**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 2](#_Toc484890228)

[2. O que é um Chatbot? 3](#_Toc484890229)

[3. Chatbot ao Serviço da Comunidade 4](#_Toc484890230)

[4. Arquitetura 5](#_Toc484890231)

[4.1. Workflow do Chatbot 5](#_Toc484890232)

[4.2. Componentes do chatbot 6](#_Toc484890233)

[4.2.1. Componentes primários 6](#_Toc484890234)

[4.2.2. Componentes secundários 7](#_Toc484890235)

[5. O que é o AIML? 8](#_Toc484890236)

[5.1. Estrutura de um Projeto 8](#_Toc484890237)

[5.1.1. startUp.xml 8](#_Toc484890238)

[5.1.2. chat.aiml 8](#_Toc484890239)

[5.1.3. run.py 10](#_Toc484890240)

[6. Implementação AIML 11](#_Toc484890241)

[6.1. Cálculos matemáticos 11](#_Toc484890242)

[6.2. Aquisição de informação através da Wikipédia 11](#_Toc484890243)

[6.3. Tradução de frases/palavras 12](#_Toc484890244)

[6.4. Aquisição de informação sobre meteorologia 12](#_Toc484890245)

[7. Base de Dados 13](#_Toc484890246)

[8. WordNet 14](#_Toc484890247)

[9. Modelo probabilístico 15](#_Toc484890248)

[10. Interpretador 16](#_Toc484890249)

[11. Projeto em Python + AIML 17](#_Toc484890250)

[12. Conclusão 17](#_Toc484890251)

[Referências 18](#_Toc484890252)

# Introdução

No âmbito da disciplina de 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 (Universidade de Aveiro), não sendo obrigatório interagir com o *bot* sobre este tema.

Neste relatório encontra-se uma descrição da arquitetura do *bot*, bem como da implementação do mesmo (AIML, modelo probabilístico e interpretador).

# 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.

# *Chatbot* ao Serviço da Comunidade

Uma das principais preocupações do grupo durante o desenvolvimento do projecto foi onde o mesmo poderia ser aplicado ao serviço da comunidade académica. Ao invés da criação de um projecto que se limitava a algo que seria entregue e esquecido o grupo ambiciona que o seu chatbot possa ter aplicações práticas fora do contexto da disciplina.

Após reuniões entre os vários membros e em conjunto com o docente foi decido que uma possível aplicabilidade do chatbot criado, e aproveitando o facto do mesmo estar a ser feito com recurso à língua inglesa, seria utiliza-lo como recurso para esclarecer novos alunos, nomeadamente a alunos de Erasmus.

O chatbot poderia ser um serviço integrado com a Universidade de Aveiro, idealmente numa plataforma web para facilitar o acesso ao mesmo, e seria primariamente um meio de fornecer informação útil como horários de cantinas, números e descrições de departamentos, localização, cursos fornecidos, etc.

# 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:

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.

# *Workflow* do *Chatbot*

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

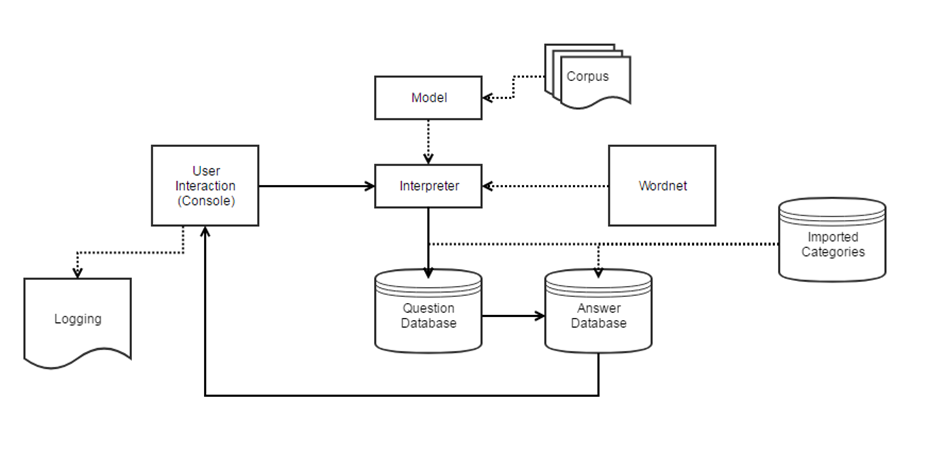
Como tal, o grupo utilizou estes exemplos e criou um *workflow* que contém um processo central sólido e que é bastante escalável, podendo ser adicionadas novas funcionalidades à medida que o desenvolvimento avance.

Figura 1 - Arquitetura do Chatbot

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.

Os componentes de modelo e de *corpus* são utilizados para dar inteligência ao *chatbot*, e também servirão como recurso, caso o *bot* não consiga responder a um *input* do utilizador com recurso ao AIML. Estes pontos serão discutidos mais á frente neste relatório.

# 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.

# 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*.

# 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.

# 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.

# Estrutura de um Projeto

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

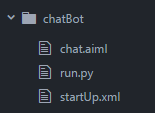


Figura 2 - Estrutura de um projeto AIML

# 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.



Figura 3 - Conteúdo do ficheiro startUp.xml

# 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>”.

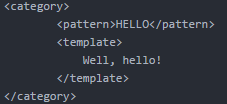


Figura 4 - Excerto do conteúdo do ficheiro chat.aiml

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*.

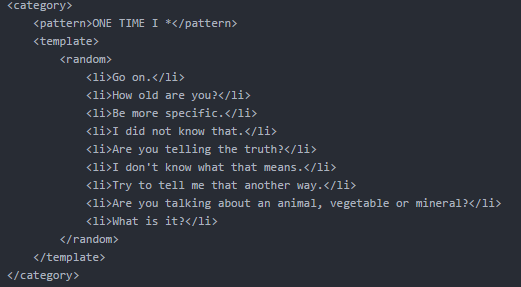


Figura 5- Excerto do conteúdo do ficheiro chat.aiml

# 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*.

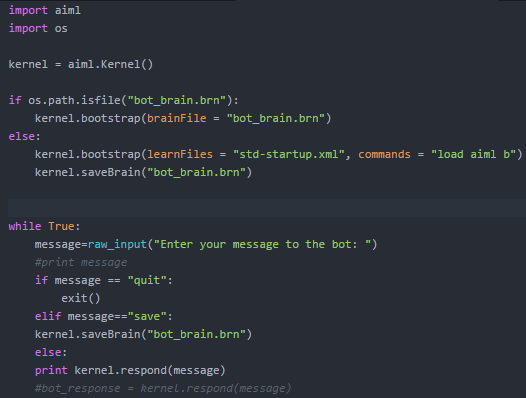


Figura 6 - Conteúdo do ficheiro run.py

# Implementação AIML

Além do facto do *chatbot* ter como propósito 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:

* Cálculos matemáticos;
* Aquisição de informação através da Wikipédia;
* Tradução de frases/palavras;
* Aquisição de informação sobre meteorologia.

# 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.

Para utilizar o módulo de calculo é necessário indicar no chat do *bot* a *keyword* “Calculate” para que o *bot* saiba que tem de efetuar um calculo matemático devolvendo o resultado.

Exemplo:

* Calculate 1+1

# 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.

Para utilizar o módulo de pesquisa, é necessário indicar no chat do *bot* a *keyword* “Search” para que o *bot* utilize a palavra a seguir como parâmetro na pesquisa sobre a informação pedida. Devolvendo a primeira frase da pagina da Wikipédia em questão.

Exemplo:

* Search Benfica

# 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](http://translate.google.com/m?hl=%25s&sl=%25s&q=%25s) 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.

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.

# Aquisição de informação sobre meteorologia

Através do uso da API do OpenWeatherMap é possível adquirir varias métricas da meteorologia no momento da pesquisa.

É possível adquirir o estado do céu (nublado ou limpo), a temperatura em graus Celcius e a velocidade do vento na cidade pretendida.

Para utilizar o modulo do tempo, é necessário indicar no chat do *bot* um conjunto de *keywords* para que o *bot* utilize o nome da cidade e do país previamente inserido pelo utilizador. Devolvendo as varias métricas. Ao inserir a cidade é necessário inserir igualmente o país devido ao facto que existe varias cidades com o mesmo nome, mas em países diferentes.

Exemplo:

* How is the weather in Aveiro,PT

# 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.

Outro ponto de debate foi o que fazer em relação ao número massivo de bibliotecas existente, já que o grupo não queria apenas utilizar bibliotecas já feitas.

Inicialmente o grupo pensou em eliminar algumas das categorias existentes nas bibliotecas da A.L.I.C.E, substituindo-as por categorias criadas pelo grupo com o intuito de dar um toque pessoal às mesmas. Após discussões com o docente foi decido que a remoção de conhecimento já existente seria contra produtivo e que ao invés disso o grupo poderia adicionar novo conhecimento que ainda não estivesse disponível. Assim, e devido ao desejo já manifestado pelo grupo de dar utilidade ao chatbot para lá do âmbito da cadeira foi decido que seria criada uma categoria, semelhante às existentes na biblioteca da A.L.I.C.E, que seria sobre a Universidade de Aveiro.

Esta categoria contém assim informações referentes a:

* + O que é a universidade, a sua criação e localização;
  + Informação sobre o número de departamentos e alunos da UA;
  + Serviços oferecidos (cantina, biblioteca), os seus horários de funcionamento e preço no caso das cantinas;
  + Cursos da UA.
  + Bibliotecas da UA.
  + Residências de Estudantes.

# 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.

No contexto deste trabalho, o WordNet foi utilizado para efetuar descrições sobre determinadas palavras, como funcionalidade extra.

# Modelo probabilístico

Com base no que foi lecionado na disciplina de Teoria Algorítmica da Informação, surgiu a ideia de utilizar a ideia conceptual dos modelos probabilísticos 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]. Este texto utilizado para treinar o modelo contém cerca de 17000 linhas, portanto cerca de 17000 frases diferentes.

O algoritmo desenvolvido para criar o modelo probabilístico encontra-se resumido de seguida.

* Foram efetuadas contagens de bigramas (pares de palavras) presentes nos diálogos, para determinar as suas frequências no *corpus*.
* Para cada um dos bigramas detetados é associada uma lista de frases que contenham esses mesmos bigramas.

O uso deste modelo é feito da seguinte maneira:

* Detetar a *keyword* fundamental do *input* do utilizador (frase).
* Verificar a posição da *keyword* na frase (se se encontra no início, no meio ou no fim da mesma).
  + Juntar em dois bigramas a *keyword* com a palavra seguinte e com uma palavra anterior, respetivamente, se a *keyword* estiver no meio da frase.
  + Juntar num bigrama a *keyword* com a palavra seguinte se a *keyword* for a primeira palavra da frase.
  + Juntar num bigrama a *keyword* com a palavra anterior se a *keyword* for a última palavra da frase.
* Identificar o conjunto de frases que contêm determinado bigrama e calcular a distância de cosseno (*cosine distance*) entre cada uma dessas frases e o *input* do utilizador. A distância de cosseno é um valor entre 0 e 1, em que 0 dissemelhança total, e 1 é semelhança total.
  + Se esse número de frases for menor que 16, é feita uma escolha aleatória de uma frase entre as 5 que têm maior semelhança com a frase que o utilizador introduziu. É devolvida, ao utilizador, essa frase aleatoriamente escolhida.
  + Se esse número de frases for maior ou igual a 16, é utilizado um fator de 4 para efetuar uma divisão com função piso do número de frases total (), sendo *x* o número de frases final que têm maior semelhança com a frase de *input*. Por fim é efetuada uma escolha aleatória de uma frase entre as *x* escolhidas. É devolvida, ao utilizador, essa frase aleatoriamente escolhida.

É importante notar que a criação do modelo para 17000 linhas de *corpus* demora cerca de dois minutos (cerca de 129 segundos), no entanto, as respostas dadas pelo *bot*, consoante o modelo, são instantâneas, pois a pesquisa no modelo é eficiente, devido ao uso de dicionários com bigramas como chaves e conjuntos de frases como valores.

# Interpretador

O interpretador do *chatbot* tem uma estrutura simples que se baseia em condições. Nestas condições analisa-se o *input* do utilizador. O utilizador pode inserir *inputs* variados, como:

* “How is the weather in PT,Aveiro”;
* “Calculate 2+1”;
* “Translate to english viva”;
* “Search Benfica”

Para cada um destes tipos de *inputs*, o *bot* responde de acordo com as funcionalidades extra que foram incluídas no mesmo, nomeadamente, meteorologia, função de calculadora básica, tradutor e pesquisa na Wikipédia.

Caso nenhum dos *inputs* anteriores seja reconhecido, verifica-se se o *input* equivale a alguma das *patterns* presentes no ficheiro AIML criado pelo grupo (ua.aiml). Se sim, é dada uma resposta consoante a *pattern*, se não, o *bot* irá utilizar o modelo probabilístico criado. Se o utilizador escrever apenas uma ou duas palavras, o modelo não é utilizado, porque são precisas pelo menos 3 palavras para efetuar a comparação de *strings* (distância de cosseno).

Caso o utilizador escreva 2 palavras, é identificada a *keyword* desse *input* e é dada uma definição da mesma, recorrendo ao Wordnet. Caso o utilizador escreva 1 palavra é dada também a definição da mesma, recorrendo ao Wordnet.

Se, porventura, o utilizador escrever 3 ou mais palavras e o modelo não conseguir responder, por não existirem bigramas no modelo que contenham pares desse *input*, é devolvido o mesmo *input* ao utilizador, mas misturando as palavras do *input* inicial, de forma aleatória.

Se o utilizador não escrever nada e inserir isso como *input*, o *chatbot* apresentará a mensagem “you have at least to write a word”.

# Projeto em Python + AIML

O projeto contém o *script* de Python “chatbot.py” que é o ficheiro a ser executado. Existem ainda outros ficheiros, como ficheiros AIML que servem de base de conhecimento do *bot.*

O ficheiro “dwnld\_wn.py” é para ser executado para instalar o *corpus* “wordnet”, necessário para a execução do “chatbot.py”.

O ficheiro “corpus\_to\_txt.py” pode ser executado para criar o ficheiro “final.txt” que irá conter o *corpus* para aprendizagem com o modelo desenvolvido.

Ao executar o “chatbot.py” é perguntado ao utilizador se pretende utilizar os ficheiros AIML do *bot* A.L.I.C.E, que não foram desenvolvidos pelo grupo, para assim o utilizador testar o *bot* com recurso a uma maior quantidade de ficheiros AIML. Foi adicionada esta parte ao programa após o grupo ter falado com o Professor e ter sido decidido que podia ser possível executar o *bot* com ou sem ficheiros AIML não criados pelo grupo.

# Conclusão

Com a elaboração deste trabalho, o grupo tem agora uma melhor ideia de como funciona um *chatbot* e como implementar um, apesar de ter havido algumas dificuldades na criação de um.

A ideia da criação do modelo probabilístico, que era um dos objetivos deste trabalho, pareceu interessante ao grupo e foi cumprido esse mesmo objetivo. No entanto, as respostas do *chatbot*, recorrendo ao modelo, por vezes não são as mais coerentes, dado o *input* do utilizador.

Após ter sido utilizado o *bot* com o modelo criado e terem sido feitos alguns testes, conclui-se que modelo funciona melhor com *strings* com mais do que 5 ou 6 palavras diferentes, porque a comparação de *strings* funciona melhor com maiores *strings*.

O *bot* demora cerca de 2 minutos a criar o modelo, o que se considera eficiente, visto que o *corpus* de aprendizagem contém 17000 linhas de texto.

A pesquisa no modelo, de bigramas e retorno de uma frase ao utilizador é extremamente rápido, devido ao facto de ter sido utilizada a estrutura de dados *dict*, em Python.

Depois de testado o *bot* com recurso aos ficheiros AIML do *chatbot* A.L.I.C.E, foi evidente notar que a inteligência e aprendizagem do *bot* é muito melhor do que com o uso do modelo probabilístico criado.

# 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]. |