learn>more_aiml.aiml</learn>-->
        </template>

    </category>

</aiml>
```

Figura 3 - Conteúdo do ficheiro startUp.xml

4.1.2. chat.aiml

O ficheiro denominado chat.aiml irá conter as regras que o *chatbot* irá ter.

As principais tags para a sua construção são:

- <aiml> → *tag* que representa o inicio e o fim de um ficheiro AIML.
- <category> → representa a unidade básica de conhecimento em AIML, esta contém várias tags dentro da mesma, pois nestas é realizada a interpretação de uma mensagem escrita por um utilizador dando uma resposta pretendida.
- <pattern> → representa um padrão, ou seja, um possível input que o utilizador pode fazer.
- <template> → contém a resposta de um input do utilizador.

Como podemos observar na Figura 4, o código utiliza a expressão "HELLO" para identificar um cumprimento e responde de forma adequada, através da *tag* "<template>".

```
<category>
        <pattern>HELLO</pattern>
        <template>
            Well, hello!
        </template>
</category>
```

Figura 4 - Excerto do conteúdo do ficheiro chat.aiml

Na figura 5, o código permite que o *chatbot* compreenda o input dado pelo utilizador "One Time I *", onde o elemento * representa qualquer conjunto de carateres. Por fim, esta resposta contém as tags *random*, representado um conjunto de respostas possíveis dadas aleatoriamente pelo *chatbot*.

```
<category>
    <pattern>ONE TIME I *</pattern>
    <template>
        <random>
            <li>Go on.</li>
            <li>How old are you?</li>
            <li>Be more specific.</li>
            <li>I did not know that.</li>
            <li>Are you telling the truth?</li>
            <li>I don't know what that means.</li>
            <li>Try to tell me that another way.</li>
            <li>Are you talking about an animal, vegetable or mineral?</li>
            <li>What is it?</li>
        </random>
    </template>
</category>
```

Figura 5- Excerto do conteúdo do ficheiro chat.aiml

4.1.3. run.py

O ficheiro denominado run.py tem como objetivo principal criar o ficheiro AIML, fazer a sua aprendizagem e posteriormente carregar o resto dos ficheiros xml.

Depois disso, ele está pronto para conversar, e entramos num *loop* infinito que continuará a solicitar ao utilizador uma mensagem até que este escreva "quit" pois assim ele sairá do programa.

Contudo, quando o *chatbot* carrega muitos ficheiros AIML, pode levar cada vez mais tempo ao realizar a sua aprendizagem. Desta forma, para que exista uma otimização deste processo é gravado para um ficheiro a memória até ao momento do *chat* o que implica um melhoramento da performance do *chat*.

```
import aiml
import os

kernel = aiml.Kernel()

if os.path.isfile("bot_brain.brn"):
    kernel.bootstrap(brainFile = "bot_brain.brn")
else:
    kernel.bootstrap(learnFiles = "std-startup.xml", commands = "load aiml b")
    kernel.saveBrain("bot_brain.brn")


while True:
    message=raw_input("Enter your message to the bot: ")
    #print message
    if message == "quit":
        exit()
    elif message=="save":
    kernel.saveBrain("bot_brain.brn")
    else:
    print kernel.respond(message)
    #bot_response = kernel.respond(message)
```

Figura 6 - Conteúdo do ficheiro run.py

5. Implementação AIML

Além do facto do *chatbot* ser propositado para um ou vários temas exclusivamente, o grupo decidiu implementar vários módulos para que o utilizador possa utilizar, assim que o desejar, em paralelo com a linguagem AIML.

Os módulos propostos pelo grupo ate à data foram os seguintes:

- Tradução de frases/palavras;
- Aquisição de informação através da Wikipédia;
- Cálculos matemáticos.

5.1. Cálculos matemáticos

Como o nome do módulo sugere, este módulo tem como objetivo receber o *input* do utilizador, em que o mesmo corresponde a uma questão do cariz matemático, e devolver a resolução do problema.

5.2. Aquisição de informação através da Wikipédia

O intuito do módulo é adquirir o primeiro parágrafo de qualquer artigo da Wikipédia que o utilizador pretenda.

A justificação de adquirir somente o primeiro parágrafo do artigo em questão deve-se à estrutura que a Wikipédia apresenta, tendo no primeiro paragrafo a informação básica e necessária sobre a definição/explicação do assunto a que o título corresponde.

5.3. Tradução de frases/palavras

O objetivo deste módulo é que o *chatbot* possa traduzir, através da utilização de ferramentas da Google, assim que requerido, palavras ou frase inseridas pelo utilizador.

Neste módulo foi utilizada a biblioteca chamada *Beautiful Soap*. Esta biblioteca tem a finalidade de adquirir informação de ficheiros HTML e XML.

Desta maneira é possível mapear e adquirir a informação pretendida embutida no elemento HTML correspondente, o que neste caso é a tradução já efetuada.

Para se conseguir efetuar a tradução em si, é utilizado o seguinte site da Google em que se pode inserir como parâmetro a linguagem original, a linguagem para a qual se pretende traduzir e a frase/palavra em si, adquirindo o resultado final.

Em anexo a este relatório encontra-se uma demonstração (bem como um ficheiro readme.txt) de tradução de palavras.

5.4. Utilização do modulo de tradução

Para utilizar os módulos, o grupo determinou a utilização de *keywords* no input do utilizador para que o *bot* consiga determinar qual módulo irá utilizar. Por exemplo, para efetuar uma tradução, o *bot* atualmente está à espera que o utilizador insira o seguinte: "*Translate to spanish Hello"*. Como o input tem a palavra *Translate*, a função desenvolvida vai adquirir a língua de tradução como também a frase/palavra que o utilizador inseriu, devolvendo a tradução retirada no site da google como explicado anteriormente.

6. Base de Dados

Sendo um *chatbot* um agente inteligente, um dos componentes essenciais ao seu funcionamento é a base de dados, o conhecimento do *bot*. Este componente é bastante importante pois determina a qualidade da "aprendizagem" que é feita ao nível da inteligência que o agente inteligente é capaz de mostrar no diálogo com o utilizador. Com uma boa base de conhecimento e correta representação da mesma, o *chatbot* é capaz de escolher informação pertinente e retornar respostas relevantes ao utilizador.

Foi pensado pelo grupo a utilização de um sistema de *tags* (categorias) consoante as questões e afirmações inseridas pelo utilizador. Cada *tag* estaria associada a um conjunto de questões/afirmações e respetivas respostas, pertencentes ao tema denotado pela *tag*. Assim sendo, as *tags* criadas ficariam também definidas na base de dados do *chatbot*.

Foram encontradas diversas bases de conhecimento com conteúdo pertinente disponíveis para utilização, entre elas são de salientar:

- "Free A.L.I.C.E. AIML Set" [1] – Um conjunto de regras AIML, em inglês e da autoria da AI Foundation. Inclui categorias como:
    - Música (compositores, grupos, preferências musicais);
    - Computadores (software, hardware, história);
    - História (guerra civil, definições);
    - Literatura (livros, escritores);
    - Política (definições).

- "Rosie AIML Set" [2] – Baseado no projeto ALICE 2.0. Inclui categorias como:
    - Datas;
    - Estações;
    - Filtro de insultos.

Embora atualmente não exista uma especificação no que diz respeito ao tema escolhido para a base de conhecimento do *chatbot*, o grupo pretende especializar o *bot* num único tema ou grupo restrito de temas futuramente.

7. WordNet

O WordNet é uma interface (*corpus reader*) [3] presente no módulo NLTK [4], que permite efetuar *Natural Language Processing*. *Natural Language Processing* encontra-se no âmbito da inteligência artificial, e tem funcionalidades que permitem tratar interações em texto entre humanos e máquinas.

Para aceder à interface WordNet basta instalar o NLTK através da linha de comandos com o comando "pip install -U nltk". Seguidamente, num módulo de Python basta importar o NLTK e transferir ficheiros de *corpora* através das seguintes linhas de código:

- import nltk
- nltk.download()

Tendo já sido transferidos os ficheiros necessários é, então, possível usar a interface WordNet, importando a mesma a partir do NLTK, com a linha de código "from nltk.corpus import wordnet as wn".

O WordNet fornece funcionalidades como:

- 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.

Em anexo a este relatório encontra-se uma demonstração (bem como um ficheiro readme.txt) de funcionalidades fundamentais do WordNet.

7.1. Potencial utilização dos modelos de ordem variável de Markov

Com base no que foi lecionado na disciplina de Teoria Algorítmica da Informação, surgiu a ideia de utilizar a ideia conceptual dos modelos de ordem variável de Markov, para interpretar texto escrito no *chat*, pela parte do utilizador, tendo por base uma base de dados que consiste em diálogos de *chats*, presentes num *corpus* do NLTK [5].

A ideia é efetuar contagens de bigramas ou trigramas presentes nos diálogos, bem como pontuação (!?.), e determinar as probabilidades de certas palavras ou sinais de pontuação se seguirem a esses bigramas ou trigramas.

Para cada bigrama ou trigrama seria associada a frase em que o mesmo aparece, e essa frase é guardada numa base de dados, sendo possível limitar o número de frases para cada bigrama ou trigrama. Seguidamente é possível efetuar semelhança semântica, recorrendo ao módulo "gensim" [6], ou fazer o cálculo da *cosine distance* entre uma frase escrita pelo utilizador que contenha determinado bigrama ou trigrama, e as frases que contêm os mesmos bigramas ou trigramas, podendo ser dada como resposta ao utilizador a frase contida na base de dados que tem a maior semelhança ao que foi escrito pelo utilizador.

A utilização desta ideia seria bastante apropriada no caso de o *chatbot* não conseguir interpretar o que o utilizador escreve, dando assim uma resposta associada ao que foi escrito.

8. Conclusão

Com a elaboração deste relatório foi possível conceber uma arquitetura para o *chatbot* a criar pelo grupo, bem como descrever todas as componentes fundamentais ao funcionamento do mesmo. Assim sendo, o grupo tem agora uma melhor ideia de como funciona um *chatbot* e como implementar um.

Existe uma componente na arquitetura do *chatbot* a criar, que ainda não está totalmente definida, que é o uso de modelos de Markov de ordem variável, para usar no *Interpreter*, bem como para efetuar aprendizagem com base em ficheiros de texto que contêm diálogos em *chats*. O facto de o grupo não ter definido ainda este tema como uma componente do *chatbot* deve-se à incerteza que existe sobre se existe *corpora* apropriados para o que se pretende desenvolver, se o *chatbot* se pode tornar demasiado lento ao processar grandes quantidades de texto (*corpora*), bem como se a *corpora* é suficiente (tem de ser suficientemente extensa para uma aprendizagem minimamente eficaz).

Futuramente pretende-se efetuar a ligação de todos os componentes do protótipo da arquitetura, nomeadamente todas as que estão relacionadas diretamente com o *Interpreter* (*User Interaction*, WordNet, *Answers*/*Questions Databases* e, possivelmente, modelos de Markov de ordem variável. Mais fulcral é conseguir estabelecer a ligação entre o AIML e a interpretação do texto escrito pelo utilizador.

Referências

[1] “Free A.L.I.C.E. AIML Set,” [Online]. Available: https://code.google.com/archive/p/aiml-en-us-foundation-alice/. [Acedido em 20 04 2017].

[2] “Rosie AIML Set,” [Online]. Available: https://github.com/pandorabots/rosie/tree/master/lib/aiml. [Acedido em 20 04 2017].

[3] “WordNet Interface,” [Online]. Available: http://www.nltk.org/howto/wordnet.html. [Acedido em 19 Abril 2017].

[4] “NLTK,” [Online]. Available: http://www.nltk.org/. [Acedido em 19 Abril 2017].

[5] “2. Accessing Text Corpora and Lexical Resources,” [Online]. Available: http://www.nltk.org/book/ch02.html. [Acedido em 19 Abril 2017].

[6] “gensim 2.0.0,” [Online]. Available: https://pypi.python.org/pypi/gensim. [Acedido em 19 Abril 2017].