Chatbot para Ajuda de Novos Alunos

Por

João Rafael da Silva Carvalho

Orientador: Pedro Miguel Mestre Alves da Silva

Co-orientador: Carlos Manuel José Alves Serôdio

Dissertação submetida à
UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO
para obtenção do grau de
MESTRE
em Engenharia Electrotécnica e de Computadores, de acordo com o disposto no
Regulamento Geral dos Ciclos de Estudos Conducentes ao Grau de Mestre na UTAD
DR, 2º série-N.º133- Regulamento nº 658/2016 de 13 de julho de 2016

Chatbot para Ajuda de Novos Alunos

Por

João Rafael da Silva Carvalho

Orientador: Pedro Miguel Mestre Alves da Silva

Co-orientador: Carlos Manuel José Alves Serôdio

Dissertação submetida à
UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO
para obtenção do grau de
MESTRE
em Engenharia Electrotécnica e de Computadores, de acordo com o disposto no
Regulamento Geral dos Ciclos de Estudos Conducentes ao Grau de Mestre na UTAD
DR, 2º série-N.º133- Regulamento nº 658/2016 de 13 de julho de 2016

Orientação Científica :

Pedro Miguel Mestre Alves da Silva

Professor Auxiliar do
Departamento de Engenharias
Escola de Ciências e Tecnologia
da Universidade de Trás-os-Montes e Alto Douro

Carlos Manuel José Alves Serôdio

Professor Associado com Agregação do
Departamento de Engenharias
Escola de Ciências e Tecnologia
da Universidade de Trás-os-Montes e Alto Douro

Chatbot para Ajuda de Novos Alunos

João Rafael da Silva Carvalho

Submetido na Universidade de Trás-os-Montes e Alto Douro
para o preenchimento dos requisitos parciais para obtenção do grau de
Mestre em Engenharia Eletrotécnica e de Computadores

Resumo — Na Universidade de Trás-os-Montes e Alto Douro (UTAD), no início de cada ano letivo, existe um aumento do tempo de resposta dos serviços da universidade porque, os funcionários desses serviços, apesar de capazes, sofrem uma enorme sobrecarga no seu trabalho. A principal razão é a grande afluência de alunos recém-chegados de diversas origens. Por isso, para ser possível responder ao crescente aumento de alunos, aliviando os funcionários de perguntas de caráter mais simples, surgiu a necessidade de criar um sistema autónomo de respostas. Desta forma, para simular a intervenção humana o melhor possível foi desenvolvido um *chatbot*, que funciona na plataforma Facebook Messenger capaz de conversar em duas línguas, português e inglês. Como caso de estudo, foram postas em prática técnicas de Processamento de Linguagem Natural e *Machine learning*, nas quais se aplicam redes neuronais para a classificação da frase recebida e extração de uma intenção. A extração de uma intenção permite selecionar a resposta adequada para a frase recebida. Implementaram-se, no total, seis redes neuronais para cada língua, e, por sua vez, cada rede neuronal foi treinada num conjunto de dados diferente, correspondente a um serviço da universidade. Testaram-se os valores da precisão e os valores da função de perda de todos os modelos criados. De seguida, considerando vários parâmetros das redes neuronais fixos, alterou-se a função de ativação no neurónio de forma a encontrar a solução mais vantajosa, desta forma, utilizaram-se duas funções de ativação diferentes: a função de ativação de tangente hiperbólica e ReLU. Destes testes, conclui-se que, nos dados relativos à língua portuguesa, a função que apresentou melhor desempenho foi a função ReLU, na qual a precisão mínima obtida nos dados de validação foi de 96%. Enquanto que nos dados relativos à língua inglesa, a função tangente hiperbólica apresentou melhor desempenho nos dados de validação, com uma precisão mínima obtida de 88%.

Palavras Chave: ***Chatbot***, **NLP**, ***Machine learning***, **Redes neuronais, Classificação de texto**

Chatbot to Help New Students

João Rafael da Silva Carvalho

Submitted to the University of Trás-os-Montes and Alto Douro
in partial fulfillment of the requirements for the degree of
Master of Science in Electrical Engineering and Computers

Abstract — In the Universidade de Trás-os-Montes e Alto Douro, in the beggining of each school year, there is an increase in the time of response from the main services of the university because, the employees of these services, although capable, suffer a huge overload in their work. The main reason of this is the great influx of students from diverse origins. Because of this, in order to respond to the growing number of students without a significant increase in staff, a need has arisen to create an autonomous system to answer more simple questions. Thus, to simulate human intervention as best as possible, a chatbot has been developed, which works on the Facebook Messenger platform capable of chatting in two languages, Portuguese and English. As a case of study, are implemented techniques from Natural Language Processing and Machine learning, where neural networks are applied for phrase classification and intention extraction. The extraction of the intent allows to select the right response for the received phrase. A total of 6 neural networks were implemented for each language, where each neural network was trained on a different data set, each corresponding to a different service of the university. Considering several fixed parameters, the activation function in the neurons was changed in order to find a better solution, thus, two activation functions were used, hyperbolic tangent and ReLU. From these tests it can be concluded that in the Portuguese data, the function that presented the best performance was the ReLU function, where the minimum precision value obtained in the validation dataset was 96 %, while in the English data, the tanh function presented the best performance, with a minimum accuracy value in the validation dataset of 88 %.

Key Words: *Chatbot*, NLP, *Machine learning*, Neural Networks, Text Classification

Agradecimentos

A realização desta dissertação mostrou ser uma experiência enriquecedora para o meu desenvolvimento, todavia, nada disto era possível sem um apoio incondicional de algumas pessoas.

Ao Prof. Doutor Pedro Miguel Mestre Alves da Silva e ao Prof. Doutor Carlos Manuel José Alves Serôdio, um enorme agradecimento, pela amizade e companheirismo que mostraram ao longo do meu percurso académico, pelas inúmeras conversas e pela orientação pessoal e profissional.

Agradeço à minha namorada, Sofia Teixeira Guedes pelo apoio incondicional, pela amizade e pelo carinho que demonstrou durante todo este percurso.

Agradeço aos meus grandes amigos, António Pereira, Carlos Cruz, Inês Silva, Luís Marinheira, Pedro Alves e Renato Silva, por todo o apoio, amizade, risos e todos os momentos que partilharam comigo.

Agradeço à Fátima Campos, pelo auxílio, tempo e ajuda durante a elaboração desta dissertação.

Por último, um agradecimento imensurável à minha mãe, Elisabete Carvalho, ao meu pai, João Manuel Carvalho e ao meu irmão Ari Carvalho, por todo o sacrifício, amor, apoio e paciência que permitiu formar a pessoa que sou hoje. A todos, um sincero obrigado!

UTAD, João Carvalho

Vila Real, Outubro de 2019

Índice geral

Índice de tabelas

Índice de figuras

Glossário, acrónimos e abreviaturas

Glossário de termos

Lista de acrónimos

Sigla	Expansão
AI	*Artificial Intelligence*
ANN	*Artificial Neural Networks*
API	*Application Programming Interface*
BOW	*Bag of Words*
CNTK	*The Microsoft Cognitive Toolkit*
CNN	*Convolutional Neural Network*
GRU	*Gated Recurrent Unit*
HTTPS	*Hyper Text Transfer Protocol Secure*
LSTM	*Long Short-Term Memory*
NLP	*Natural Language Processing*
PSID	*Page-scopep Identification Number*
ReLU	*Rectified Linear Unit*

Sigla	**Expansão**
RNN	*Recurrent Neural Network*
TF-IDF	*Term Frequency–Inverse Document Frequency*
URL	*Uniform Resource Locator*

1 Introdução

Neste trabalho pretende-se estudar, projetar e implementar um *chatbot* capaz de responder de forma autónoma a perguntas colocadas pelos alunos da Universidade de Trás-os-Montes e Alto Douro (UTAD) e, desta forma, permitir que os funcionários não tenham de dispensar o seu tempo com estas questões simples cuja resposta passa a ser automatizada. Assim, neste capítulo são descritas as seguintes secções: na secção 1.1, são abordados a motivação que levou ao desenvolvimento da solução apresentada nesta dissertação e os dados estatísticos que tiveram influência nas decisões tomadas. Na segunda secção, 1.2, são expostos os principais objetivos a ser cumpridos com a dissertação. Por fim, na terceira secção, 1.3, são numerados e introduzidos os diferentes capítulos que são abordados.

1.1 Motivação

A tecnologia está cada vez mais presente nas vidas das pessoas e, é notório o crescimento assistido nos últimos anos de aplicações e serviços informáticos que facilitam a vida destas. As gerações mais novas cresceram sob a influência da tecnologia, o que as predispõe a usá-la mais facilmente. Assim, os novos alunos da universidade

naturalmente recorrem a este tipo de soluções para os auxiliar em várias questões. Em 2017, o número de alunos na UTAD ultrapassava os 6.600, em concreto, 6.651 alunos (UTAD, 2017). Já em 2018, verificou-se um grande aumento na entrada de alunos, pois, a UTAD foi a instituição de ensino superior que alcançou o maior aumento do número de estudantes a nível nacional, na primeira fase, e foi ainda a instituição localizada em regiões de menor pressão demográfica com a maior taxa de ocupação de vagas (UTAD, 2018). Mais uma vez, em 2019, verificou-se um aumento de 3%, que se traduz em 1.339 estudantes, que representa uma ocupação de 92% das vagas (UTAD, 2019). Este aumento do número de alunos tem provocado a constante congestionamento nos serviços principais da universidade, porque provoca uma sobrecarga de trabalho sobre os funcionários desses serviços e, como consequência, é fornecido um serviço aos seus alunos com margem de ser melhorado. Nota-se, também, que grande parte destes alunos apenas precisam de informações de caráter simples, como localização de serviços e horários. Aumentar o número de funcionários é uma opção que pode ser dispendiosa, assim, é necessário criar uma solução para este problema, com o menor custo possível, criando um sistema autónomo capaz de responder a questões colocadas, mas, ao mesmo tempo, proporcionar uma rápida resposta a toda a gente que se encontre no campus da UTAD, disponível 24 horas por dia, 7 dias por semana. Além disso, o sistema deve ser apelativo e de fácil interação.

Surgiu, por isso, a hipótese de criar um agente de conversação, conhecido frequentemente por *Chatbot*, pois oferece uma resposta às necessidades apresentadas e permite uma forma intuitiva de interagir com os utilizadores. Esta solução permite uma maior rapidez de resposta, em qualquer local do Campus, sem a necessidade de aumentar o número de funcionários disponíveis. O *Chatbot* será dotado de inteligência artificial e comunicará através de uma plataforma de *messaging*. Para isso utilizam-se duas áreas da computação: Processamento De Linguagem Natural e Redes Neuronais. O uso de Processamento de Linguagem Natural permite recolher certas características e elementos fundamentais de frases, focando-se no que é essencial para a sua classificação. As redes neuronais são utilizadas para classificar a frase e, desta forma, apreender a intenção do utilizador.

Selecionar uma plataforma de *messaging* em prol de uma aplicação foi a escolha natural, pois, como se observa na Figura 1.1, o uso destas plataformas continua a aumentar e não exige que as pessoas instalem mais aplicações nos seus dispositivos, tornando o processo de comunicar e pedir informações mais simples, mais rápido e mais apelativo. A plataforma escolhida foi o Messenger, do Facebook, dado ser uma das plataformas mais populares, principalmente em Portugal, como se pode observar na Figura 1.2. Outra grande vantagem de utilizar uma plataforma de *messaging* é a de não forçar, por parte dos utilizadores, a instalação de uma nova aplicação.

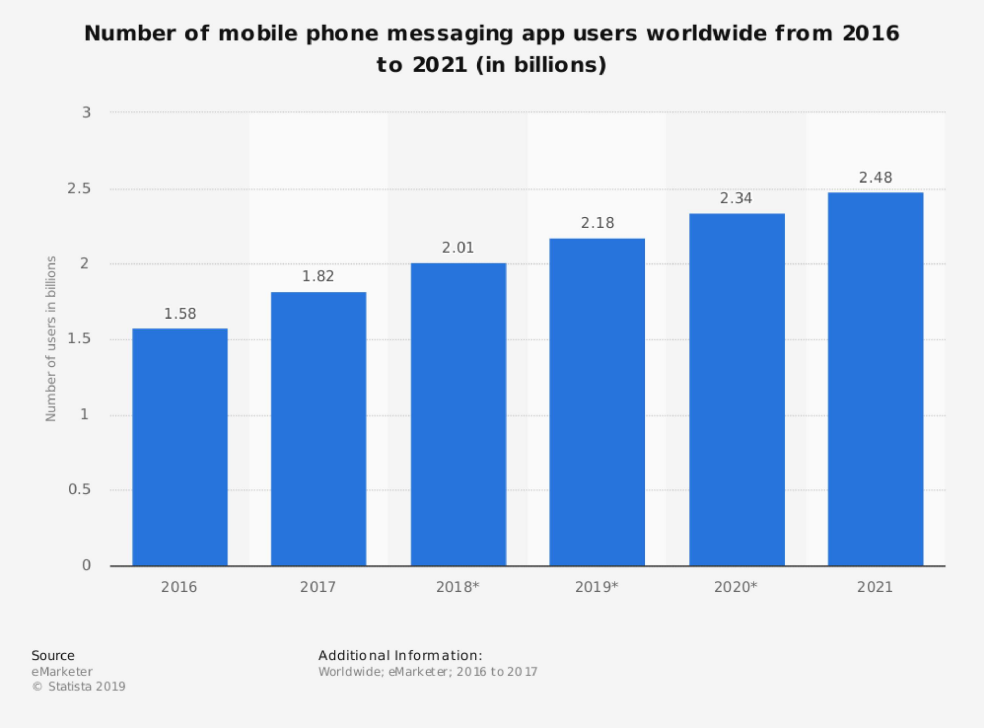

Figura 1.1 – Número utilizadores no mundo entre 2016 e 2021 (em biliões) de aplicações *messaging*. Retirado de statista (2019).

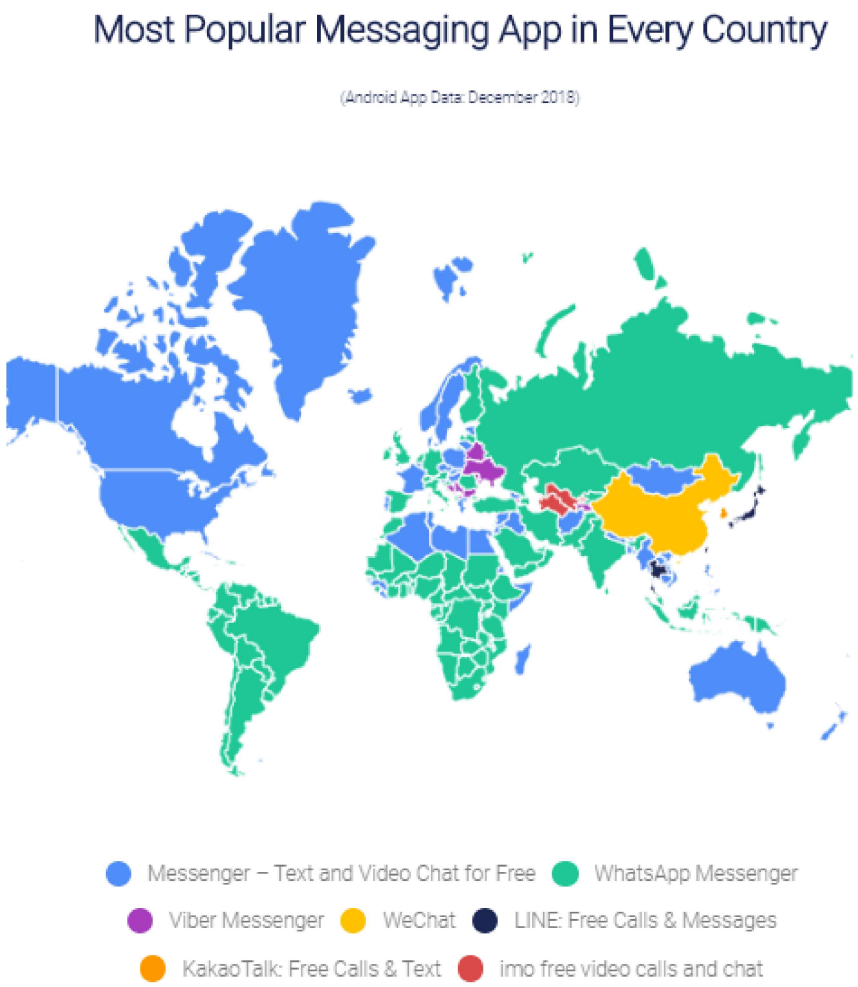

Figura 1.2 – Comparação da plataforma de *messaging* mais usada, por país. Retirado de SimilarWeb (2018).

1.2 Objetivos

O objetivo principal da dissertação é, portanto, implementar um *Chatbot* capaz de fornecer informações acerca da Universidade aos alunos, porém alguns objetivos mais específicos da dissertação são apresentados:

- Fornecer o horário correto dos Serviços de Ação Social, dos Serviços Académicos,

da cantina, da secretaria da Escola de Ciências e Tecnologia (ECT) e da biblioteca;

- Fornecer a ementa da cantina;
- Fornecer a localização dos Serviços de Ação Social, dos Serviços Académicos, da cantina, da secretaria da Escola de Ciências e Tecnologia (ECT) e da biblioteca, pedir a localização do utilizador e sugerir o caminho que o aluno pode escolher;
- Fornecer outras informações específicas a cada serviço;
- Funcionar na plataforma Facebook Messenger.

1.3 Organização da dissertação

Esta dissertação encontra-se organizada em seis capítulos. No Capítulo presente, foi realizada uma breve introdução ao trabalho, e foram expostos a motivação e objetivos desta dissertação.

No segundo capítulo, é feito um levantamento do estado da arte, em que são analisados alguns sistemas de conversação existentes. São, ainda, apresentados conceitos sobre os componentes, modelos de resposta, domínio e conversação de um *chatbot* e, por fim, são identificados e caracterizados sucintamente trabalhos relacionados com o tema desta dissertação.

É realizado o enquadramento tecnológico, no terceiro capítulo, onde se expõem as áreas científicas que serviram como base para o desenvolvimento do protótipo, como Processamento Linguagem Natural e *Machine Learning*. São apresentados conceitos específicos destas duas áreas, bem como, as ferramentas de *software* utilizadas.

No quarto capítulo, é feita uma descrição da conceção e implementação do *chatbot* que utiliza Processamento Linguagem Natural e *Machine Learning* para entender uma intenção presente na frase e responder corretamente. São explicadas todas as

etapas de criação dos modelos e manipulação de dados de treino, como se procedeu à configuração e à conexão da aplicação com o Facebook, à criação da base de dados capaz de armazenar a informação útil e relevante e, por fim, é descrito o funcionamento da aplicação.

No quinto capítulo, são apresentados e analisados os testes ao protótipo implementado e apresentados os resultados do desempenho dos modelos de *machine learning* criados.

Por último, no sexto capítulo, é feita uma análise final sobre os conteúdos abordados nesta dissertação, uma discussão dos resultados obtidos e, para concluir, são apresentadas algumas propostas de trabalho futuro.

2 Chatbots

Neste capítulo são abordados conceitos importantes acerca dos *chatbots*. Serão estudados a sua evolução ao longo dos anos, o seu estado corrente e a razão para a sua recente ascensão. De seguida, são apresentadas as diferentes abordagens tomadas para a sua conceção e arquiteturas e, por fim, são analisados trabalhos relacionados com o tema desta dissertação.

2.1 Agente de conversação

Um agente de conversação, ou do inglês *chatbot*, é definido pelo dicionário de Língua Portuguesa como: "Programa desenhado para simular uma conversa com utilizadores humanos, utilizado sobretudo em ambiente *online*" (Porto Editora, 2018).

Uma outra aceção do vocábulo é: Um *chatbot* é um agente conversacional que consegue interagir com humanos usando linguagem natural (Kuhn and Buitenhek, 2017). Deste modo, de acordo com estas aceções, um *chatbot* deve ser capaz de entender a conversa ou, pelo menos, dar a sensação ao utilizador que percebe o contexto da mesma e responder de acordo com esse conhecimento (Peters, 2018).

Para além das técnicas de Processamento de Linguagem Natural, é comum utilizar técnicas de inteligência artificial, que permitem o *chatbot* classificar as frases duma conversa, extrair informações relevantes para a compreensão destas frases e responder apropriadamente, ou seja, responder dentro do contexto da conversa (Kuyven et al., 2018).

Na sua forma mais básica, uma conversa pode ser vista como uma interação alternada, ou seja, o *chatbot* e a pessoa participam num diálogo. Assim, pode-se assumir, que um *chatbot* dependendo do tipo de diálogo em que se envolve, pode ser do tipo transacional ou conversacional (Mctear, 2018). Quando o *chatbot* é do tipo transacional, este é usado para alcançar um determinado objetivo, como agendar uma reunião ou criar um lembrete. Em comparação, um *chatbot* do tipo conversacional envolve-se numa conversa com o utilizador, sem um objetivo em mente, tornando-se crucial que este tipo de *chatbot* seja capaz de reconhecer o contexto da conversa e responder adequadamente ao utilizador (Silva, 2018).

Este trabalho tem como objetivo criar um agente com a capacidade de falar naturalmente com as pessoas, mas focado em alcançar um objetivo específico, por essa razão, o *chatbot* implementado é do tipo transacional.

2.1.1 História dos Chatbots

Será explorada nesta subsecção a evolução ocorrida ao longo dos últimos anos na área dos *chatbots*, desde o artigo de Alan Turing [1] até aos mais recentes avanços tecnológicos. Serão introduzidos diferentes *chatbots* por ordem cronológica do seu lançamento e referidas as suas importantes contribuições para este ramo de investigação.

[1] Alan Turing foi um matemático, lógico, criptoanalista e cientista britânico.

Teste de Turing

Em 1950, o matemático britânico Alan Turing, em resposta à sua pergunta, *"Can machines think?"* propôs um jogo, *Imitation Game* (Turing, 1950). Este é jogado por três pessoas, um homem (A), uma mulher (B) e um interrogador (C). O interrogador está numa sala diferente dos outros dois jogadores. O objetivo do jogo é o interrogador, através de um conjunto de perguntas, determinar qual dos sujeitos é o homem e qual é a mulher. O interrogador no final tem de chegar a uma conclusão: "A é homem"e "B é mulher". O interrogador pode colocar perguntas a ambos de forma a formular e fomentar a sua resposta. Da mesma forma, surge a pergunta, o que aconteceria se um computador substituir, por exemplo, o sujeito (A) neste jogo? Será que o interrogador conseguirá decidir qual a mulher e qual é o computador? Esta é a base para o famoso teste de Turing (Turing, 1950). Alan Turing previu que, no ano 2000, o poder computacional seria enorme, o que significava que um computador conseguiria enganar um interrogador comum durante 5 minutos, em cerca de 70% das vezes. Motivado por esta afirmação, em 1991, Dr. Hugh Loebner no Cambridge Centre for Behavioural studies iniciou uma competição intitulada *"Loebner Prize Competition"* que ocorre anualmente entre programas de computador. O objetivo é identificar os programas mais "humanos"e galardoar com 100.000 dólares o primeiro *chatbot* que passe num teste de Turing sem restrições (Mauldin, 1994).

O concurso é composto por quatro rondas, em cada ronda, os quatro jurados interagem com duas entidades. Uma é a pessoa e outra o *chatbot*. Depois de 25 minutos de perguntas, o júri tem de decidir qual entidade é a pessoa e qual é o *chatbot*. Em 1994, devido ao estado corrente dos programas em competição, foi usado o teste de Turing com restrições, no qual as conversas podiam estar limitadas a um tema (Mauldin, 1994). Porém, ainda em 2018, nenhum dos *chatbots* em competição conseguiu enganar o júri. Este posicionou-os de acordo com as suas respostas "humanas", ou seja, quão semelhantes eram as suas interações às do Homem (AISB, 2018).

Para além da competição, *"Loebner Prize Competition"*, o artigo de Turing motivou também a criação do primeiro *chatbot*, denominado de ELIZA, que demonstrou que um "simples programa"conseguia jogar o jogo da imitação, o que provocou um

interesse da comunidade científica da época (Mauldin, 1994).

ELIZA

ELIZA foi o primeiro *chatbot* criado (Kuhn and Buitenhek, 2017), desenvolvido em 1966 no Instituto de Tecnologia de Massachusetts por Joseph Weizenbaum (Peters, 2018). ELIZA não foi criada com a intenção de ser inteligente, mas de demonstrar o quão superficial as interações entre um humano e um computador podem ser (Kuhn and Buitenhek, 2017).

O programa simulava um psiquiatra e usava técnicas de processamento de linguagem natural para reformular o texto que o utilizador introduzia. A frase introduzida era analisada, as palavras-chave detetadas e, usando regras básicas, a frase era decomposta. Estas regras eram acionadas consoante as palavras-chave encontradas na frase introduzida, de seguida, as respostas eram geradas pelas regras de construção de frases associadas às regras de decomposição das frases (Weizenbaum, 1966). Apesar de ser relativamente simples, o programa foi bem sucedido em dar impressão que entendia os problemas do utilizador. Nas décadas seguintes, a abordagem dos *chatbots* foi semelhante à ELIZA, com apenas algumas pequenas diferenças (Peters, 2018), como exemplo, o programa pode ser observado na Figura 2.1.

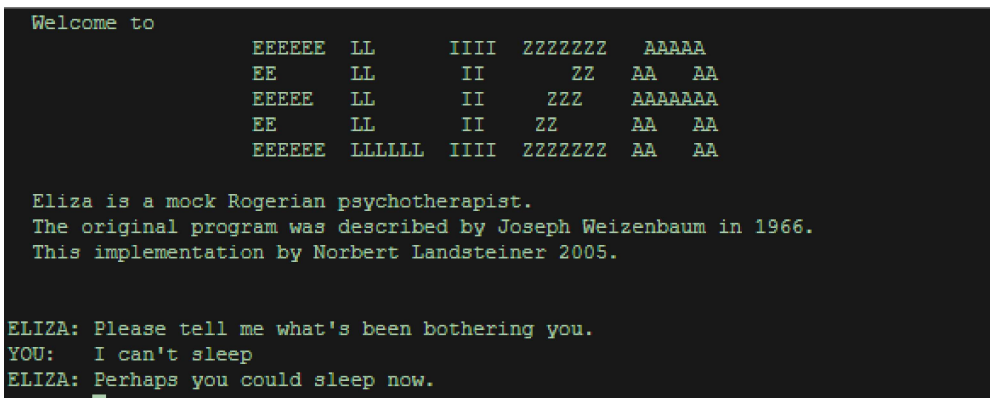

Figura 2.1 – Exemplo do *chatbot* ELIZA. Retirado de Landsteiner (2005).

SmarterChild

O *chatbot* SmarterChild surgiu em 2001, desenvolvido por ActiveBuddy, Inc., a atual Colloquis e operava no AOL Instant Messenger e MSN Messenger. Foi inspirado pelo aparecimento das plataformas de *texting* instantâneo, este *chatbot* permitia o acesso a notícias, a meteorologia, a resultados de desportos, entre outras informações.

A principal inovação de SmarterChild foi que este disponha de um conhecimento base e possuía informações sobre os seus utilizadores, o tornava as suas conversas, na perspetiva utilizadores, menos artificiais.

Watson AI project

Uma equipa na IBM em 2006 fez, também, um grande avanço neste domínio (Peters, 2018). O *chatbot* foi construído com o único objetivo de ganhar o programa americano *Jeopardy!*, do qual saiu vencedor em 2011 contra dois vencedores de edições anteriores. *Jeopardy!* é um concurso interessante do ponto de vista de Processamento Natural de Linguagem, pois as perguntas envolvem muito conhecimento sobre as palavras e uma rápida resposta em vários temas. Apesar dos avanços que teve, o agente ainda não conseguia ter uma conversa verosímil com uma pessoa, pois não conseguia manter o contexto da conversa, respondendo apenas à última pergunta recebida.

A ascensão das plataformas de *messaging*

A partir de 2010, assistiu-se à ascensão dos assistentes como a Siri da Apple, Cortana da Microsoft, Google Assistant da Google, Alexa da Amazon. Estes agentes trouxeram o conceito de conversação através de diálogo mas também de diálogo orientado para um fim específico. Mais recentemente, em 2016, o lançamento da plataforma de Messenger do Facebook permitiu a criação de inúmeros *chatbots* para todo o tipo de aplicações que, por consequência, provocou um aumento exponencial do mediatismo destes simuladores (Alves, 2018; Peters, 2018).

Atualmente tanto o Skype, o Slack e Telegram e WeChat oferecem suporte ao lançamento e implementação de *chatbots* nos seus serviços e é provável que esta lista continue a aumentar. Pois ao trazer os *chatbots* para locais onde as pessoas passavam mais tempo no telemóvel, a adesão sofreu um enorme crescimento (Alves, 2018). Por exemplo, na plataforma WeChat, existem *chatbots* para várias finalidades, como a simulação de uma namorada, tradução de frases, reconhecimento facial, entre outros (Graziani, 2016).

2.2 Arquitetura de um *chatbot*

Um *chatbot* pode ser idealizado de vários modos. Decidir a arquitetura de um *chatbot* é crucial para a sua implementação. Desta forma, nesta secção são abordados alguns conceitos importantes de um *chatbot*, como os componentes que o constitui, os diferentes modelos de resposta, os domínios em que pode ser inserido e, por fim, o tipo de conversa que tem com o utilizador.

2.2.1 Componentes de um *chatbot*

Não existem regras definidas sobre quais os componentes integrantes de um *chatbot*, dependendo, por isso, da implementação escolhida. No entanto, é possível generalizar a composição dos *chatbots* a três componentes principais: *Responder*, *Classifier* e *Graphmaster* (Alves, 2018; Stoner et al., 2004).

- ***Responder***: é o componente que serve como ponte entre a interface do utilizador e os componentes do *chatbot*, mais concretamente, tem a função de transferir informação proveniente da interface do utilizador para o *Classifier*.
- ***Classifier***: é a parte entre o *Responder* e o *Graphmaster*. Este componente tem as funções de filtrar, normalizar e segmentar os dados provenientes do *Responder* em componentes lógicos, transferir o resultado para o *Graphmaster*,

processar o resultado do *Graphmaster* e, por fim, manipular as instruções de sintaxe da base de dados.

- ***Graphmaster***: tem a função de organizar o conteúdo armazenado e guardar os algoritmos de correspondência de padrões.

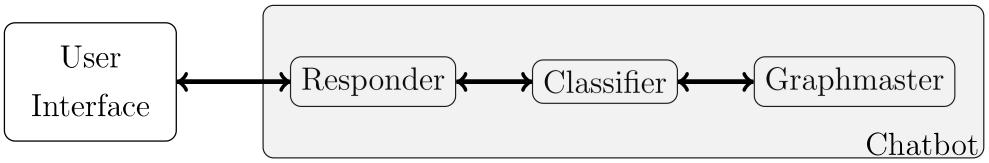

Figura 2.2 – Componentes de um *chatbot*. Adaptado de Stoner et al. (2004)

2.2.2 Modelos de respostas

Para implementar um *chatbot* é necessário escolher um modelo no qual se irão basear as suas respostas. Existem dois tipos de modelos que afetam a arquitetura de um *chatbot*, o modelo baseado no retorno de respostas e o modelo baseado na geração de respostas:

- **Baseado no retorno de respostas:** neste tipo de modelo, as respostas dadas pelo *chatbot* são predefinidas. A resposta é escolhida de entre um conjunto de respostas. A escolha desta é baseada num sistema de classificação no qual a resposta ou respostas com maior classificação são as escolhidas (Wu et al., 2016). Assim, através do uso de inteligência artificial, pode ser possível determinar uma intenção do utilizador e desta forma atribuir uma classificação às respostas.

- **Baseado na geração de respostas:** estes modelos geram as respostas de raiz, não precisando de um conjunto de respostas predefinidas (Silva, 2018). Estes métodos usam técnicas de geração de linguagem natural, para responder a uma mensagem (Wu et al., 2016). Desta forma, podem ter uma conversa mais aprofundada com o utilizador. É mais complexo que o modelo baseado em retorno de respostas e mais suscetível a erros gramaticais. Outra grande

desvantagem é estes sistemas necessitarem de um maior conjunto de dados de treino, o que, por vezes, não é possível (Silva, 2018).

2.2.3 Domínio

Outro aspeto a considerar é o domínio em que estarão inseridas as respostas do *chatbot*. Existem dois tipos de domínios que um *chatbot* pode estar inserido: domínio aberto e domínio fechado (Silva, 2018).

- **Domínio fechado:** Das duas possibilidades é a mais simples, pois o *chatbot* pode ser treinado com um conjunto de entradas e saídas fixas, porque o âmbito da conversa é limitado tal como o conhecimento exigido ao *chatbot*. É evidente que o utilizador pode encaminhar a conversa para a direção que deseja, porém não é expectável que o sistema lide com estes casos.

- **Domínio aberto:** Como o nome indica, neste tipo domínio, o utilizador pode encaminhar a conversa para o tema que desejar. É esperado que o *chatbot* entenda o fluidez da conversa e seja capaz de manter o seu contexto. É, por isso, muito difícil treinar um sistema como este, devido à infinidade de possibilidades e tópicos que este tem que ser capaz de lidar.

Com base na informação anterior, pode combinar-se os modelos com os domínios e antever 4 tipos de *chatbots* (Kuhn and Buitenhek, 2017), exemplificados na figura 2.3.

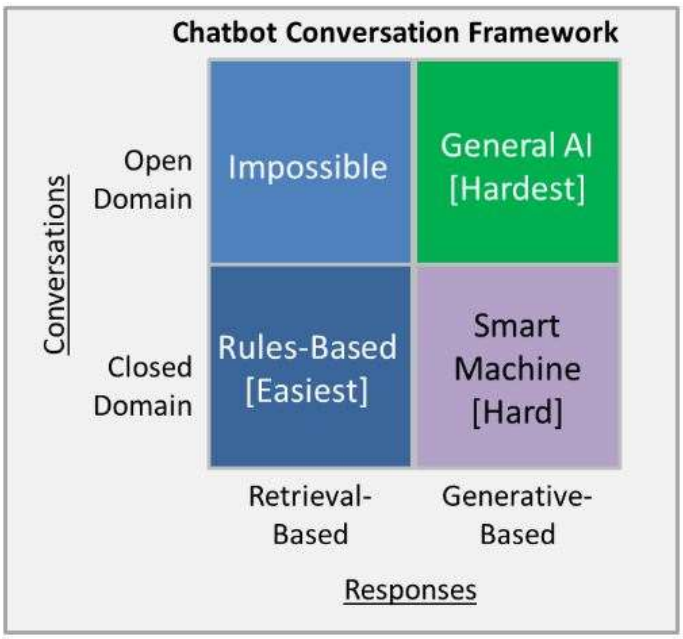

Figura 2.3 – Diagrama dos tipos de *chatbots*. Retirado de Clark (2016).

- **Domínio aberto e de retorno de respostas:** Atualmente, este *chatbot*, é, na prática, impossível, pois um *chatbot* baseado no retorno de respostas tem um conjunto fixo de respostas, num domínio aberto, o utilizador pode orientar a conversa para qualquer tópico que pretenda, daí ser impraticável ter uma resposta para cada pergunta possível que possa ser feita.

- **Domínio aberto e de geração de respostas:** Neste tipo de *chatbot*, a pessoa tem de ser capaz de perguntar qualquer possível questão e receber uma resposta adequada. Isto implica que o *chatbot* seja dotado de um nível de inteligência significativo, para, com êxito, interagir com o outro interveniente da conversa.

- **Domínio fechado e de retorno de respostas:** Sendo de domínio fechado, este tipo de *chatbot* pode ser treinado num conjunto fixo de dados, que é específico ao domínio em questão. As questões serão respondidas com respostas fixas. Evidentemente respostas fora do contexto ou domínio não serão respondidas. Este é o tipo mais implementado, por exemplo, nas empresas, onde na situação em que o *chatbot* não tenha uma resposta para a pergunta pode

ocorrer a intervenção de um humano.

- **Domínio fechado e de geração de respostas:** O *chatbot* deve ser capaz de gerar respostas dentro do domínio em questão. Ser dotado de capacidade de geração de respostas sobre um determinado domínio torna-o mais flexível, conseguindo responder a perguntas desconhecidas. Outra vantagem visível é conseguirem lidar com interações longas (mais que uma pergunta e uma resposta), proporcionando interações naturais com o utilizador. As desvantagens deste tipo são as mesmas que afetam qualquer tipo de *chatbot* baseado na geração de respostas, ou seja, a necessidade de grandes quantidades de dados de treino, o que muitas vezes não existe, e além disso a propensão para a ocorrência de erros gramaticais.

2.2.4 Conversação

As pessoas participam em conversas desde muito pequenas e, por isso, existe a assunção que projetar um *chatbot* é uma tarefa simples. Apesar dos recentes avanços dos *chatbots*, estes apresentam, ainda, grandes dificuldades a exibir naturalidade e fluência numa conversa. As principais dificuldades dos *chatbots* está em lidar com questões seguidas e relacionadas, entradas inesperadas e mudanças de tópico. Assim, é importante saber como planear uma conversa, de tal forma que, o *chatbot* seja capaz de receber e interpretar vários tipos de mensagens e fornecer respostas adequadas que mantêm a conversa coerente. Existem, atualmente, dois tipos principais de interações que um utilizador tem com um *chatbot*: *One-shot queries* e *Slot-filling dialogues* (Mctear, 2018):

- ***One-shot queries:*** A conversação é iniciada pelo utilizador e a iniciativa também é controlada por este. Tomam a forma de um único par, entrada-saída, onde o utilizador pergunta algo ou emite um comando, por exemplo:

  Utilizador: Onde é a secretaria?

  Utilizador: Mudar idioma

  Geralmente, não existem questões consecutivas por parte do utilizador e o

sistema não tem iniciativa, ou seja, não questiona, mesmo quando não é claro para ele o que o utilizador pretende.

- ***Slot-filling dialogues:*** Nestes diálogos, existe iniciativa por parte do sistema. A interação é controlada pelo sistema, no exemplo seguinte, observa-se como este recolhe informação de forma a responder corretamente ao utilizador:

  Sistema: Para onde queres voar?

  Utilizador: Madrid.

  Sistema: Em que data queres viajar?

  Utilizador: Próxima sexta

O exemplo anterior retrata o estado da maioria dos sistemas atuais, porém, existe outro tipo de interação é a ***Mixed-initiative*** onde ambos, o sistema e o utilizador, podem tomar a iniciativa da interação (Mctear, 2018).

2.3 Trabalhos Relacionados

Nesta dissertação, o *chatbot* a ser desenvolvido, é baseado no retorno de respostas, porque, o propósito do *chatbot* é fornecer informações úteis, daí ser essencial haver controlo nas respostas dadas por este. O *chatbot* irá operar num domínio fechado de conhecimento, esse domínio abrange os serviços da UTAD como os Serviços de Ação Social, os Serviços Académicos, a cantina, a secretaria da ECT e a biblioteca. De forma a facilitar a escrita, a partir deste momento usar-se-á apenas a palavra secretaria em vez das palavras secretaria da ECT.

A operação num domínio fechado, o retorno de respostas, o uso de processamento natural de linguagem e a classificação da frase introduzida pelo utilizador de forma a retirar uma intenção são, todos, fatores considerados para a escolha dos trabalhos apresentados.

2.3.1 *Interactive Bot to Support the Use of the UPTEC Intranet*

Em Silva (2018) foi desenvolvido um *chatbot* para uma nova plataforma de gestão do Parque de Ciência e Tecnologia da Universidade do Porto (UPTEC), denominada UPTEConnect. A plataforma permite a gestão de todas as empresas, colaboradores, pedidos de cartões de acesso, assistência e outros. Os autores procuraram resolver a dificuldade dos utilizadores de encontrar ferramentas que estes precisam, bem como, todos os *work-flows* associados a cada ação possível de ser executada na plataforma.

O *chatbot* proposto, com inteligência artificial, reconhece as entradas dos utilizadores, ou seja, possui capacidades de reconhecimento natural de linguagem e consegue guiar o utilizador corretamente. Para manipular e classificar as frases introduzidas pelos utilizadores, o autor utilizou Dialogflow, uma plataforma da Google. Esta plataforma recebe a frase que o utilizador introduz, manipula-a e prevê uma intenção. Na Figura 2.4 está representada a arquitetura do *chatbot* desenvolvido, por Silva (2018).

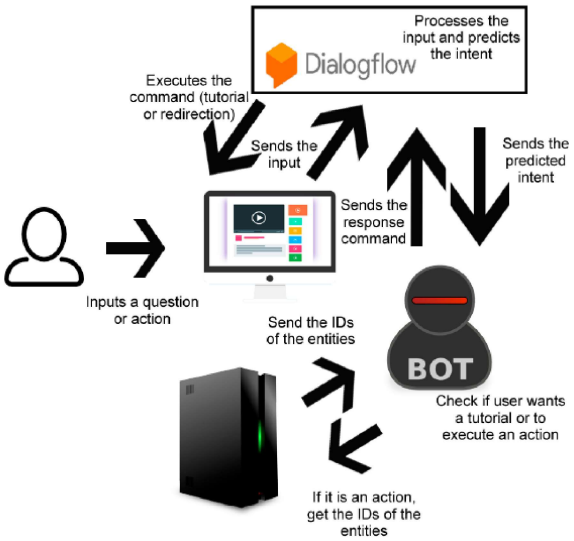

Figura 2.4 – Diagrama da interação da plataforma desenvolvida. Retirado de Silva (2018).

De forma a testar a eficácia do seu protótipo, foi realizado um inquérito a duas

pessoas que testaram o *chatbot*. Ambas concordaram que o *chatbot* entendeu as suas questões e que foi bastante útil quando mostrou o tutorial introdutório, assim que, um utilizador efetuou *login* pela primeira vez. Na Figura 2.5 é apresentado um exemplo de uma interação com o *chatbot*.

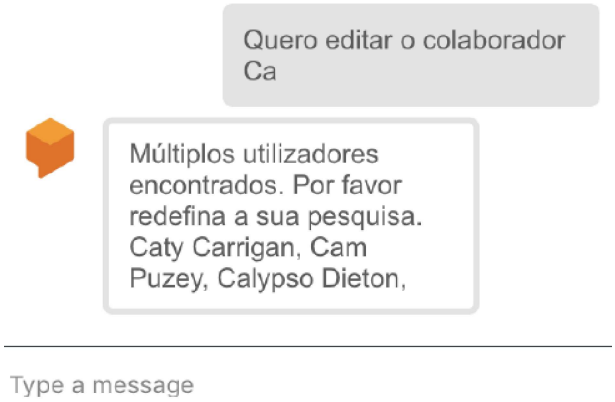

Figura 2.5 – Interação com o *chatbot*. Retirado de Silva (2018).

Por fim, os autores concluiram que o *chatbot* foi capaz de distinguir os dois principais objetivos do utilizador, no qual, um é o utilizador querer aprender algo e o outro é de executar uma ação. Os autores afirmaram, assim, que usando ferramentas já disponíveis, é possível criar um *chatbot* capaz de classificar as intenções do utilizador. Porém, quanto aos resultados obtidos sobre o desempenho do sistema são pouco explícitos, porque apenas duas pessoas foram interrogadas e, dessa forma, os resultados dos testes de satisfação são insuficientes. Algumas propostas de melhoria por parte dos autores foram: Adicionar novas intenções e adicionar o suporte a outros idiomas.

2.3.2 Chatbot Para Serviços Bancários

Em Alves (2018) implementou-se um *chatbot* para responder a diversos serviços bancários, desde a criação de uma conta até à realização de transferências ou pagamentos. Teve como objetivo resolver o problema da necessidade de atendimento físico, que pode ser demorado, apresentar gastos desnecessários em serviço de suporte

e obrigatoriedade de ter trabalhadores em funções que podem perfeitamente ser realizadas por esta tecnologia.

Foram estudadas três abordagens de implementação, porém o autor chegou à conclusão que a escolha mais adequada seria utilizar a API LUIS, da Microsoft, para classificar todas as frases e descobrir a intenção e os valores-chave das mesmas.

A primeira abordagem foi a utilização de uma biblioteca (*Chatterbot*) em conjunto com um *dataset* de diálogos. Mas com esta solução não foi possível classificar uma frase e extrair uma intenção, pois não era capaz de perceber significados nas frases e determinar, com mais exatidão, o que o utilizador pretendia.

Na segunda abordagem, são criadas manualmente as intenções e classificadas através de ferramentas de processamento linguagem natural, usando a biblioteca NLTK (Loper and Bird, 2019). Posteriormente, foram criadas várias listas, com frases associadas à *tag*, que corresponde à intenção e respetivas respostas. Também este método não apresentou bons resultados, visto que o processo de classificação de intenções não era satisfatório, pois era necessário que se detetasse a *tag* ou uma palavra das frases definidas para que o programa entendesse a intenção da frase. A solução final recaiu sobre o uso da API LUIS. A Figura 2.6 ilustra a abordagem final tomada. Foi criado conjunto de intenções e entidades que posteriormente foi utilizado para classificar as frases usadas na conversa.

De forma a testar esta solução, os autores criaram 10 frases de diálogo centradas em serviços bancários e observaram que o *score* mínimo obtido foi de 0.7767221, o que, na conclusão dos autores, provou ser um grande sucesso. Um exemplo de um diálogo como *chatbot* pode ser observado na Figura 2.7. Por fim, os autores indicaram algumas melhorias, como: integrar o *chatbot* numa plataforma de *messaging*, como o Facebook Messenger, aumentar o conjunto de dados e sofisticar o processamento natural de linguagem.

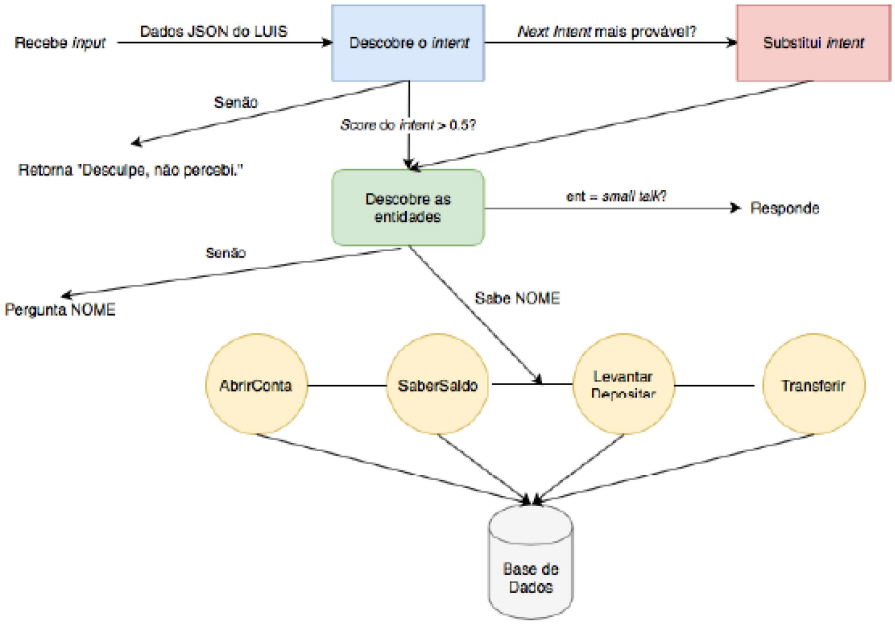

Figura 2.6 – Diagrama da interação da plataforma desenvolvida. Retirado de Alves (2018).

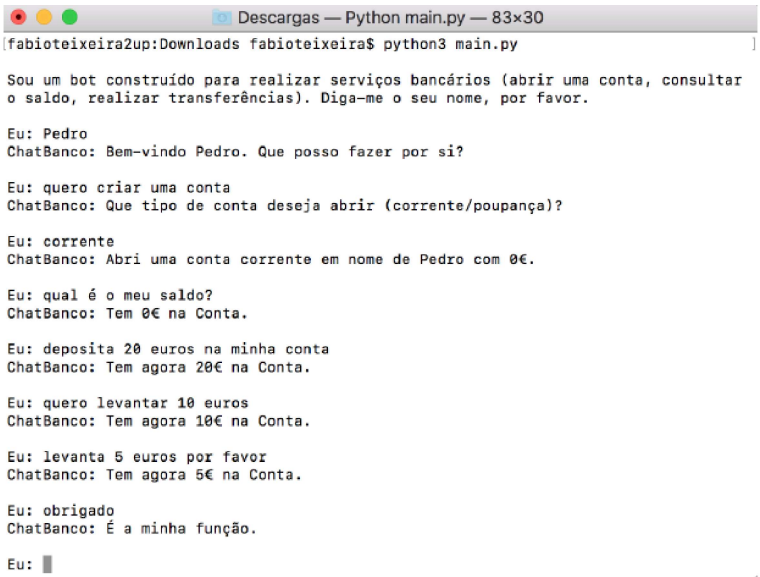

Figura 2.7 – Exemplo de um diálogo bancário com o *chatbot*. Retirado de Alves (2018).

A abordagem em estudo nesta dissertação será diferente das anteriores. Nem o Dialogflow nem LUIS serão utilizados, pois apesar de permitirem uma implementação rápida, não permitem o grau de detalhe que se procura neste trabalho. Porém,

como observado nos trabalhos anteriores, as outras abordagens tomadas para a classificação e extração da intenção não foram suficientes, por isso propõe-se a utilização de redes neuronais aliadas à utilização de técnicas de processamento linguagem natural para concretizar a classificação e extração da intenção.

2.3.3 *Design and Implementation of a Chatbot in the Context of Costumer Support*

No trabalho de Peters (2018), destaca-se o problema de haver falta de automatização no suporte aos utilizadores de plataformas online de jogos. Um *chatbot* para melhorar o suporte ao cliente foi proposto.

São analisadas várias estruturas de redes neuronais para a classificação da intenção do utilizador e, caso seja necessário, modelos para solicitar uma pessoa.

O *chatbot* recorre a redes neuronais, mais precisamente a *Recurrent Neural Networks*, ou *RNNs*, para classificar a intenção do utilizador.

Para responder aos utilizadores, os autores optaram por uma abordagem de retorno de respostas. Várias arquiteturas para a rede neuronal foram testadas como o uso de uma única camada de *Long Short-term Memory*, ou LSTM, uma única camada *Gated Recurrent Unit*, ou GRU, uma camada GRU com entrada invertida e duas camadas GRU. Foi escolhida uma única camada GRU com a entrada invertida que apresentou uma perda de validação de 0.49 e uma precisão de 88%. Na Figura 2.8 está representado o diagrama com a arquitetura final do *chatbot*.

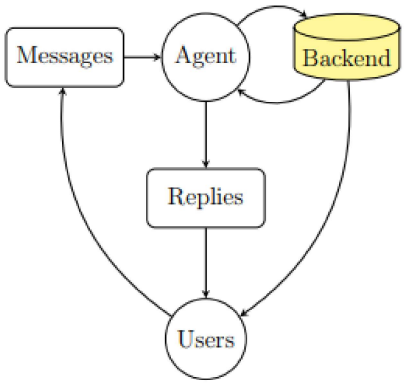

Figura 2.8 – Diagrama da interação da plataforma desenvolvida. Retirado de Peters (2018).

Para permitir o treino das redes neuronais, foram extraídos cerca de 1,400,000 conversas entre agentes de serviço ao consumidor e clientes. No total, estavam presentes nove idiomas, porém, os autores apenas consideraram os três idiomas mais numerosos: francês, inglês e holandês. As conversas em francês, são as mais numerosas, com cerca de 660,000 exemplos. As conversas em holandês com 222,000 e as conversas em inglês com cerca de 12,000. Os modelos de redes neuronais criados conseguiram atingir uma precisão de 87.6% no *dataset* francês, 81.6% no *dataset* holandês e 70.9% no *dataset* inglês. Na Figura 2.9 pode-se observar uma resposta do *chatbot* em francês, na qual, o utilizador afirma que o seu registo foi recusado. O *chatbot* identifica corretamente o problema e pede uma cópia do cartão de identificação, bem como outras informações. Na Figura 2.10 apresenta-se uma resposta do *chatbot* em inglês, na qual, o *chatbot* indentifica o problema do utilizador, procedendo à alteração da *password*. Na Figura 2.11 é apresentado um exemplo de uma resposta do *chatbot* em holandês. Neste exemplo, o utilizador pergunta para o seu e-mail ser alterado e, por isso, o *chatbot* solicita uma cópia do cartão de identificação e por um e-mail novo.

USER : Je voulais m'inscrire sur votre site mais l'inscription est refusée.
Merci !

CHATBOT : Bonjour User,

Afin de résoudre votre problème d'inscription, veuillez nous fournir
une copie de votre carte d'identité ainsi que vos coordonnées exactes
(adresse e-mail en particulier).

Bien à vous,
Bot

Figura 2.9 – Excerto de uma conversa, em francês, com o *chatbot*. Retirado de Peters (2018).

USER : I forgot my password and can't login.

CHATBOT : Hello User,

Your password has been successfully reset, your new password is :
<password>

All the best,
Bot

Figura 2.10 – Excerto de uma conversa, em inglês, com o *chatbot*. Retirado de Peters (2018).

USER : Hallo, ik wil mijn email veranderen alstublieft.

CHATBOT : Goedendag User,

Geef ons een kopie van uw identiteitskaart en uw nieuwe e-mailadres
om uw e-mailadres te wijzigen.

Het bestje,
Bot

Figura 2.11 – Excerto de uma conversa, em holandês, com o *chatbot*. Retirado de Peters (2018).

Para finalizar, os autores apontam algumas possíveis melhorais: o *chatbot* abranger mais idiomas, conseguir tratar de problemas mais complexos, estar disponível nos serviços de *messaging*, como o Facebook Messenger, ser capaz de relacionar as

perguntas consecutivas e, por fim, reencaminhar o utilizador para um interveniente humano, caso o *chatbot* não seja capaz de responder à pergunta feita.

2.3.4 *Designing a Dutch Financial Chatbot*

Em Kuhn and Buitenhek (2017) destaca-se a necessidade de um sistema autónomo de resposta para a plataforma Kandoor. Kandoor é uma plataforma que oferece recursos de aprendizagem na área financeira de forma gratuita. Desta forma, o *chatbot* desenvolvido, tem como objetivo responder a questões complexas, que normalmente requerem respostas complexas, aplicado ao domínio financeiro.

Os autores estudaram três implementações, a primeira consistiu num conjunto de técnicas de *Natural Language Processing*, ou NLP, e Term Frequency–Inverse Document Frequency, ou TF-IDF, para ponderar as palavras presentes nas perguntas e criar vetores para conseguir comparar a semelhança entre a pergunta recebida e uma pergunta presente num conjunto fixo (pergunta-resposta). Quando encontrada a maior similaridade, a resposta correspondente à pergunta encontrada no conjunto fixo é retornada ao utilizador.

A segunda implementação foi um modelo *Machine Learning* treinado num conjunto (pergunta - pergunta), para classificar e atribuir um *score* de semelhança entre duas questões. Quanto maior o *score*, mais semelhantes são. Assim, o *score* é o nível de confiança do modelo na classificação realizada. Os modelos testados foram *XGBoost*, *Neural Networks*, *Random Forest* e *Support Vector Machines*. Por fim, é retornada a pergunta com o mais alto nível de confiança. O modelo utilizado foi XGBoost, pois foi o que teve melhor desempenho, apresentando a mais alta precisão, um *F-score* de 0.79 e um AUC de 0.88.

A terceira implementação é uma combinação das duas anteriores e foi a escolhida pelos autores. Na Figura 2.12 está representado o diagrama da terceira implementação. Como observado no diagrama, C_l indica o nível de confiança da correspondência entre a questão recebida e a questão presente no *dataset*. Se o nível de confiança é alto, uma resposta retirada do conjunto (Questão-Resposta) é dada pelo *chatbot*. Pelo

contrário, um nível de confiança baixo, leva a que a questão recebida seja enviada a um especialista.

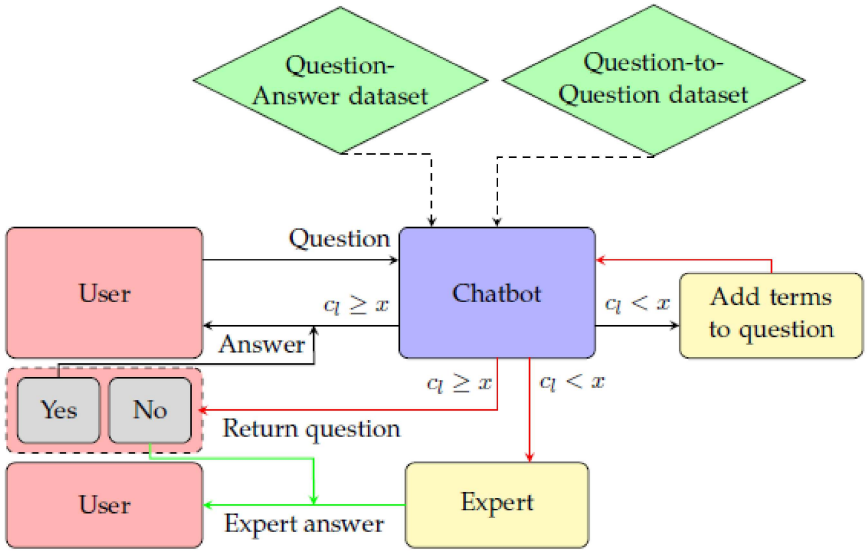

Figura 2.12 – Diagrama da implementação do *chatbot*. Retirado de Kuhn and Buitenhek (2017).

O melhor resultado obtido pelos autores foi um *chatbot* capaz de responder a 23% das questões colocadas pelos utilizadores, em que a restante percentagem é redirecionada para um especialista. Das respostas dadas pelo *chatbot* 74% foram respondidas corretamente. Foi concluído que as respostas dadas pelo *chatbot* não apresentavam grande qualidade. Devido à pequena dimensão do conjunto (Questão-Resposta), não havia resposta para muitas questões. Apesar de os autores optarem por escolher uma combinação da primeira implementação e da segunda, concluíram que esta combinação não ofereceu necessariamente melhores resultados que a primeira implementação.

Os autores por fim, recomendaram que um conjunto de (Questões-Respostas) maior fosse usado no futuro, adicionar mais funcionalidades ao *chatbot*, um melhoramento do processamento de linguagem natural para a língua holandesa fosse feito e também tornar a resposta do *chatbot* mais rápida, pois o *chatbot* pode demorar até 15 segundos até dar uma resposta.

3 Enquadramento tecnológico

Como referido anteriormente, o objetivo de um *chatbot* é estabelecer uma "conversação"coerente de modo a responder corretamente ao utilizador, para isso, é necessário classificar a frase introduzida pelo utilizador, retirar uma intenção e dar uma resposta adequada.

Porém, antes da classificação, é efetuada uma etapa de pré-processamento, pois foi verificado que a aplicação de pré-processamento aumenta o desempenho da classificação (Haddi et al., 2013). Pré-processar uma frase significa representá-la de forma estruturada para, ser possível, a sua classificação. De forma a representar as frases em dados estruturados, foram usados conceitos de processamento de linguagem natural. O classificador é baseado em *Machine Learning*, mais propriamente em redes neuronais. Por isso, neste capítulo, será feita uma introdução à *Machine Learning* e também será explicado o bloco básico de uma rede neuronal (neurónio), bem como os diferentes tipos de redes neuronais. Para completar o capítulo, será apresentado uma breve explicação de todas as técnicas e tecnologias utilizadas para o desenvolvimento do protótipo.

3.1 Processamento de Linguagem Natural

Processamento de Linguagem Natural (NLP do inglês *Natural language processing*) é um conjunto de técnicas computacionais para analisar e representar textos. O objetivo final é um processamento de linguagem semelhante ao do humano para uma série de tarefas e aplicações (Liddy, 2001). O NLP tem em consideração a hierarquia presente numa linguagem, ou seja, tem em consideração que caracteres formam palavras e que palavras formam frases.

É referido como um problema difícil na área da ciência da computação, devido a lidar com a ambiguidade das línguas humanas. Assim, o processo de compreensão de uma língua pode ser dividido em três etapas: Análise Morfológica, Análise Sintática e Análise Semântica (Kuhn and Buitenhek, 2017). Nesta secção, são abordadas estas três etapas e também o problema de classificação de texto.

3.1.1 Análise Morfológica

A Análise Morfológica foca-se no estudo da composição das palavras. Estas são compostas por morfemas, que são unidades mínimas capazes de expressar significado. Por exemplo, existem três morfemas na palavra infelicidade, o prefixo "in", a raiz "feliz"e o sufixo "dade". Por serem unidades mínimas, os morfemas não podem ser fragmentados, desta forma, o seu significado permanece o mesmo em diferentes palavras. Os humanos conseguem dissecar uma palavra desconhecida nos seus morfemas e assimilar o seu significado, de forma semelhante, um sistema NLP consegue reconhecer o sentido de cada morfema de forma a representar o significado da palavra (Liddy, 2001). Alguns termos importantes desta etapa são:

- ***Tokenization***: Processo de segmentação de um texto em símbolos, palavras, frases ou outros elementos textuais designados *tokens*. Normalmente, nas línguas nos quais as palavras estão separadas por espaços e pontuação, como na língua portuguesa ou outras línguas de origem europeia, a *Tokenization* é considerada uma fase relativamente fácil do processamento de texto, no

qual não são necessários métodos complexos. Um desafio maior para a *Tokenization* aparece nas línguas orientais, porque não apresentam espaços entre as palavras. A dificuldade é amplificada pelo facto de, nestas línguas, muitos caracteres representarem uma palavra e juntando-se a outros caracteres podem formar outras palavras. Naturalmente, nestas línguas, o processo de *Tokenization* é mais complexo, sendo necessários métodos mais sofisticados dos usados em línguas ocidentais (Mikheev, 2003). O resultado da *Tokenization* é uma lista de *tokens* que serve como dados de entrada para os passos seguintes de NLP (Kuhn and Buitenhek, 2017).

- **Remoção de *stop-words***: As *stop-words*, são palavras que não adicionam informação, como artigos definidos e indefinidos, preposições e conjunções. Por exemplo, na língua portuguesa, são palavras como "a", "o", "que", já na língua inglesa, "*a*", "*the*", "*are*", entre outras. Estas palavras ocorrem com grande frequência em textos, porém não desempenham um papel relevante na tarefa de classificação de texto, pois não apresentam potencial para distinguir as diferentes categorias (Silva and Ribeiro, 2003).

- ***Stemming***: processo de redução das palavras com a mesma raiz para uma forma comum, geralmente retirando os seus sufixos. Por exemplo, as palavras, "marítimo", "marinheiro" e "marujo" podem ser reduzidas à sua raiz comum, "mar" (Lovins, 1968). Esta técnica é utilizada para facilitar o processo de correspondência entre documentos de texto com o mesmo conteúdo. A pesquisa no campo de recuperação de informação mostra que a raiz das palavras funciona bem como uma unidade de representação e, para muitas tarefas, a ordem das palavras pode ser ignorada sem existir perda significativa de informação (Joachims, 1999). Lovins (1968) descreveu o primeiro algoritmo de *stemming* desenvolvido especificamente para aplicações de retorno de informação. Em 1980, Porter (1980) introduziu o algoritmo Porter, desenvolvido para a lingua inglesa, para melhorar o desempenho da tarefa de retorno de informação, utilizando uma lista de sufixos. Mas tarde, foi introduzido o Snowball (Porter, 2001), que permitiu o *stemming* noutras línguas, como

românticas (português e francês), germânicas (alemão e holandês), línguas escandinavas (norueguês e dinamarquês) e russo.

3.1.2 Análise Sintática

Na Análise Sintática o foco é descobrir uma estrutura gramatical presente na frase, analisando as palavras que a compõem. As sequências de palavras são, por isso, transformadas em estruturas que mostram como cada palavra na sequência está relacionada com as outras. O resultado desta análise é, assim, uma representação da frase, que revela as relações de dependência estrutural entre as palavras (Liddy, 2001). Alguns termos importantes nesta etapa são: *Bag-of-words* e *Term Frequency-Inverse Document Frequency* (TF-IDF).

- ***Bag-of-Words***: O modelo *Bag-of-Words*, ou BOW, é uma representação simples de textos usada em NLP e apenas representa as palavras do texto e a sua multiplicidade. É aplicado frequentemente em tarefas de classificação de texto (Kuhn and Buitenhek, 2017). Neste modelo, um texto é representado como uma coleção desordenada das suas palavras, não considerando a sua ordem nem a gramática (George K and Joseph, 2014). Deste modo, é criado um vetor, denominado dicionário, com um índice para cada palavra que ocorre no texto, tendo como valor a representação do número de vezes que a palavra ocorre na frase (Silva and Ribeiro, 2003). Por exemplo, considere-se as frases seguintes:

  A: "O sol é uma estrela. O sol é bonito."
  B: "A lua é um satélite."

  O dicionário destas duas frases é construído da seguinte forma:

  ["O", "sol", "é", "uma", "estrela", "bonito", "Lua", "Satélite"]

  Juntando as duas frases existem, no total, oito palavras diferentes, são essas palavras que compõem o dicionário de palavras. De seguida, cada frase é representada como um vetor de oito elementos, que corresponde ao tamanho do dicionário:

A1: [2,2,2,1,1,1,0,0]
B1: [1,0,1,1,0,0,1,1]

Cada índice dos vetores A1 e B1 indica o número de vezes que a palavra de índice correspondente no dicionário de palavras aparece na frase, ou seja, por exemplo, no vetor A1, a palavra "sol", que corresponde à posição de índice 1 (considerando que o primeiro índice do vetor é 0), aparece 2 vezes na frase A. Por outro lado, a palavra "estrela", que corresponde ao índice 5 no dicionário de palavras, aparece uma vez na frase A, por consequência, no vetor A1 o índice 5 terá o valor de 1. De forma semelhante, no vetor B1, a palavra "lua"que corresponde ao índice 7 no dicionário de palavras aparece uma vez na frase B, assim, no vetor B1, o índice número 7 apresenta valor de 1.

- ***Term Frequency-Inverse Document Frequency* (TF-IDF)**: TF-IDF é uma estatística que reflete a importância de uma palavra de um documento em relação a uma coleção de documentos (Kuhn and Buitenhek, 2017). TF-IDF é uma combinação de *Term Frequency* (TF) e *Inverse Document Frequency* (IDF). TF refere-se à frequência de ocorrência de um termo num documento Rebala et al. (2019). É fácil assumir que quanto mais alta a frequência de ocorrência do termo, mais alta a relevância do documento. Porém, em documentos grandes (quando comparados com outros documentos), um termo pode aparecer mais vezes pelo simples facto que o documento é grande e, desta forma, o termo aparece mais vezes. Para reduzir esta tendência é considerado, na equação, o comprimento do documento. A Equação de TF, 3.1, corresponde à razão entre o número de vezes que um termo ocorre num documento e o número total de palavras.

$$TF = \frac{Número \quad de \quad vezes \quad que \quad ocorre \quad um \quad termo}{Número \quad total \quad de \quad palavras} \tag{3.1}$$

Inverse Document Frequency identifica a frequência de ocorrência de um termo em todos os documentos. Esta análise ajuda a encontrar termos que são comuns em múltiplos documentos, ou seja, termos que não ajudam a distinguir os documentos. Tendo como exemplo a palavra "de", procurar documentos

baseando-se nesta palavra não traz benefícios quando comparada com a palavra "professor", que é muito mais útil para a análise de um documento. A Equação do IDF, 3.2, corresponde ao logaritmo da razão entre o número total de documentos e o número de documentos onde a palavra t é encontrada.

$$IDF = \log(\frac{Número \quad total \quad de \quad documentos}{Número \quad de \quad documentos \quad onde \quad t \quad é \quad encontrado}) \tag{3.2}$$

Juntando as duas equações, a Equação 3.3 mostra a fórmula do TF-IDF. Na Equação, t representa o termo que é utilizado na procura, d é o documento em que o termo está a ser procurado e D representa o número total de documentos.

$$TF - IDF = TF(t, d) * IDF(t, D) \tag{3.3}$$

Assim, um termo que aparece em todos os documentos é irrelevante para a separação dos documentos, porque terá um valor muito pequeno de TF-IDF. Pelo contrário, um termo que aparece apenas num documento apresenta um valor de TF-IDF alto (Kuhn and Buitenhek, 2017; Rebala et al., 2019).

3.1.3 Análise Semântica

A análise semântica foca-se em associar significado a palavras, frases e textos. São determinados os possíveis sentidos de uma frase, focando-se nas interações ao nível das palavras presentes na frase. Na Análise Semântica pode incluir-se a desambiguação semântica de palavras com múltiplos sentidos, ou seja, apenas um significado de palavras polissémicas é incluído na representação semântica da frase (Liddy, 2001).

- ***Named Entity Recognition***: É uma técnica que trata do reconhecimento de entidades e extração de sentido das palavras. Em NLP, reconhecer nomes ou entidades e colocá-los em categorias corretas, como, por exemplo, nomes de pessoas, nomes de organizações ou nomes de lugares é importante na extração do contexto do texto. Como exemplo, considere-se o nome "Tesla", que

pode ser reconhecido como o nome de uma pessoa, nome de uma empresa automóvel ou até uma unidade de medida da densidade de um campo magnético. Métodos utilizados nesta técnica são: métodos baseados em regras e métodos de inteligência artificial (Rebala et al., 2019).

- ***Word Sense Disambiguation***: Utilizada para determinar qual o sentido de uma palavra inserida no texto. Muitas palavras têm mais que um significado e, distingui-los pode ser difícil. Métodos utilizados nesta técnica são: métodos empíricos, métodos *knoledge-based* e métodos de inteligência artificial (Kuhn and Buitenhek, 2017).

3.1.4 Classificação de Texto

Em processamento de liguagem natural, uma tarefa fundamental é a classificação de texto.

Numa tarefa de classificação, é necessário produzir uma função que relacione uma determinada caraterística com uma classe pré-definida (Ikonomakis et al., 2005).

Considerando x como sendo um documento e y a sua correspondente classe, um exemplo de um par de treino é (x_i, y_i). O objetivo do processo de treino ou aprendizagem é de encontrar um h que prevê corretamente a classe $y = h(x)$ do documento x (Pang, 2018).

Como a programação de "força bruta"para classificar um texto é impraticável, pois não é possível classificar cada combinação de palavras que formam um documento, é preferível aprender classificadores através de exemplos e é também crucial que o classificador seja capaz de generalizar usando poucos dados de treino (Joachims, 1999). Cada documento pode pertencer a uma categoria, a nenhuma, ou até a múltiplas.

Em (Lilleberg and Yanqing Zhang, 2015) utilizou-se *word2vec* (Mikolov et al., 2013) em combinação com TF-IDF para determinar eficácia de *word2vec* na tarefa de classificação de texto, suportado na ideia que *word2vec* é capaz de fornecer mais

recursos semânticos que outras técnicas de *Word Embeddings*. Vários cenários foram testados, sendo que o melhor resultado foi um *score* de 0.897297 para a combinação de *word2vec* ponderado por *TF-IDF* sem *stop-words* + TF-IDF sem *stop-words*, Figura 3.1.

2 Different Categories of different topics	*Score*
Word2vec with stop words	0.841892
Tf-idf with stop words	0.894595
Word2vec weighted by tf-idf w/o stopwords	0.895946
Tf-idf without stop words	0.881081
Word2vec weighted by tf-idf w/o stop words + Tf-idf without stop words	**0.897297**

Figura 3.1 – Precisão das diferentes técnicas utilizando *word2vec* e TF-IDF. Retirado de Lilleberg and Yanqing Zhang (2015).

Em (Zhang et al., 2015) tratou-se o texto como um sinal ao nível dos caracteres e aplicou-se *convolutional networks*, denominadas de *ConvNets*, para extrair informação desse sinal. Concluiu-se que *ConvNets* apresentam melhor desempenho em *datasets* de vários milhões de exemplares, enquanto que métodos como TF-IDF e *N-grams* apresentam melhores resultados para *datasets* de dimensões menores (milhares de exemplares), porém, a conclusão mais importante foi que é possível utilizar *ConvNets* para a classificação de texto sem a necessidade de palavras. Estas conclusões comprovam que uma língua pode ser vista como um sinal, não sendo diferente de qualquer outro tipo de sinal.

3.2 *Machine Learning*

Machine Learning pode ser definida como sendo um conjunto de métodos computacionais que utilizam a experiência para melhorar o seu desempenho a fazer previsões corretas. A experiência, neste caso, refere-se à informação disponível anteriormente, que normalmente apresenta-se sob a forma de dados. Estes podem ser digitais,

manualmente etiquetados por pessoas, ou outros tipos de informações obtidas pelo sistema através da interação com o ambiente. A qualidade dos dados é extremamente importante para o sucesso do algoritmo em fazer as previsões corretas. Desta forma, *Machine Learning* consiste na conceção de algoritmos capazes de fazerem previsões exatas e eficientes. A complexidade espacial e temporal são, como noutras áreas da computação, duas formas de medir a qualidade destes algoritmos. Porém, a quantidade de dados necessários para o algoritmo aprender um conjunto de conceitos é também uma forma de medir a qualidade do algoritmo de *Machine Learning* (Mohri et al., 2018). Por estas razões, *Machine Learning* pode oferecer uma alternativa eficiente para os problemas que acarretam um desenvolvimento de soluções caras e demoradas, ou quando o problema é demasiado complexo para permitir o desenvolvimento de soluções com garantias de otimização (Simeone et al., 2018). Podem dividir-se os métodos de *Machine Learning* de várias formas. Uma forma de dividir os algoritmos de *Machine Learning* é através da saída que produzem, que pode ser Classificação, Regressão, *Clustering* (Kuhn and Buitenhek, 2017; Simeone et al., 2018):

- **Classificação**: Consiste em atribuir uma categoria a cada item. Por exemplo, a classificação de um documento, que pode atribuir a textos as categorias como política, desporto ou meteorologia, enquanto, a classificação de imagens, pode atribuir a imagens categorias como paisagem, retrato ou animal. Normalmente, o número de categorias é pequeno, porém não existe limite para o número de classes a serem atribuídas.

- **Regressão**: Consiste na previsão de valores reais. Um bom exemplo de um problema de Regressão são as previsões feitas na bolsa de valores. Neste tipo de problema, a penalização para uma previsão errada depende da magnitude da diferença entre os valores corretos e os valores previstos. É, por causa disso, diferente dos problemas de classificação, porque, nestes não existe a noção de proximidade entre as várias categorias.

- ***Clustering***: Consiste na divisão de itens em regiões homogéneas. *Clustering* é comummente utilizado para analisar conjuntos de dados muito grandes. Por

exemplo, na análise de estatística feita nas redes sociais, os algoritmos de *Clustering* tentam identificar comunidades com gostos semelhantes de entre grandes grupos de pessoas.

Outra forma de dividir os algoritmos de *Machine Learning* é através da forma de aprendizagem, que pode ser *Supervised learning*, *Unsupervised Learning* e *Reinforcement Learning* (Kuhn and Buitenhek, 2017).

- ***Supervised Learning***: É a forma mais comum de *Machine learning*. Consideremos imagens que contêm, por exemplo, uma casa, um carro ou uma pessoa. Primeiro, recolhe-se uma grande quantidade de imagens de casas, de carros e de pessoas. De seguida, cada imagem é etiquetada com a sua categoria correspondente, ou seja, são criadas categorias ou *tags*. Durante o treino, é exposta uma imagem ao sistema e este produz um vetor de classificações, um para cada categoria. O que se pretende é que a categoria desejada apresente a classificação mais alta, o que é altamente improvável acontecer antes de ocorrer o treino. É determinada uma função objetivo que mede o erro entre as classificações de saída e o padrão desejado de classificações. O sistema modifica o seu estado interno, alterando parâmetros internos para reduzir o erro. Estes parâmetros ajustáveis são também designados pesos (Lecun et al., 2015). Este tipo de aprendizagem é normalmente associado a problema de Classificação ou Regressão (Simeone et al., 2018).

- ***Unsupervised Learning***: O sistema recebe um conjunto de dados não etiquetados e faz previsões para todos os dados não observados durante o treino (Mohri et al., 2018). Existe, por isso, um conjunto de variáveis para um determinado número de observações, confiando ao algoritmo a capacidade de encontrar uma estrutura nos dados (Martins et al., 2018). É aplicado geralmente em *Clustering* e *Dimensionality Reduction*, ambas relacionadas com o problema de representar dados num espaço mais pequeno e, por isso, mais conveniente (Simeone et al., 2018).

- ***Reinforcement Learning***: *Reinforcement Learning* refere-se ao problema

de inferir uma sequência ótima de decisões, baseadas em recompensas ou penalizações, recebidas de ações anteriores. Ocorre uma interação do sistema com o ambiente e, em alguns casos, afeta o ambiente, recebendo uma recompensa ou penalização pela sua ação. Por exemplo, um sistema pode ser treinado para navegar num ambiente com obstáculos, penalizando as decisões que resultam em colisões (Simeone et al., 2018; Mohri et al., 2018).

.

3.2.1 Redes Neuronais

As Redes Neuronais Artificias (ANN, do inglês *Artificial Neural Network*), ou apenas redes neuronais como são frequentemente designadas, são vagamente inspiradas no modelo do cérebro humano (Pang, 2018) e são utilizadas numa variedade de tarefas de NLP, como tradução e síntese de textos (Rush et al., 2015).

O cérebro humano é uma estrutura complexa, composta por várias camadas de neurónios que pode conter entre 10 mil milhões e 1 trilião de neurónios (Herrup and Karl, 1988) sendo capaz de realizar tarefas com grande complexidade. Um exemplo é a visão humana que, na prática, é uma tarefa de processamento de informação. O sistema visual fornece uma representação do ambiente à nossa volta, que observamos através dos olhos e fornece informação que permite ao ser humano interagir com o ambiente (Simon Haykin, 2009). De forma semelhante, uma rede neuronal é composta por unidades de processamento simples, os neurónios artificiais. Apresenta uma tendência natural para armazenar conhecimento, obtido através da experiência, tornando-o disponível para o uso e, assemelha-se, por isso, ao cérebro humano em dois aspetos:

- Obtém conhecimento do ambiente através de um processo de aprendizagem;
- As forças das conexões entre neurónios, conhecidas como pesos sinápticos, são usadas para armazenar o conhecimento adquirido;

O algoritmo utilizado para executar o processo de aprendizagem é denominado de *learning algorithm*, cuja função é de modificar, de forma ordenada, os pesos das sinapses para se alcançar o objetivo desejado. A modificação dos pesos é o método tradicional para treinar as redes neuronais, porém, é possível treiná-las, alterando a sua própria topologia. Esta forma de treinar as redes neuronais é motivada pelo facto dos neurónios no cérebro humano morrerem e novas conexões são criadas.

As redes neuronais derivam o seu poder computacional, em primeiro lugar, da sua estrutura paralela e, em segundo lugar, da sua capacidade de aprender e generalizar, ou seja, a rede neuronal é capaz de produzir resultados ou saídas satisfatórias perante entradas não observadas durante o treino (Simon Haykin, 2009).

O neurónio

O neurónio é o constituinte estrutural do cérebro (Simon Haykin, 2009). Os neurónios biológicos comunicam através de sinais elétricos, que são impulsos rápidos ou "picos"na tensão da parede celular ou membrana. As conexões entre neurónios são mediadas por junções eletroquímicas, denominadas sinapses. Estas estão localizadas nas dendrites, que são ramificações do neurónio e atuam na receção de estímulos nervosos do ambiente ou de outros neurónios. Estes, geralmente, recebem milhares de conexões de outros neurónios, por isso, estão constantemente a receber uma multitude de sinais, que eventualmente alcançam o corpo da célula, denominado de soma. Aqui, os sinais são, de alguma forma, adicionados e de, um modo geral, se o sinal resultante exceder um determinado *threshold* o neurónio ativa-se, ou, por outras palavras, gera um impulso elétrico como resposta. Este impulso é transmitido para outros neurónios, através do axónio (Gurney, 2014). A Figura 3.2 apresenta um exemplo de neurónio biológico e artificial. A estrutura e funcionamento do neurónio artificial é inspirada no neurónio biológico (Andrej Krenker and Kos, 2011).

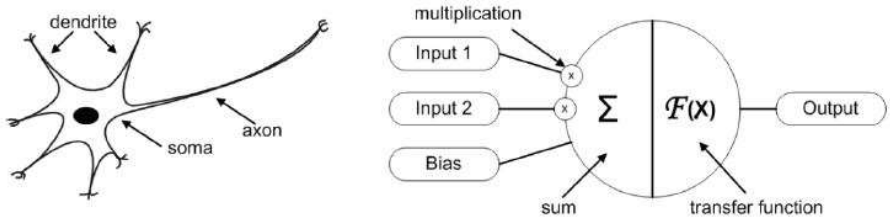

Figura 3.2 – Figura de um neurónio biológico (esquerda) e um neurónio artificial (direita). Retirado de Andrej Krenker and Kos (2011).

O princípio de funcionamento do neurónio está representado na Figura 3.3, na qual podemos identificar três elementos básicos que constituem o neurónio:

1. Um conjunto de sinapses, no qual, cada uma é caracterizada por um peso. Especificamente, um sinal x_j, na entrada da sinapse j, conectada ao neurónio k, é multiplicado pelo peso sináptico w_{kj}. Ao contrário dos pesos presentes numa sinapse do neurónio humano, o peso da sinapse num neurónio artificial pode assumir valores positivos e negativos.

2. Um somador, para adicionar os sinais de entrada, que foram multiplicados pelos seus respetivos pesos.

3. Uma função de ativação, para limitar a amplitude de saída do neurónio.

O neurónio inclui também uma tendência, designada por b_k, que é aplicada externamente. A tendência b_k tem o efeito de aumentar ou diminuir a entrada da função de ativação, dependendo, respetivamente, se a função é positiva ou negativa.

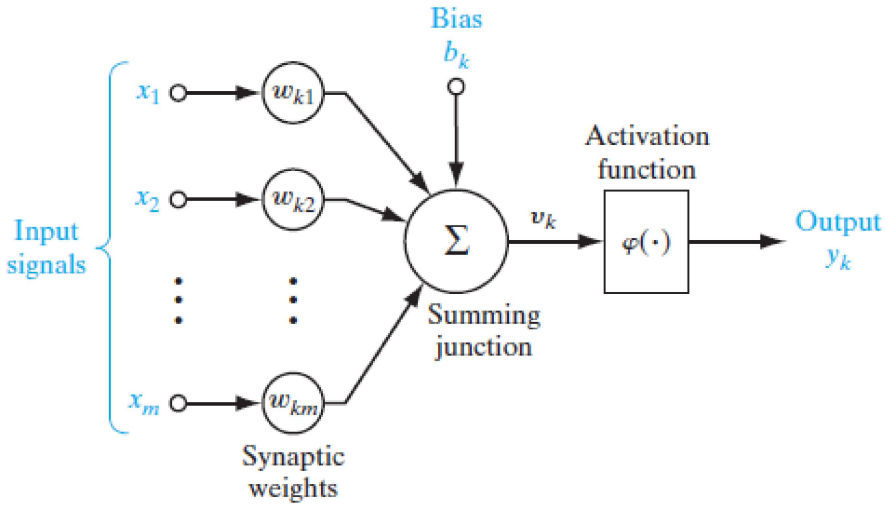

Figura 3.3 – Princípio de funcionamento de um neurónio artificial. Retirado de Simon Haykin (2009).

Em termos matemáticos, é possível descrever um neurónio, utilizando as equações: 3.4 e 3.5. A Equação 3.4 representa o somatório das multiplicações entre pesos e as entradas do neurónio. A Equação 3.5 define o sinal de saída do neurónio.

$$u_k = \sum_{j=1}^{m} w_{kj} \cdot x_j \tag{3.4}$$

e assim temos:

$$y_k = \varphi(u_k + b_k) \tag{3.5}$$

no qual, x_1, x_2, ... , x_m são sinais de entrada; w_{k_1}, w_{k_2}, ... , w_{k_m} são pesos sinápticos do neurónio k. Desta forma u_k é, por isso, uma combinação linear; b_k é a tendência; φ define a função de ativação; e y_k é o sinal de saída do neurónio (Simon Haykin, 2009).

Como exposto na Equação 3.5, uma incógnita no modelo do neurónio é a função de ativação ou função de transferência. Esta define as propriedades do neurónio e

pode ser qualquer função matemática (Andrej Krenker and Kos, 2011), que permite introduzir a não-linearidade característica das redes neuronais (Karlik and Olgac, 2011).

Assim, as funções de ativação são usadas para controlar a saída de um neurónio e podem ser lineares e não-lineares (Nwankpa et al., 2018). São apresentadas as três funções de ativação que desempenham um papel importante no desenvolvimento desta dissertação, pois são utilizadas na otimização das redes neuronais criadas.

- **Tangente Hiperbólica**: É uma função centrada em 0 (Nwankpa et al., 2018), cujos valores variam entre -1 e 1 e é definida pela Equação 3.6. A Figura 3.4 apresenta a função Tangente Hiperbólica.

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{3.6}$$

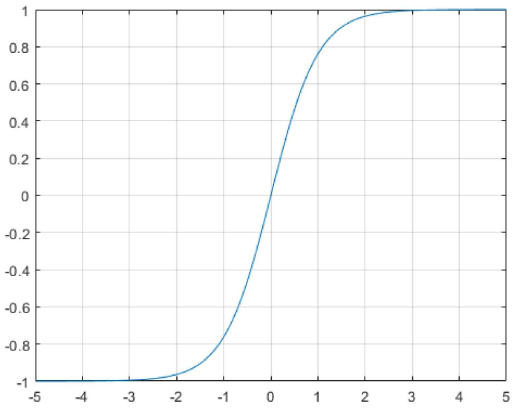

Figura 3.4 – Função Tangente Hiperbólica. Retirado de Mathworks (2019)

- **ReLU (Rectifier Linear Unit)**: Foi introduzida em Hahnloser et al. (2000). Quando $x < 0$ o seu valor é 0, contrariamente, quando $x \geqslant 0$, assume o valor de uma função linear, sendo definida pela Equação 3.7. A Figura 3.5 apresenta a função ReLU.

$$f(x) = max(0, x) = \begin{cases} x_i, & if \quad x_i \geq 0 \\ 0, & if \quad x_i < 0 \end{cases} \tag{3.7}$$

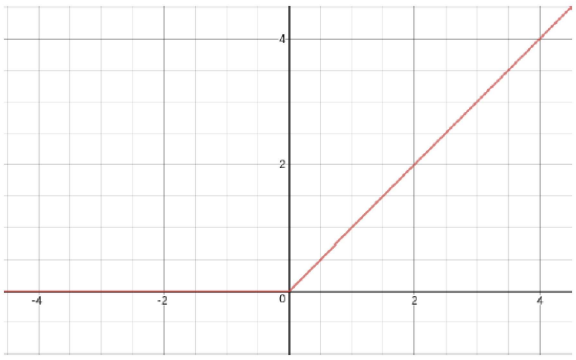

Figura 3.5 – Função ReLU. Retirado de Agarap (2018).

- **Softmax**: É outro tipo de função de ativação. É usada para calcular a distribuição de probabilidades a partir de um vetor de números reais. A função produz um vetor de probabilidades, que são valores reais entre 0 e 1, com a soma das probabilidades sendo igual a 1. É usada em problemas de classificação multi-classe, onde retorna a probabilidade de cada classe. É definida pela Equação 3.8.

$$f(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{3.8}$$

Uma rede neuronal pode conter um grande número de camadas. A camada de entrada tem tantos neurónios quantas variáveis independentes existem e a camada de saída tem tantos neurónios quantas variáveis dependentes existem.

A quantidade de camadas interiores e a quantidade de neurónios nas camadas escondidas depende do tipo e quantidade de dados. Quando uma rede neuronal é treinada, os pesos das ligações entre os neurónios são modificados até se atingir o modelo ideal. Quanto maior for a diferença entre o resultado que o modelo prevê

e os resultados reais, mais rapidamente são alterados os pesos das conexões (Kuhn and Buitenhek, 2017).

A representação matemática de uma rede neuronal simples pode ser explicada, utilizando como referência a Figura 3.6.

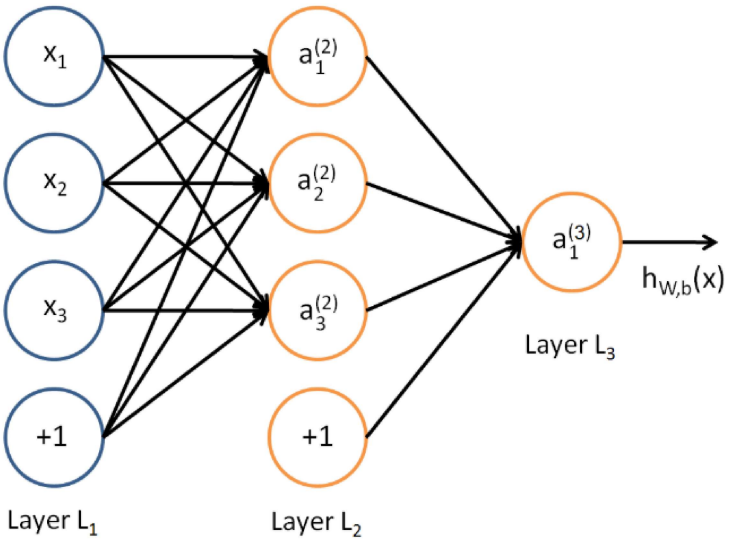

Figura 3.6 – Rede neuronal simples. Retirado de Ognjanovski (2019).

O modelo tem quatro nós na camada de entrada L_1, $[x_1, x_2, x_3, +1]$, quatro nós na camada interior L_2, $[a_1^{(2)}, a_2^{(2)}, a_3^{(2)}, +1]$, e um nó na camada de saída L_3, $[a_1^{(3)}]$ (Ognjanovski, 2019). De forma a entender a notação utilizada, é necessário referir que:

- $a_i^{(j)}$ corresponde à "ativação"da unidade i na camada j, ou seja, o valor de saída do neurónio i.
- $\theta^{(j)}$ corresponde à matriz dos pesos que controlam a função que define os valores da camada j para a camada $j+1$.

É de notar que os valores +1 na camada de entrada e na camada de saída correspondem às tendências, desta forma, podem ser marcadas com x_0 e a_o, respetivamente. Deste modo, pode-se observar o vetor da camada de entrada, X, na Equação 3.9 e o vetor da camada interior, A, na Equação 3.10.

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{3.9}$$

$$A = \begin{bmatrix} a_0^{(2)} \\ a_1^{(2)} \\ a_2^{(2)} \\ a_3^{(2)} \end{bmatrix} \tag{3.10}$$

Para calcular a dimensão das matrizes de pesos, considera-se que se a rede neuronal conter s_j unidades na camada j e s_{j+1} unidades na camada $j+1$, então, $\theta^{(j)}$ é de dimensão $s_{j+1} * (s_j + 1)$. Assim, a matriz dos pesos entre a camada de entrada L_1 e camada interior L_2 tem dimensão $3 * 4$, já a matriz dos pesos entre a camada de interior L_2 e a camada de saída L_3 tem dimensão $1 * 4$. A matriz de pesos, na camada de entrada, pode ser observada na Equação 3.11.

$$\theta^{(1)} = \begin{bmatrix} \theta_{10} & \theta_{11} & \theta_{11} & \theta_{13} \\ \theta_{20} & \theta_{21} & \theta_{22} & \theta_{23} \\ \theta_{30} & \theta_{31} & \theta_{32} & \theta_{33} \end{bmatrix} \tag{3.11}$$

Assim, para calcular os valores de saída da camada interior L_2 multiplica-se o vetor de entrada X com os pesos da matriz θ^1 e aplica-se a função de ativação. O resultado observa-se na Equação 3.12.

$$\begin{aligned} a_1^{(2)} &= g(\theta_{10}^{(1)} x_0 + \theta_{11}^{(1)} x_1 + \theta_{12}^{(1)} x_2 + \theta_{13}^{(1)} x_3) \\ a_2^{(2)} &= g(\theta_{20}^{(1)} x_0 + \theta_{21}^{(1)} x_1 + \theta_{22}^{(1)} x_2 + \theta_{23}^{(1)} x_3) \\ a_3^{(2)} &= g(\theta_{30}^{(1)} x_0 + \theta_{31}^{(1)} x_1 + \theta_{32}^{(1)} x_2 + \theta_{33}^{(1)} x_3) \end{aligned} \tag{3.12}$$

É feito o mesmo procedimento para os valores de saída da camada L_3, no qual se

multiplica $A * \theta$, obtendo-se a Equação 3.13.

$$h_\theta(x) = a_1^{(3)} = g(\theta_{10}^{(2)} a_0^{(2)} + \theta_{11}^{(2)} a_1^{(2)} + \theta_{12}^{(2)} a_2^{(2)} + \theta_{13}^{(2)} a_3^{(2)} \tag{3.13}$$

Vantagens e Desvantagens das Redes Neuronais

A aplicação de Redes Neuronais apresenta algumas vantagens (Svozil et al., 1997; Simon Haykin, 2009).

- As redes neuronais têm a capacidade de aprender, mesmo não tendo a assistência do utilizador.
- Podem ser não lineares. Um neurónio pode ser linear ou não linear. Consequentemente, uma rede neuronal, constituída por neurónios não lineares, é também não linear. A não linearidade é uma propriedade bastante importante se a relação entre a entrada e a saída é intrinsecamente não linear.
- Mapeiam pares entrada-saída. Em *Supervised Learning* cada exemplo consiste num único sinal de entrada e a sua correspondente saída. Um exemplo escolhido no *dataset* de treino é exibido à rede neuronal e os pesos sinápticos são modificados de forma a minimizar a diferença entre a saída desejada e a resposta efetiva da rede. O treino é repetido para vários exemplos no *dataset*, até que a rede atinja um estado estável, onde não ocorram mais mudanças significativas nos pesos sinápticos. Deste modo, a rede aprende com os exemplos, construindo um mapeamento entrada-saída para o problema específico.
- Apresentam grande adaptabilidade. As redes neuronais têm a capacidade de adaptar os seus pesos sinápticos às mudanças no ambiente circundante. Desta forma, uma rede neuronal, que opera num ambiente específico, pode ser treinada novamente para lidar com as pequenas mudanças nas condições operacionais desse ambiente. A arquitetura de uma rede neuronal para classificação de padrões, processamento de sinais e aplicações de controlo, em conjunto com a capacidade adaptativa da rede, é uma ferramenta útil na classificação de

padrões adaptativos, processamento de sinais adaptativos e controlo adaptativo.

- Exibem uma arquitetura paralela. A natureza paralela de uma rede neuronal torna-a potencialmente rápida em certas tarefas que exigem grande poder de processamento.

Apesar de inúmeras vantagens, as redes neuronais apresentam desvantagens (Cilimkovic, 2015; Vlad, 2005).

- Requerem tempo para treinar, que aumenta com o incremento da complexidade da rede neuronal e com o tamanho do conjunto de dados utilizado.
- As suas previsões são influenciadas pela qualidade e precisão dos dados de treino.
- Necessitam de grandes quantidades de dados para atingir bons resultados.
- É difícil justificar um resultado obtido de uma rede neuronal.
- Não existe uma metodologia para escolher, treinar e verificar o seu desempenho.

Tipos de redes neuronais

- ***Feed-Forward***: Numa rede neuronal *Feed-Forward*, a informação flui apenas numa direção, da camada de entrada para a camada de saída, passando pelas camadas interiores. A saída de uma camada serve de entrada para a camada seguinte (Goyal et al., 2018). Não existe qualquer tipo de limite para o número de camadas, tipo de função transferência usada no neurónio ou o número de conexões entre neurónios (Andrej Krenker and Kos, 2011). El Assaf et al. (2016) propuseram um algoritmo, contra a atenuação do sinal anisotrópico, induzida por *fading*, sombras, interferências, entre outras, presentes em qualquer canal de redes sem fios. O algorimo baseia-se numa abordagem de estimativa de

distância. Também desenvolveram um mecanismo de correção da estimativa da distância, baseado em redes neuronais *feed-forward*, que tem em consideração a atenuação anisotrópica do sinal. Os resultados obtidos mostram que o algoritmo proposto apresenta melhor desempenho que a maioria dos algoritmos utilizados atualmente para o mesmo fim, não apenas em precisão, mas também, em robustez contra a atenuação anisotrópica. Em Marquez et al. (2013) foi proposto um método híbrido que combina informação processada de imagens de satélite com ANNs para prever valores futuros de irradiação global horizontal em certos intervalos de tempo de 30, 60, 90 e 120 minutos. Obtiveram resultados semelhantes aos métodos que utilizam as abordagens convencionais, porém, os autores afirmaram que uma vantagem da sua implementação é a utilização de um processamento de imagem mais simples que as abordagens atuais.

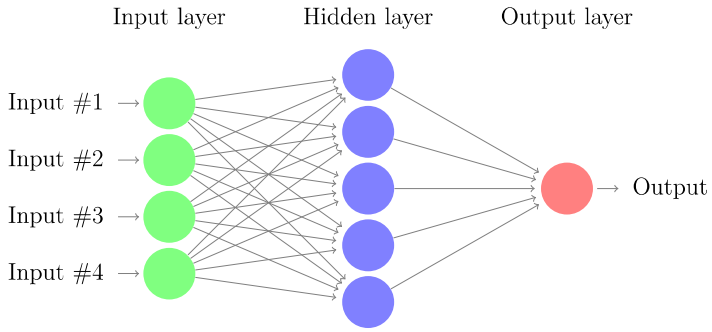

Figura 3.7 – Rede neuronal *Feed Forward*.

- ***Convolutional Neural Networks***: *Convolutional Neural Networks* (CNNs), são um tipo de rede neuronal, que utiliza a convolução, em vez da multiplicação de matrizes, em pelo menos uma camada da rede (Pang, 2018), porém são aplicadas em áreas como, reconhecimento de imagem e padrões, reconhecimento de voz, NLP e análise de vídeo (Hijazi et al., 2015). As CNNs estão bem adaptadas para tarefas como reconhecimento de imagem e reconhecimento de escrita manual, pois estas processam eficazmente matrizes de múltipla dimensão, por exemplo, uma imagem a cores composta por três matrizes de 2 dimensões, contendo as intensidades dos píxels nos 3 canais de cor (Lecun et al., 2015). A sua estrutura é baseada na amostragem de uma janela ou porção de uma

imagem, detetando as suas características, e usa-as para construir uma representação (Goyal et al., 2018).

Em Kim (2014) foi proposto utilizar CNN para a classificação de frases. Treinaram uma CNN simples com apenas uma camada de convolução para aprender vetores de palavras obtidos de (Google, 2013). Estes vetores foram treinados com 100 mil milhões de palavras oriundas do Google News. Apesar de pouco ajuste nos parâmetros da rede, o modelo foi capaz de apresentar excelentes resultados em múltiplos testes.

Hijazi et al. (2015) utilizaram CNNs para reconhecer sinais de trânsito alemães, atingindo o melhor resultado no *German Traffic Sign Recognition Benchmark*, ou GTSRB, que é um desafio de classificação de imagens de trânsito. Um exemplo de um bloco típico de uma CNN está representado na Figura 3.8.

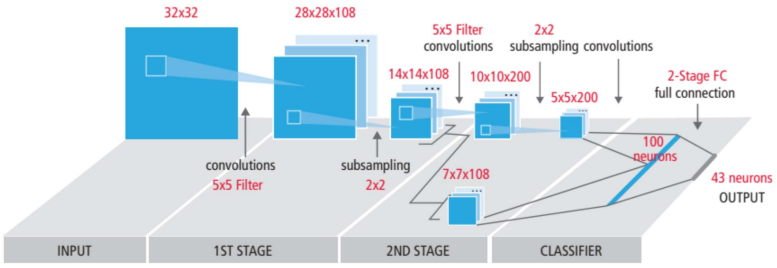

Figura 3.8 – Bloco típico de uma CNN. Retirado de Hijazi et al. (2015).

- ***Recurrent Neural Networks***: *Recurrent Neural Networks* ou RNN contêm pelo menos um *feedback loop*, onde a saída anterior no tempo T é reintroduzida como entrada no tempo $T+1$. São bastante utilizadas em aplicações que envolvem sequências de dados, como tarefas de processamento, em aplicações de vídeo (sequência de imagens) e para problemas de tradução de línguas, onde o significado da próxima palavra é baseada no contexto anterior (Goyal et al., 2018). As RNNs processam uma sequência de entrada um elemento de cada vez, mantendo nas suas unidades ocultas um "vetor de estado"que contém implicitamente informação sobre o histórico de eventos passados na sequência (Lecun et al., 2015). Sutskever et al. (2011) aplicaram RNNs para

prever o próximo carácter num fluxo de texto. Implementaram um tipo diferente de RNN, que introduz ligações multiplicativas (ou *gated*) que permitem o carácter de entrada atual determinar a matriz de pesos das camadas interiores. Constataram que esta nova arquitetura, utilizando RNNs, aprendeu de forma fácil as palavras. Gregor et al. (2015) aplicaram RNNs na geração de imagens. Introduziram uma nova arquitetura, *Deep Recurrent Attentive Writer*, que é composta por um par de RNNs: uma rede codificadora que comprime as imagens reais durante o treino; e uma rede descodificadora que reconstitui as imagens. A arquitetura implementada demonstrou ser capaz de gerar imagens altamente realistas como fotografias de números de casas. Na Figura 3.9 é apresentada a arquitetura típica de uma RNN. Como observado, os pesos são partilhados ao longo to tempo.

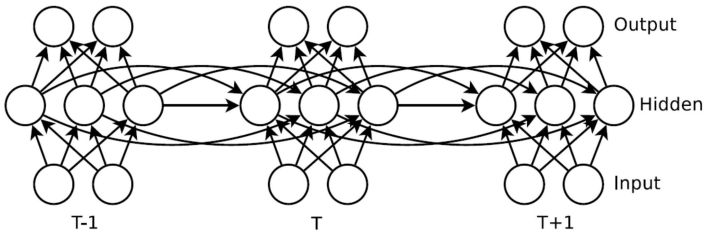

Figura 3.9 – Bloco exemplificativo de uma RNN. Retirado de Sutskever et al. (2011).

4 Conceção e Implementação

Para lidar com a necessidade de responder de forma automática a questões simples dos alunos, decidiu-se implementar um *chatbot* capaz de utilizar NLP e redes neuronais para classificar uma frase numa intenção, ou categoria e, desta forma, ser selecionada a resposta adequada para o aluno.

Neste capítulo, aborda-se a conceção e implementação do *chatbot*. Na conceção, explica-se a arquitetura e funcionalidade dos diversos módulos que compõem o sistema. Na implementação, descreve-se de forma detalhada como se procedeu para criar e interligar os diferentes módulos descritos na conceção.

Assim que o utilizador envia uma mensagem ao *chatbot*, este recebe-a e classifica-a. De seguida, responde adequadamente ao utilizador, ou seja, a cada mensagem recebida é escolhida uma nova resposta ou ação. Na Figura 4.1 pode observar-se um diagrama geral da estrutura do *chatbot*.

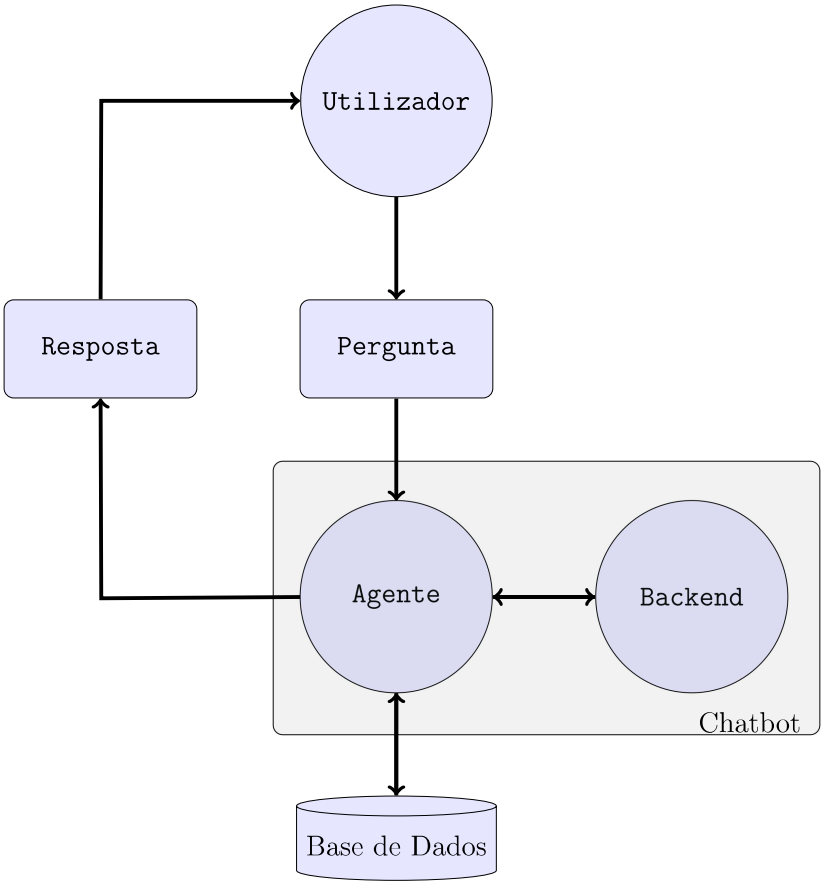

Figura 4.1 – Diagrama ilustrativo do funcionamento do *chatbot*

4.1 Conceção

Nesta secção é efetuada uma análise dos múltiplos componentes do sistema, onde se explica a função destes e como funcionam em conjunto. Em primeiro lugar, é abordado o tipo de mensagens que são esperadas receber pelo *chatbot*. De seguida, são expostos os componentes que fazem parte do módulo *Backend* e explica-se a função da base de dados utilizada. Para finalizar, é explicada a função do módulo Agente.

4.1.1 Mensagem

Antes de se apresentar e detalhar as funções do Agente ou do *Backend* do *chatbot*, será caracterizado o tipo de mensagens que serão recebidas por este. É importante referir que mensagens de diferentes tipos provocam ações e respostas diferentes do *chatbot*. Este recebe mensagens do tipo *Postback* ou *Message*. Uma mensagem do tipo *Message* pode ainda ser dividida em três subtipos: *Attachments*, *Text* e *Quick Replies* (Facebook, 2019b), conforme a Figura 4.2.

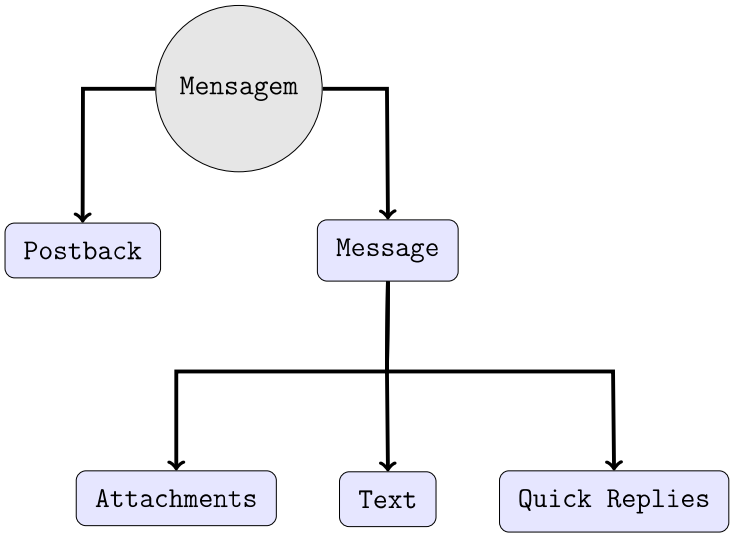

Figura 4.2 – Tipos de mensagens recebidos pelo *chatbot*.

Quando uma mensagem é inserida pelo utilizador, o primeiro passo é determinar de que tipo de mensagem se trata.

Uma *Postback* é recebida quando o utilizador envia um pedido para mudar o idioma de conversação, dado que o *chatbot* suporta dois idiomas.

Uma *Message* é recebida em várias ocasiões, deste modo, é necessário ainda determinar qual o seu subtipo:

- ***Attachments***: recebida quando o utilizador partilha uma localização com o *Chatbot* ou elementos não textuais, como imagens.

- ***Quick Replies***: recebida quando o utilizador envia uma mensagem pré-definida. Este subtipo de *message* é apenas recebido na primeira interação do utilizador com o *Chatbot*.
- ***Text***: recebida quando uma frase é enviada ao *Chatbot*. Só com este subtipo é que será iniciado o processo de classificação da frase para se retirar uma intenção.

4.1.2 *Backend*

O *Backend* tem como função receber a frase e classificá-la numa intenção. É possível dividir o *Backend* em duas etapas. O pré-processamento dos dados é a primeira, seguida da classificação da intenção. As etapas são caracterizadas de seguida:

Pré-processamento

O pré-processamento é composto por uma série de etapas de preparação da frase para a classificação.

1. Remoção de acentos e pontuação;
2. Substituição das letras maiúsculas por minúsculas;
3. *Tokenization*;
4. Remoção de *stop-words*;
5. Redução das palavras à sua raiz (*Stemming*);
6. Criação do *Bag-of-words*.

Em primeiro lugar, são removidos elementos da frase que não são importantes para a classificação, como acentos, pontuação e também se substitui as maiúsculas por minúsculas. Depois disto, é feita a *tokenization*. Este processo transforma a frase

num vetor e cada índice desse vetor contém uma palavra da frase. De seguida, faz-se a remoção das *stop-words* e a redução das palavras de interesse à sua raiz. Por fim, é criado um *Bag-of-words*, explicado na subsecção 3.1.2, que será utilizado para a classificação.

Classificação

Depois de se obter o *Bag-of-words*, dá-se início à etapa de classificação, cujas fases e sequência podem ser observadas na Figura 4.3. O vetor recebido é usado como entrada da rede neuronal e como resultado é obtido um vetor de probabilidades. Este é ordenado de forma descendente, sendo ignoradas as probabilidades cujo valor é inferior a um determinado limiar. Como resultado, o primeiro índice do vetor corresponde à classificação com maior probabilidade e os restantes índices correspondem às restantes possíveis classificações. A primeira classificação é, portanto, a intenção retirada da frase.

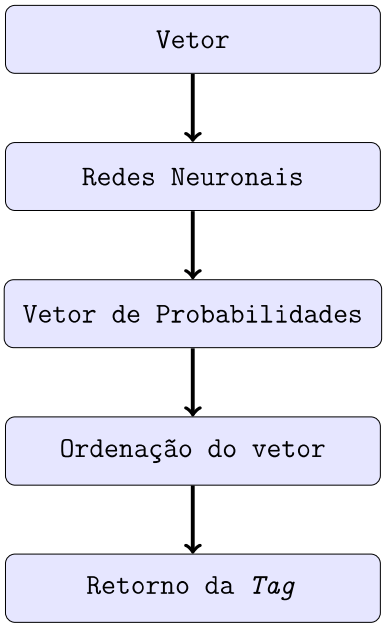

Figura 4.3 – Diagrama da classificação da frase.

4.1.3 Base de Dados

A base de dados tem como função armazenar o PSID do utilizador e o seu idioma. Desta forma, a cada mensagem recebida pelo *chatbot* é verificado se o utilizador existe na base de dados e qual o seu idioma, Figura 4.4.

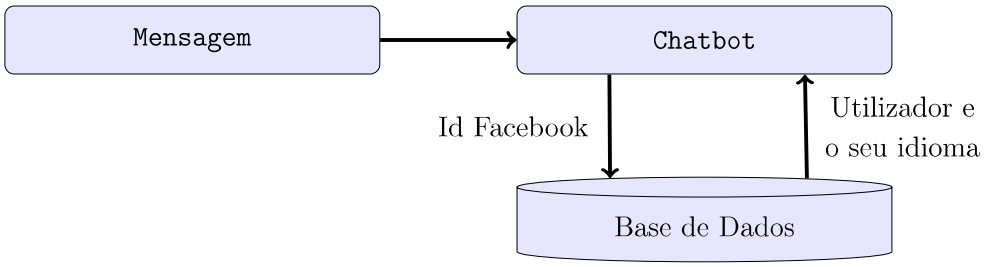

Figura 4.4 – Diagrama com os dados recebidos e enviados pela base de dados

4.1.4 Agente

O agente atua como componente central do *Chatbot*, pois coordena todas as suas funcionalidades. A cada mensagem recebida, consulta a base de dados para verificar se o utilizador está inserido nesta e para confirmar o seu idioma. O agente averigua que tipo de mensagem se trata e, consequentemente, executa uma série de ações diferentes. Tratando-se de uma mensagem do tipo *Message* e subtipo *Text*, como explicado em 4.1.1, o Agente acede a todos os dados necessários para fazer a primeira classificação. Depois de obter uma resposta do *Backend*, dependendo da *tag* resultante da primeira classificação, o Agente consulta o conjunto de respostas ou solicita uma segunda classificação. O processo é apresentado na Figura 4.5. A Figura 4.6 apresenta as *tags* resultantes da primeira e da segunda classificações.

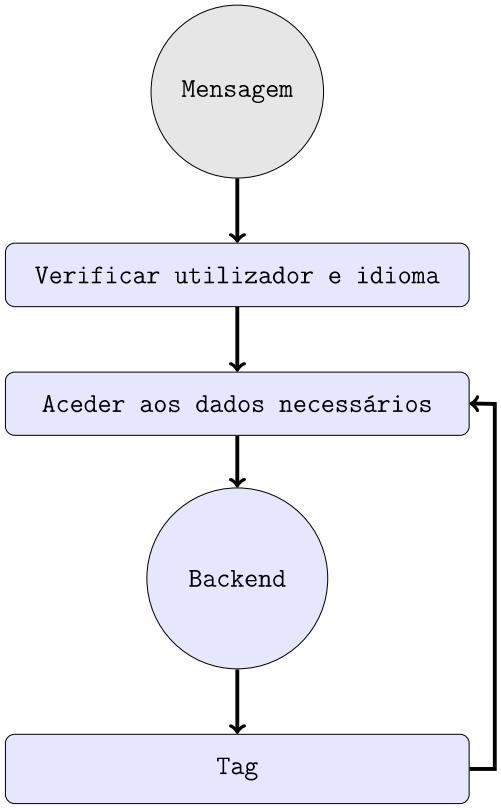

Figura 4.5 – Diagrama dos processos de verificação e carregamento de dados que ocorrem no Agente.

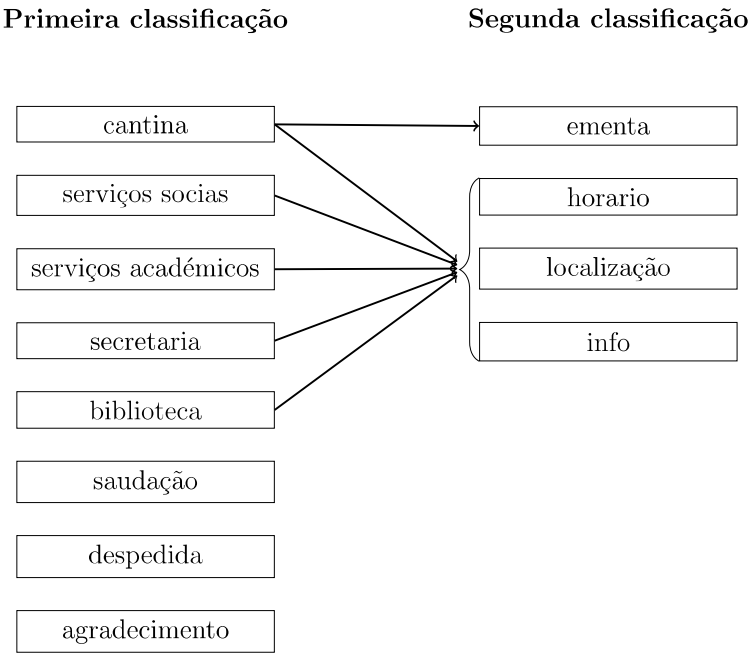

Figura 4.6 – *Tags* resultantes da primeira e segunda classificações

Se a *tag* resultante da primeira classificação corresponder a "saudação", "despedida"ou "agradecimento", o Agente envia imediatamente a resposta ao utilizador, ou seja, não é feita uma segunda classificação.

Se a *tag* resultante da primeira classificação corresponder a "cantina", o resultado da segunda classificação só pode ser "ementa", "horário"ou "localização".

Se a *tag* resultante da primeira classificação corresponder a "serviços sociais", "serviços académicos", "secretaria"ou "biblioteca", a segunda classificação só pode ser "info", "horário"ou "localização". Finalmente, após realizada a segunda classificação, o Agente recorre ao conjunto de respostas e seleciona a resposta mais adequada.

4.2 Implementação

Nesta secção, são abordadas as tecnologias utilizadas e os detalhes da implementação do *chatbot*, os quais são divididos em quatro etapas:

- É explicada a abordagem tomada para a extração de dados usados no treino das redes neuronais e o treino propriamente dito para gerar os modelos.
- São apresentados os procedimentos para configurar o *chatbot* de forma a comunicar com a plataforma Facebook Messenger.
- Descreve-se a estrutura da base de dados.
- Expõe-se detalhadamente do funcionamento da aplicação.

4.2.1 Tecnologias Utilizadas

Para o desenvolvimento do *Chatbot*, foi utilizado *Python*, devido ao seu extenso número de bibliotecas que auxiliam o desenvolvimento de algoritmos e pela sua simplicidade de sintaxe em relação a outras linguagens.

4.2.2 Tensorflow

Tensorflow é uma biblioteca para o cálculo numérico que permite criar algoritmos *Machine learning* (Abadi et al., 2015; Dominic, 2019). Foi desenvolvido por investigadores e engenheiros do grupo Google Brain para uso interno da empresa, porém, mais tarde foi disponibilizado para uso público (Pang, 2018). Oferece flexibilidade porque permite expressar uma variedade de algoritmos, em múltiplas áreas, nomeadamente: reconhecimento de voz, visão computacional, robótica, retorno de informação e NLP.

É uma ferramenta para definir grafos computacionais e um ambiente de execução dos grafos em múltiplos tipos de *hardware*. O Tensorflow utiliza grafos direcionados para representar operações e são denominados *Data Flow Graphs*. Os nós num *Data Flow Graph* representam operações e variáveis. As operações servem para criar e manipular regras específicas e as variáveis para representar os estados partilhados ou persistentes que podem ser manipulados através de operações. As arestas transportam informação como matrizes multi-dimensionais, ou tensores, de um nó para

outro. De forma a entender os conceitos descritos, considere-se o exemplo seguinte:

```
a = 15
b = 5
prod = a * b
sum = a + b
resultado =  prod/sum
```

Representando estas operações através de *Data Flow Graphs*, o resultado é observado na Figura 4.7.

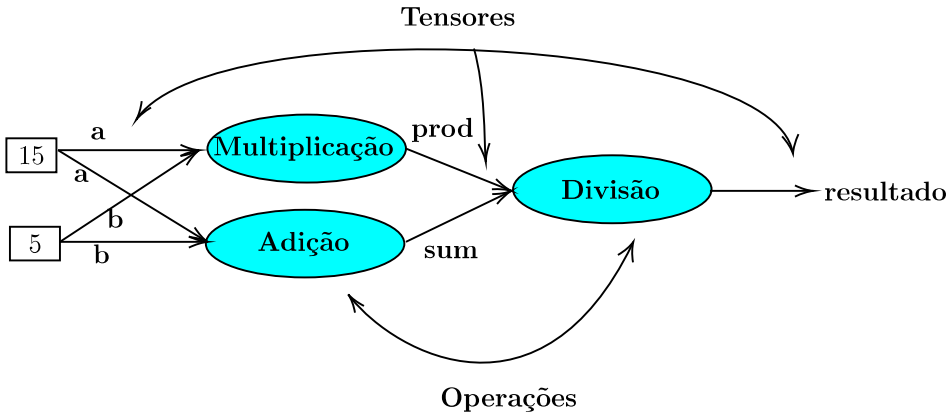

Figura 4.7 – Funcionamento Tensorflow. Adaptado de (Dominic, 2019).

É possível executar o Tensorflow em: *Graphic Processor Units*, ou GPUs, em *Tensor Processing Units*, ou TPUs e em *Computer Processing Units*, ou CPUs, incluindo também as plataformas embebidas e móveis (Dominic, 2019).

4.2.3 Keras

Keras é uma API de alto nível para a implementação de redes neuronais, capaz de executar com *backend* Tensorflow, ou The Microsoft Cognitive Toolkit (CNTK), ou Theano (Chollet et al., 2019). O Keras promove um desenvolvimento rápido, é extremamente modular, ou seja, permite que o processo de teste e de afinação seja feito de forma muito mais rápida e eficaz.

Utilizando Keras, o modelo é entendido como uma sequência de camadas. Os módulos podem ser combinados livremente. O número de camadas da rede neuronal, o tipo de camadas, as funções de erro, as funções de otimização, as estratégias de inicialização, as funções de ativação e os métodos de regularização são todos parâmetros que é possível controlar, editar e usar para gerar os modelos. O *Keras* foi utilizado com *backend* em *Tensorflow* para desenvolver o classificador baseado em redes neuronais (Pang, 2018).

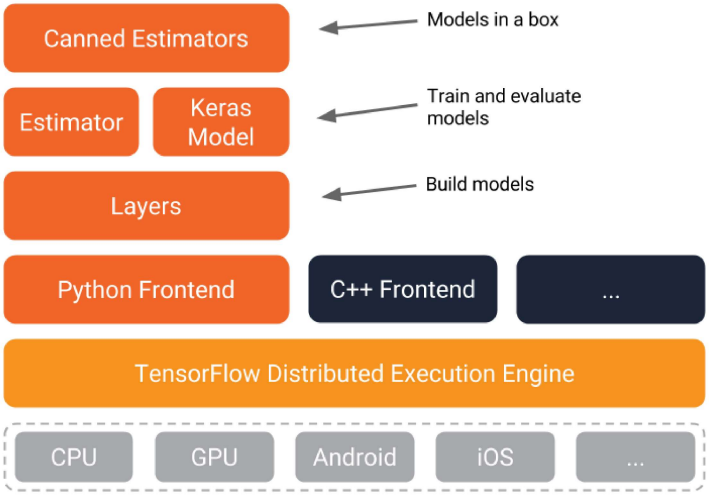

Figura 4.8 – Arquitetura Keras. Retirado de (Unruh, 2019)

4.2.4 Facebook Messenger

A plataforma escolhida para a interação com o *chatbot* foi o Facebook Messenger. A integração do *chatbot* é feita através de um *webhook*, que é um *endpoit* Hyper Text Transfer Protocol Secure, ou HTTPS, preparado para receber pedidos, sendo estes mensagens enviadas ao *chatbot* (Bergholm, 2018).

A plataforma fornece um conjunto de propriedades que permitem criar experiências interativas no *Messenger*, incluindo *APIs* e *plug-ins*. Assim, uma conversa no *Messenger* pode conter muito mais que apenas texto, pois a plataforma permite enviar áudio, vídeo, imagens, e fornece várias opções de estruturação de uma mensagem, onde existem modelos, *quick replies* e botões. Os componentes de conversação são

mensagens de texto, *Attachments*, *Message Templates* e *Quick Replies*. As mensagens de texto são o âmago de qualquer experiência no *Messenger*. Os *Attachments* são elementos como vídeo, áudio, ficheiros ou uma localização. *Message Templates* são mensagens estruturadas, com o propósito de apresentar informação em conversa, que, de outra forma, seria difícil apresentar (Facebook, 2019b). A Figura 4.9 exemplifica uma *Message template*.

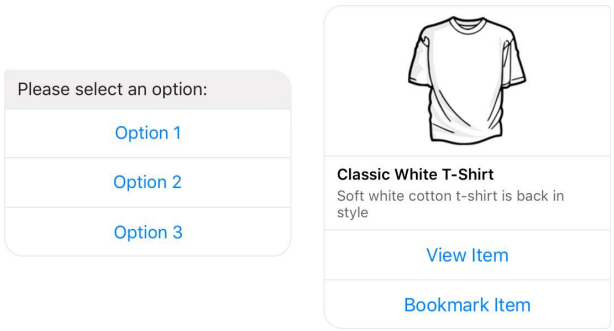

Figura 4.9 – Exemplo de uma *Message Template*. Retirado Facebook (2019b).

- ***Generic Template***: é uma mensagem estruturada que inclui um título, subtítulo, imagem e até três botões. Também suporta a especificação de um objeto que redireciona para um URL, que é acedido diretamente do *Messenger*, Figura 4.10 (Facebook, 2019c).

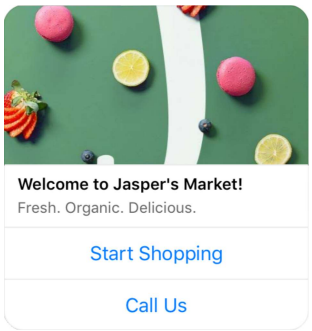

Figura 4.10 – Exemplo de um *Generic Template*. Retirado de Facebook (2019c).

- ***Button Template***: é enviada uma mensagem com limite de três botões. É útil quando se pretende fornecer ao utilizador opções pré-determinadas, respostas ou ações a serem selecionadas, Figura 4.11 (Facebook, 2019a).

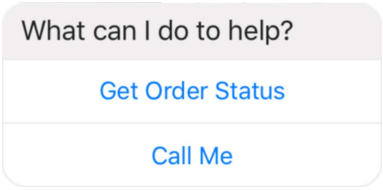

Figura 4.11 – Exemplo de uma *Button Template*. Retirado de Facebook (2019a).

Por fim, as *Quick Replies* permitem apresentar uma série de opções ao utilizador. Quando uma *Quick Reply* é escolhida, uma mensagem de texto é enviada ao *webhook*, Figura 4.12.

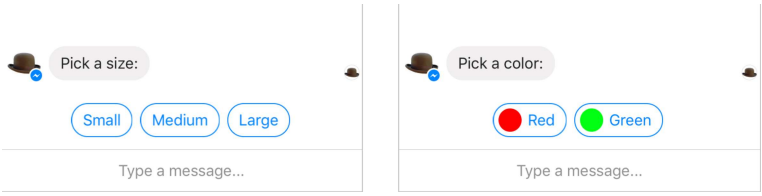

Figura 4.12 – Exemplo de uma *Quick Reply*. Retirado de Facebook (2019b).

Webhook

O *webhook* é um *endpoint* HTTPS para o qual o Facebook envia eventos, é único e aceita pedidos POST. Existe um conjunto de eventos que o *webhook* pode receber. De forma a implementar uma operação básica, existem dois eventos obrigatórios, *messages* e *messaging postbacks* (Facebook, 2019i).

- ***Messages***: este evento ocorre quando uma mensagem é enviada para a página. Pode-se receber mensagens de texto, *templates* ou anexos, como imagens, áudio, ficheiros ou localizações (Facebook, 2019e).

- ***Messaging Postbacks***: este evento ocorre quando um botão é pressionado, ou um item no menu (Facebook, 2019f).

4.2.5 Criação dos Modelos e Treino

O objetivo do *chatbot* é fornecer informações ao utilizador, por esse motivo, é necessário extrair a intenção que o utilizador transmite com a frase de forma a responder corretamente a este. Para ser possível classificar a frase numa intenção optou-se por uma abordagem supervisionada, ou seja, usou-se exemplos de frases para treinar as redes neuronais.

Todos os dados de treino são relativos à UTAD e foram introduzidos manualmente. É também importante referir que o *chatbot* suporta 2 idiomas, português e inglês, sendo que as frases encontradas no conjunto inglês são apenas a tradução das mesmas encontradas no conjunto português.

Os dados foram inseridos num ficheiro numa folha de cálculo. Escolheu-se este formato com o fim de, futuramente, facilitar a um funcionário da universidade a inserção de mais dados para continuar a treinar o *chatbot*, porque é habitual os funcionários administrativos terem conhecimentos neste tipo de ferramentas.

Dados utilizados

Os dados em português e inglês são exatamente iguais em estrutura e significado das frases, por isso, esclarece-se que a abordagem utilizada nos dados em português é igual à utilizada nos dados em inglês. A extração dos dados e a sua manipulação também foram automatizados, de forma que, se forem adicionados mais dados não seja necessário realizar alterações na implementação, o que torna este processo mais rápido.

Os ficheiros português e inglês estão divididos em seis páginas com o nome de:

Principal, *Cantina*, *Serviços Académicos*, *Serviços Sociais*, *Secretaria* e *Biblioteca*.

Cada página contém as frases que serão utilizadas para treinar uma rede neuronal. Em cada página os dados estão organizados por colunas: *tag*, *patterns* e *responses*.

- ***tag***: corresponde ao rótulo da frase que pode também ser referida como a classe que a frase pertence.
- ***patterns***: corresponde à frase propriamente dita.
- ***responses***: resposta para a frase enviada ao utilizador.

A coluna *tag* contém as classes ou intenções a que as frases da coluna *patterns* pertencem. Por exemplo, a frase "sabes informações úteis dos serviços sociais?"pertence à *tag* "serviços sociais".

Na Tabela 4.1 estão representadas as *tags* presentes nas páginas do ficheiro português e inglês.

Página	**Tags das páginas do ficheiro português**	**Tags das páginas do ficheiro inglês**
Principal	serviços sociais serviços académicos cantina secretaria biblioteca agradecimento despedida saudação	social services academic services canteen secretary library thanks farewell salutation
Cantina	localização horário ementa	location schedule menu
Serviços Académicos ***Serviços Sociais*** ***Secretaria*** ***Biblioteca***	localização horário info	location schedule info

Tabela 4.1 – Tabela ilustrativa das tags presentes nas diferentes páginas

Um exemplo dos dados em português está representado na Figura 4.13 e dos dados em inglês na Figura 4.14.

tag	patterns	responses
localizaçao	trajeto até a cantina	Vou-te mandar a localização das cantinas
localizaçao	Em que lugar se localiza a cantina?	Vou-te mandar a localização das cantinas
ementa	diz-me a ementa da cantina	http://www.sas.utad.pt/Lists/Ementas/AllItems.aspx
localizaçao	direções para a cantina	Vou-te mandar a localização das cantinas
ementa	almoço na cantina de além rio	http://www.sas.utad.pt/Lists/Ementas/AllItems.aspx

Figura 4.13 – Exemplo dos dados em Português, da página *Cantina*.

tag	patterns	responses
location	way to the canteen	I'm going to send you the location of the canteen
location	Where is the canteen located?	I'm going to send you the location of the canteen
menu	tell me the menu of the canteen	http://www.sas.utad.pt/Lists/menus/AllItems.aspx
location	directions to the canteen	I'm going to send you the location of the canteen
menu	lunch in the canteen of além rio	http://www.sas.utad.pt/Lists/menus/AllItems.aspx

Figura 4.14 – Exemplo dos dados em Inglês, da página *Cantina*.

Cada página contém dois conjuntos de dados, um destinado ao treino e outro a validação. As frases de validação são utilizadas para validar o desempenho da rede neuronal durante o treino e não são utilizados para treinar os modelos. Optou-se por utilizar aproximadamente 80% das frases para treino e 20% das frases para validação.

A tabela 4.2 apresenta o número de exemplos para treino e validação de cada página.

	Principal	***Cantina***	***Serviços Sociais***	***Serviços Académicos***	***Secretaria***	***Biblioteca***
Nº de exemplos de treino	193	74	70	80	70	70
Nº de exemplos de validação	64	25	23	26	23	23

Tabela 4.2 – Número de exemplos de treino e validação por página.

Pode-se observar que a página que contém mais exemplos de treino e validação é a página *Principal*, isto porque contém frases que estão presentes nas outras páginas e apresenta mais *tags*.

A tabela 4.2 indica o número de frases presente em cada página, porém, não indica qual a distribuição de *tags* em cada página.

Um aspeto importante quando se aborda a classificação multi-classe é a distribuição das diferentes *tags* no conjunto de dados. Porque um conjunto de dados é altamente desequilibrado quando o número de exemplos de uma classe é muito superior ao número de exemplos de outra classe, o que implica que a classificação será tendenciosa (Longadge and Dongre, 2013). Nestas situações, um classificador não é imparcial, favorecendo a classe mais numerosa e ignorando a classe menos numerosa, o que não é satisfatório. Este problema é denominado de "Desequilíbrio de classes" (Chawla et al., 2004) e, normalmente, é observado em conjuntos que apresentam uma diferença entre classes de 100 exemplos para cada 1 exemplo, de 1000 para 1, de 10000 para 1 e assim sucessivamente. Por esta razão, analisou-se os conjuntos de dados, para averiguar se existia desequilibro entre as classes de dados. Assim procedeu-se à análise da distribuição das *tags* em cada página.

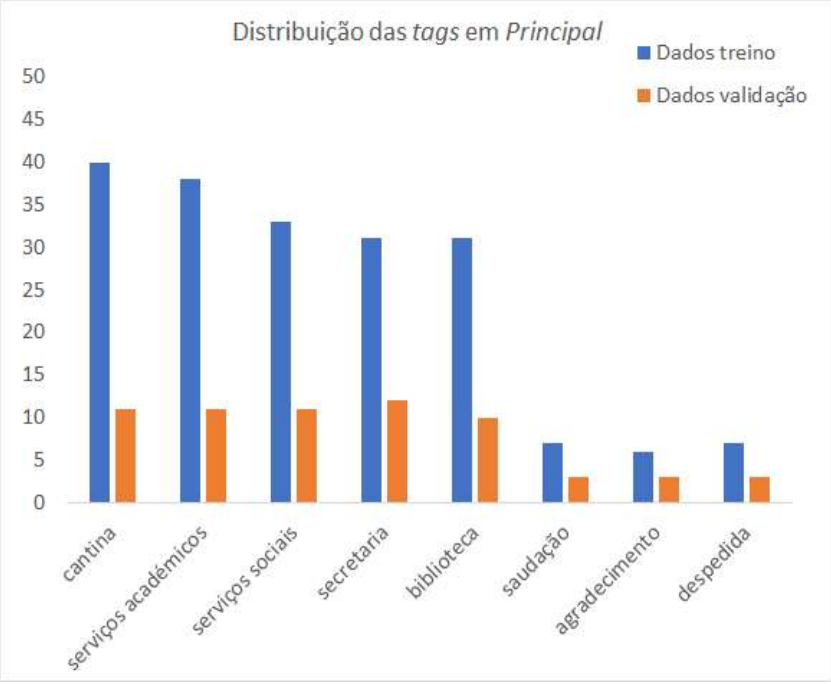

Figura 4.15 – Distribuição das classes (*tags*) na página *Principal*

Observou-se que no gráfico da Figura 4.15 as *tags* "cantina", "serviços sociais", "serviços académicos", "secretaria", "biblioteca"contêm entre 30 e 40 frases de

treino. As *tags* de "saudação", "agradecimento"e "despedida"apresentam muito menor número de frases em relação às outras *tags*. Avaliando a distribuição da 4.15, conclui-se que não existe o problema do "Desequilíbrio de classes". No entanto, se ocorrer uma tendência na classificação, as *tags* prejudicadas serão a *tag* "saudação", "agradecimento"e "'despedida"porque são as que apresentam menor número de frases.

De forma análoga, observou-se a distribuição das *tags* na página *Cantina* (Figura 4.16), da página *Serviços Sociais* (Figura 4.17), da página *Serviços Académicos* (Figura 4.18), da página *Secretaria* (Figura 4.19) e da página *Biblioteca* (Figura 4.20).

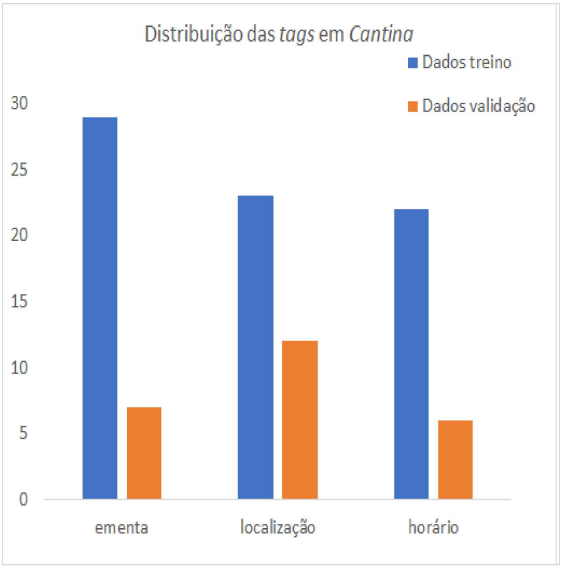

Figura 4.16 – Distribuição das classes (*tags*) na página *Cantina*

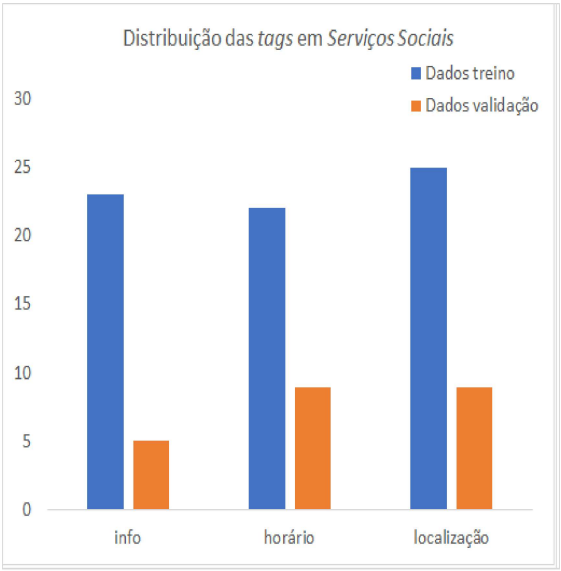

Figura 4.17 – Distribuição das classes (*tags*) na página *Serviços Sociais*

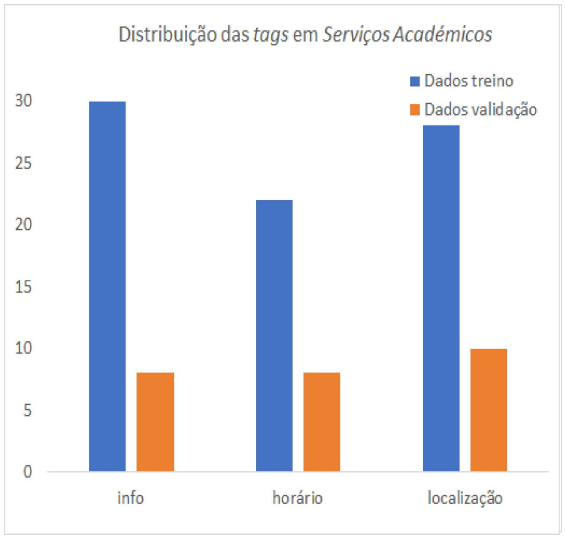

Figura 4.18 – Distribuição das classes (*tags*) na página *Serviços Académicos*

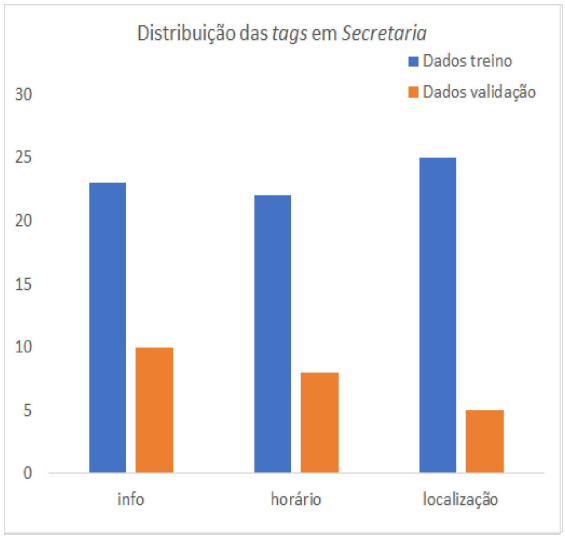

Figura 4.19 – Distribuição das classes (*tags*) na página *Secretaria*

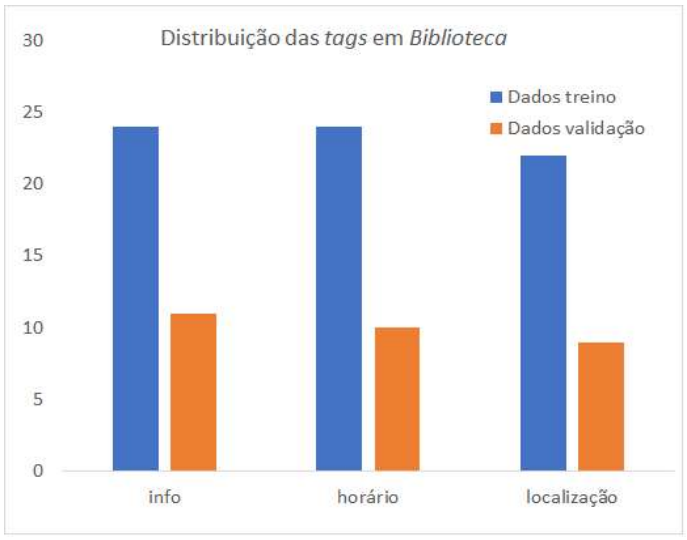

Figura 4.20 – Distribuição das classes (*tags*) na página *Biblioteca*

Dos gráficos anteriores, é possível observar que não existe o problema do "Desequilibro de classes", ou *tags*, porque estas contêm todas entre 20 a 30 exemplos de treino e 5 a 10 exemplos de validação. Desta forma, pode-se concluir que não deve existir uma tendência nos modelos para escolher alguma das classes. No entanto, se existir

tal tendência, esta não é provocada pelo desequilibro entre as *tags*.

Extração e pré-processamento dos dados

Depois de se obter um conjunto de dados relevantes para treinar as redes neuronais efetuou-se o seu processamento. As frases não podem ser introduzidas diretamente numa rede neuronal, porque não é possível exprimir as palavras imutáveis computacionalmente, assim, os passos seguintes são importantes para se obter uma estrutura que permite as redes neuronais aprender a informação presente nos dados.

Como exemplo, é considerada a página *Principal*, mas o processo descrito é idêntico para as restantes páginas.

Em primeiro lugar, é criada uma lista das *tags* presentes na página e uma lista de todas as palavras únicas retiradas da mesma página. Estas listas desempenham um papel crucial pois são usadas para criar os *Bag-of-words*. De seguida, é criada uma lista de documentos, em que cada documento contém uma frase e a sua respetiva *tag*, tal como é apresentado no exemplo seguinte:

(“horario da biblioteca”, “biblioteca”)

Neste documento, a frase é submetida ao pré-processamento explicado anteriormente, na sub-subsecção 4.1.2, ou seja, retira-se todos os acentos, pontuação e substitui-se as letras maiúsculas por minúsculas. Para efetuar o pré-processamento da frase foi utilizada a biblioteca Natural Language Toolkit, ou NLTK. Esta fornece recursos de classificação de texto, de *tokenization*, de *stemming*, de *parsing*, entre outros (Loper and Bird, 2019). Depois, para efetuar a *tokenization* utilizou-se a função *tokenize* (NLTK, 2019b). Removeu-se as *stop-words* e reduziu-se as palavras à sua raiz usando o algoritmo *Snowball Stemmer* (NLTK, 2019a). Adotando o documento acima apresentado, a palavra “horario”é reduzida para “hora”e a palavra “biblioteca”para “biblio”.

([“hora”,“biblio”], “biblioteca”)

No final, o *Bag-of-words* é criado e o resultado é o seguinte:

([0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0....], [0 1 0 0 0 0 0 0])

Neste exemplo em particular, depois dos processos explicados, pode-se observar que a frase "horário da biblioteca"originou um vetor com o tamanho correspondente ao número total de palavras únicas na página *Principal*, nos quais, os índices 8 e 25 têm o valor de 1, que corresponde às raízes "hora"e "biblio". O segundo vetor, correspondente à *tag*, apresenta valor de 1 no índice 2, porque este corresponde à *tag* "biblioteca". Para finalizar, todos os *Bag-of-words* são colocados numa lista.

Criação dos modelos de redes neuronais

O próximo passo consiste em criar os modelos de redes neuronais, usando o Keras. A lista resultante do processo anterior é baralhada e transformada numa matriz Numpy (Oliphant, 2006). A matriz Numpy é dividida, em 2 listas, lista X que contém a frase e a lista Y que contém a *tag*. Assim, retomando o exemplo anterior, a frase está representada na Figura 4.21 e a *tag* na Figura 4.22.

Lista X

↓

[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 .....]

Figura 4.21 – Lista X - Frase

Lista Y

↓

[0 1 0 0 0 0 0 0]

Figura 4.22 – Lista Y - *tag*

As redes neuronais criadas apresentam uma camada de entrada, uma de saída e uma interior, como observado na Figura 4.23. Durante o treino, a entrada da rede

neuronal corresponde à lista X e a saída à lista Y. O número de neurónios da camada de entrada e de saída é igual ao tamanho de cada uma das listas, sendo que a camada de entrada tem sempre o tamanho da lista X, que depende do número de palavras únicas presentes em cada página e a camada de saída tem sempre o tamanho da lista Y, que depende do número de *tags* da folha.

Por exemplo, na página *Principal*, existem oito *tags* diferentes, logo, o tamanho da lista Y é de oito, porém, nas outras páginas, existem apenas três *tags* diferentes, nestas páginas o tamanho da lista Y é de apenas três. As funções de ativação da camada interior podem variar, nesta dissertação implementou-se a função tangente hiperbólica e a função ReLU. Para a camada de saída optou-se por utilizar a função softmax, pois permite ter uma distribuição probabilística das classes.

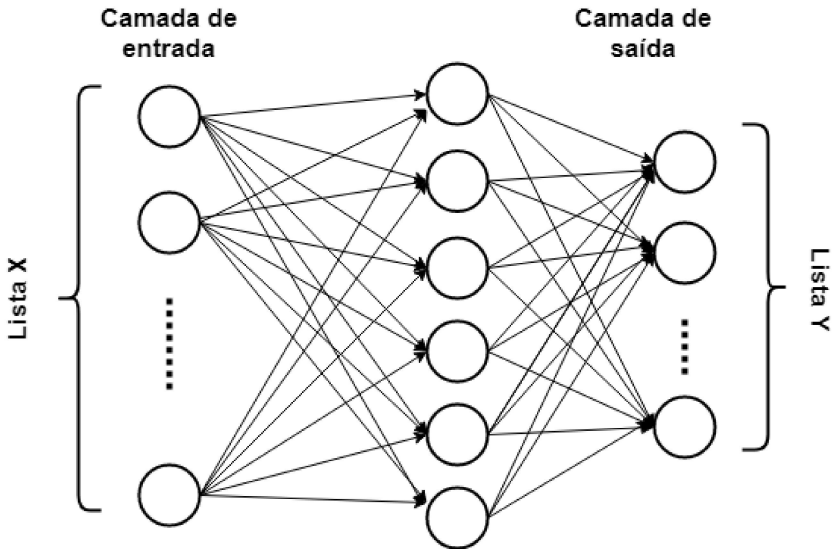

Figura 4.23 – A arquitetura das redes neuronais usadas

O resultado do treino foram doze modelos de redes neuronais, seis modelos neuronais para o idioma português e seis para o idioma inglês.

Criaram-se também dois ficheiros de configuração, um para o idioma português e outro para o idioma inglês.

Os ficheiros de configuração são utilizados pelo *chatbot*, para a saber a que rede neuronal recorrer e que vocabulário usar para criar o *bag-of-words* quando conversa

com o utilizador. A tabela 4.3 exemplifica o tipo de dados armazenados num ficheiro de configuração.

Ficheiro de configuração	dados.json
chave	cantina
modelo	DATA_model_cantina.h5
palavras	lista das palavras únicas retiradas da página *Cantina*
documentos	lista de documentos retirados da página *Cantina*
classes	lista de tags retiradas da página *Cantina*
respostas	lista de respostas retiradas da página *Cantina*

Tabela 4.3 – Tabela exemplificativa dos dados presentes num documento de configuração.

4.2.6 Conexão com a plataforma Facebook Messenger

Quando uma pessoa envia uma mensagem no Facebook Messenger, a sua mensagem é recebida pelo servidor do Facebook. De seguida, este envia eventos para o servidor onde aplicação está instalada, denominado *Business Server*. Utilizando o Send API, a aplicação consegue responder à pessoa que enviou a mensagem no Facebook Messenger (Facebook, 2019d).

Um exemplo do funcionamento da plataforma está representado na Figura 4.24.

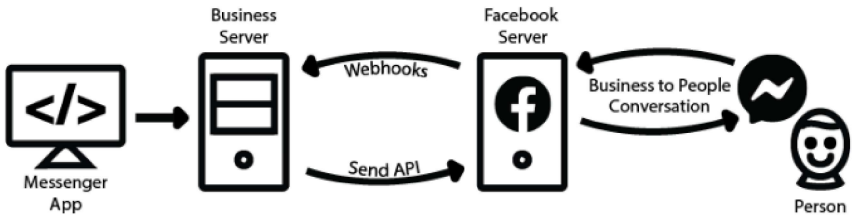

Figura 4.24 – Funcionamento da plataforma *Messenger*. Retirado de Facebook (2019d).

Para criar o *Chatbot* é necessário (Facebook, 2019d):

- Criar uma página de Facebook

- Criar uma conta de *developer* do Facebook
- Criar e configurar um *webhook*
- Criar uma aplicação Facebook

A página de Facebook identifica o *chatbot*. Nesta dissertação, a página tem o nome de Flask_chatbot.

Configuração do *Webhook*

O *webhook* é o URL público onde o código está hospedado, onde são recebidas, envidas as mensagens e processada a informação. Todas as instruções sobre a configuração de um *webhook* podem ser encontradas em (Facebook, 2019h), porém uma breve explicação do processo será dada, assim, será descrito como o *webhook* foi configurado para passar a verificação da plataforma Facebook Messenger e aceitar eventos HTTP.

O servidor que contém o *webhook* tem que ter os seguintes requisitos:

- Suporte HTTPS, qualquer certificado auto-assinado não é suportado
- Um certificado SSL válido
- Uma porta aberta, capaz de receber pedidos GET e POST

O serviço utilizado para hospedar a aplicação foi o Heroku, que é uma plataforma baseada na *cloud* que permite a construção, distribuição, monitorização e o desenvolvimento de aplicações (Heroku, 2019).

Para criar o servidor HTTP, utilizou-se a biblioteca Flask (Ronacher, 2019). É criado um *endpoint* capaz de receber pedidos GET e POST. Cada vez que um evento é recebido no servidor é necessário responder com 200"OK", que indica à plataforma Facebook Messenger que este evento foi recebido.

De seguida foi necessário adicionar a verificação do *webhook* para garantir que o *webhook* é autêntico e funcional. O processo de verificação tem os seguintes passos (Facebook, 2019h):

1. Criação de um *token* de verificação. No nosso caso, este token não foi inserido manualmente no código, mas sim, inserido no servidor Heroku como uma variável de configuração como é recomendado pelo Heroku (Heroku, 2020). Esta medida está em conformidade com a metodologia desenvolvimento de aplicações *Software-as-a-Service* (SaaS), denominada de Twelve-Factor App. Segundo esta metodologia uma variável de configuração pode ser uma credencial para um serviço externo, como o Facebook. Para não violar a metodologia não se pode armazenar as variáveis de configuração no código fonte da aplicação garantindo, desta forma, que não seja necessário alterar o código fonte da aplicação cada vez que seja necessário alterar a variável de configuração (Wiggins, 2017).

2. Fornecimento do *token* de verificação à plataforma Facebook Messenger, quando se subscreve o *webhook* para receber eventos de uma aplicação Facebook.

3. A plataforma Facebook Messenger envia um pedido GET ao *webhook* com o *token* de verificação inserido na plataforma.

4. Verificação se o *token* recebido corresponde ao *token* criado.

5. Se o token for confirmado, a plataforma subscreve ao *webhook*

Configuração da aplicação Facebook

Depois de se configurar o *webhook*, criar uma página Facebook e conta de *developer* é necessário criar a aplicação Facebook e configurar esta de forma que todos os componentes fiquem conectados, pois uma aplicação Facebook é a ponte de ligação entre o *webhook* e a página criada (Facebook, 2019g).

Para conectar o *webhook* à aplicação:

1. Adiciona-se à aplicação o URL do *webhook* criado

2. Insere-se o *token* de verificação do *webhook* à aplicação

3. Subscreve-se aos eventos *messages* e *messages postbacks*

De seguida, para se subscrever a aplicação à página de Facebook criada foi necessário:

1. Selecionar a página que se pretende subscrever a aplicação, no nosso caso, a página Flask chatbot

2. Copiar o *token* de acesso da página

3. Subscrever o *webhook* também à página Flask chatbot

Com os passos explicados a plataforma Facebook Messenger foi capaz de enviar eventos para o qual se subscreveu, (*messages* e *messages postbacks*) para o *webhook* (Facebook, 2019g).

4.2.7 Base de dados MongoDB

Uma das características desejadas no *chatbot* é a capacidade de falar com todos os novos alunos, de várias origens. Por essa razão, é importante o *chatbot* suportar inglês. Optou-se por criar uma base de dados em MongoDB (MongoDB, 2006), para armazenar o idioma de cada utilizador. Cada utilizador tem um ID único associado à página Flask_chatbot denominado PSID, deste modo, pode-se associar um idioma a cada utilizador. Na base de dados é guardado o PSID do utilizador e o seu idioma de preferência. Por defeito, a primeira vez que um novo utilizador contacta o *chatbot*, é inserido o idioma inglês, porém o utilizador pode alterá-lo a qualquer momento, escrevendo no *chatbot* "linguagem"ou "*language*".

4.2.8 Funcionamento da aplicação

Nas subsecções anteriores, explicou-se o processo de preparação dos dados, bem como a abordagem por detrás da construção das redes neuronais. Foi explicado o processo de criação de uma aplicação no Facebook Messenger, como configurar todos os seus componentes e, por fim, a função da base de dados criada. Nesta subsecção é explicado o funcionamento da aplicação desenvolvida.

A primeira vez que o utilizador interage com o *chatbot* é pedida a permissão para armazenar o seu PSID na base de dados. A partir deste momento, a cada mensagem recebida é feita uma *query* à base de dados, a fim de consultar o idioma de preferência do utilizador. Este passo é importante para o processamento e classificação da frase, pois o *chatbot* necessita de saber que conjunto de *stop-words* é necessário utilizar e aceder ao ficheiro de configuração correto. De igual forma, como se processou as frases utilizadas no treino usando as etapas expostas na subsecção 4.1.2, também todas as frases recebidas são pré-processadas para executar primeira classificação.

O primeiro modelo de redes neuronal a ser utilizado para classificar a frase é o modelo *Principal*, ou seja, DATA_model_principal. O modelo tem a função de fornecer a primeira intenção da frase, que pode ser qualquer das *tags* presentes na página *Principal*, apresentadas na Tabela 4.1.

No caso da primeira *tag* corresponder a "agradecimento", "despedida"ou "saudação", uma resposta é retirada do conjunto de respostas e, por isso, já não é feita uma segunda classificação.

No entanto, se o resultado da primeira classificação for qualquer uma das outras *tags*, presentes na Tabela 4.1, é realizada uma segunda classificação obtendo-se uma *tag*. O segundo modelo a ser utilizado depende da primeira classificação, por exemplo, quando a primeira classificação corresponde a "serviços sociais"o segundo modelo de redes neuronais utilizado é DATA_model_serviços_sociais. Neste caso, da segunda classificação apenas resulta uma das *tags* presentes na página *Serviços Sociais*.

Se o resultado da segunda classificação é a *tag* "localização", é enviado ao utilizador

um pedido para este partilhar a sua localização. Em resposta, o *chatbot* envia um *template* para o utilizador, que o permite redirecionar para o *site* do Google Maps. O trajeto disponibilizado é o trajeto desde a localização partilhada pelo utilizador até ao edifício que resultou da classificação.

5 Testes e Resultados

Neste capítulo são efetuados os testes e analisado o desempenho do *chatbot*. Em primeiro lugar, foram observados os valores de precisão e erro obtidos pelos modelos de redes neuronais durante o treino. Em segundo lugar, foram apresentadas as matrizes de confusão, ou matrizes de erro, que permitem calcular a precisão dos modelos de redes neuronais depois do treino. Por fim, foi testado o comportamento do *chatbot* na plataforma Facebook Messenger, apresentando imagens de conversas com o *chatbot*.

5.1 Testes das Redes Neuronais

O objetivo de qualquer modelo de *Machine Learning* é de apresentar algum grau de generalização. Esta generalização, em *Supervised Learning* é conseguida através do treino dos modelos, para isso, é necessário evitar que ocorra *underfitting* ou *overfitting* do modelo aos dados utilizados no treino. Quando o modelo não consegue ajustar-se aos dados sofre *underfitting*. O *underfitting* pode ser observado quando o valor das funções de erro, nos dados de treino e validação, são altos e não se aproximam de 0. Por outro lado, quando o modelo aprende demasiada informação dos

dados de treino, sofre *overfitting*, que pode ser observado quando existe um aumento do valor da função de erro nos dados de validação e simultaneamente uma diminuição do valor da função de erro nos dados de treino (Jordan, 2018). Em qualquer destes casos, o modelo não generaliza bem, ou seja, num caso de classificação, não consegue classificar corretamente exemplos que não foram observados durante o treino. Deste modo, é necessário encontrar os parâmetros corretos do modelo, para prevenir *underfitting* e *overfitting*, daí ser importante visualizar as curvas de validação (Islam and Morimoto, 2015). Foram testadas duas abordagens para encontrar uma solução satisfatória, no qual se procurou minimizar o valor da função de erro e maximizar o valor da precisão observados nos dados de treino e validação.

Considerou-se alguns parâmetros das redes neuronais fixos, como o número de neurónios, o número de camadas internas das redes neuronais e também o *Batch size*. Pode-se definir o *Batch size* como o número de exemplos com que um modelo treina, antes de atualizar os seus parâmetros internos, ou seja, os pesos das redes neuronais (Brownlee, 2019a). Assim, utilizou-se um *Batch size* de 5 e uma única camada interna de neurónios. Foi utilizado o Adam como método de otimização (disponibilizado pelo Keras (Chollet et al., 2015a)), que apresenta uma alternativa ao clássico *Gradient Descent* para atualizar os pesos das redes neuronais durante o treino (Brownlee, 2019b). Optou-se pelo Adam porque é eficiente computacionalmente, requerendo pouca memória (Kingma and Ba, 2014). A função de erro utilizada foi *categorical crossentropy loss*, que compara a distribuição das previsões com a distribuição correta. Quanto mais perto estiver a saída do modelo dos valores reais, menor será o valor da função. Utilizou-se esta função de erro porque é normalmente utilizada em problemas multi-classe, no qual, apenas uma classe está correta (Peltarion, 2019). A equação 5.1 traduz a *categorical crossentropy loss*, em que y representa o valor real e $\hat{y}$ o valor previsto.

$$L(y, \hat{y}) = -\sum_{j=0}^{M}\sum_{i=0}^{N}(y_{ij} * log(\hat{y}_{ij})) \tag{5.1}$$

É importante referir que todas as experiências foram realizadas 3 vezes, porque

devido à inicialização aleatória do valor dos pesos dos neurónios, experiências consecutivas podem ter valores de precisão e erro diferentes. Assim, as experiências apresentadas apresentam os melhores resultados obtidos.

Para escolher o número de neurónios presentes na camada interior, considerou-se a junção dos três pontos:

- O número de neurónios na camada interior deve ser entre o tamanho da camada de entrada e o tamanho da camada de saída.
- O número de neurónios na camada interior deve ser 2/3 do tamanho da camada de entrada mais o tamanho da camada de saída.
- O número de neurónios na camada interior deve ser menos de metade do tamanho da camada de entrada.

Experimentou-se como funções de ativação, a tangente hiperbólica e o ReLU (Chollet et al., 2015b) na camada interior. Na camada de saída utilizou-se a função *softmax*, para se obter na saída uma distribuição de probabilidades das classes.

5.1.1 Treino e Testes no *Dataset* em Português

Com os seguintes testes pretendeu-se obter o maior valor de precisão e menor valor da função de erro possível, observados nos dados de treino e validação. Para isso, testou-se duas funções de ativação: tangente hiperbólia e ReLU. A escolha destas duas funções, recaiu no facto de ambas serem comummente utilizadas em redes neuronais. Assim, por estas razões, decidiu-se testar o seu desempenho no conjunto de dados português e averiguar, segundo os nossos objetivos, qual função utilizar.

Teste com a função tangente hiperbólica

Com a utilização da função tangente hiperbólica espera-se um valor de precisão nos dados de validação (Val.ACC) o mais próximo possível de 1, e um valor da função de

erro nos dados de validação (Val.LOSS) o mais próximo possível de 0. O resultado dos testes usando a função de ativação tangente hiperbólica são apresentados na Tabela 5.1.

Modelo	Nº de neurónios	Train ACC	Train LOSS	Val. ACC	Val. LOSS
Principal	74	0.99	0.02	0.94	0.2
Cantina	27	0.99	0.13	0.76	0.7
Serviços académicos	28	0.95	0.05	0.96	0.06
Serviços sociais	33	0.94	0.13	0.92	0.25
Secretaria	27	0.99	0.03	0.96	0.08
Biblioteca	30	0.95	0.12	0.88	0.49

Tabela 5.1 – Tabela dos resultados finais usando a tangente hiperbólica no *dataset* português.

Teste com a função ReLU

De forma semelhante ao teste anterior, ao utilizar a função ReLU, espera-se um valor precisão nos dados de validação (Val.ACC) o mais próximo de 1 possível, e um valor da função de erro (Val.LOSS) o mais próximo de 0 possível. O resultado dos testes usando a função de ativação ReLU é apresentado na Tabela 5.2.

Modelo	Nº de neurónios	Train ACC	Train LOSS	Val. ACC	Val. LOSS
Principal	74	0.99	0.02	0.99	0.02
Cantina	27	0.99	0.03	0.96	0.03
Serviços académicos	28	0.98	0.04	0.96	0.04
Serviços sociais	33	0.95	0.09	0.96	0.16
Secretaria	27	0.99	0.03	0.96	0.08
Biblioteca	30	0.99	0.07	0.99	0.01

Tabela 5.2 – Tabela dos resultados finais usando ReLU no *dataset* português.

Observou-se uma melhoria significativa na escolha de ReLU como função de ativação em vez da tangente hiperbólica. Modelos como a *Cantina* e *Biblioteca* foram os que mais beneficiaram, porque ocorreu uma grande redução do valor da função de erro

e um aumento da precisão nos dados de validação. Desta forma, concluiu-se que a função ReLU ajusta-se melhor aos dados utilizados.

No entanto, analisar só os valores finais obtidos pelos diferentes modelos não é suficiente para saber se os modelos sofreram *underfitting* ou *overfitting* durante o treino. Um modelo sofre *underfitting* quando os valores das suas funções de erro (validação e treino) são elevados e não se aproximam de 0. Está-se perante um caso de *overfitting* se o valor da função de erro nos dados de validação começar a aumentar, divergindo-se do valor da função de erro nos dados de treino que, por sua vez, diminui. Isto acontece porque o modelo está a aprender demasiado sobre os dados de treino e a generalizar pouco, daí o erro nos dados de validação aumentar (Jordan, 2018). Os gráficos seguintes correspondem ao treino e validação de todos os modelos, utilizando a função ReLU.

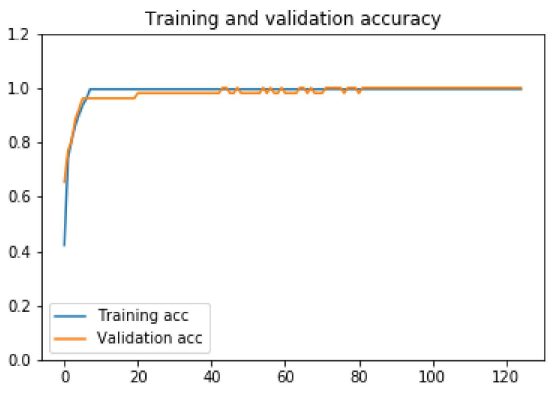

Figura 5.1 – Gráfico das funções de precisão do modelo *Principal* ao longo do treino, no *dataset* em português, usando a função ReLU.

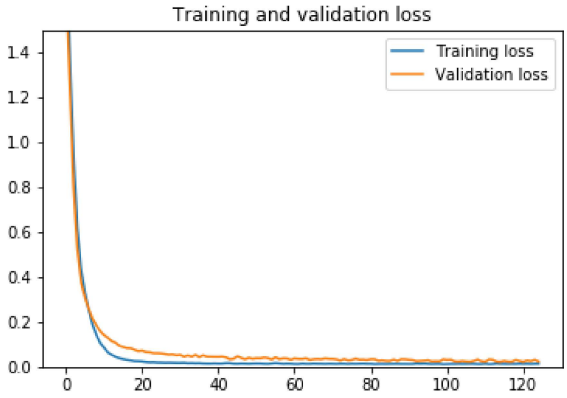

Figura 5.2 – Gráfico das funções de erro do modelo *Principal* ao longo do treino, no *dataset* em português, usando a função ReLU.

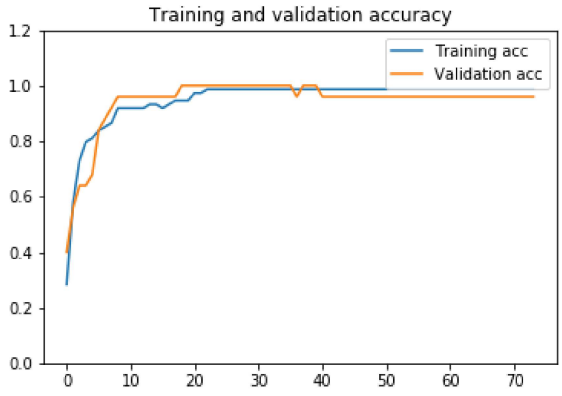

Figura 5.3 – Gráfico das funções de precisão do modelo *Cantina* ao longo do treino, no *dataset* em português, usando a função ReLU.

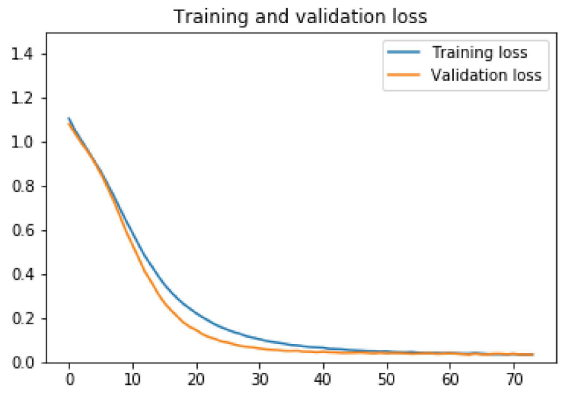

Figura 5.4 – Gráfico das funções de erro do modelo *Cantina* ao longo do treino, no *dataset* em português, usando a função ReLU.

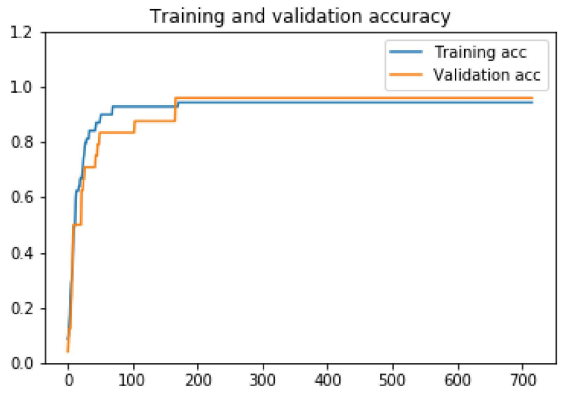

Figura 5.5 – Gráfico das funções de precisão do modelo *Serviços Sociais* ao longo do treino, no *dataset* em português, usando a função ReLU.

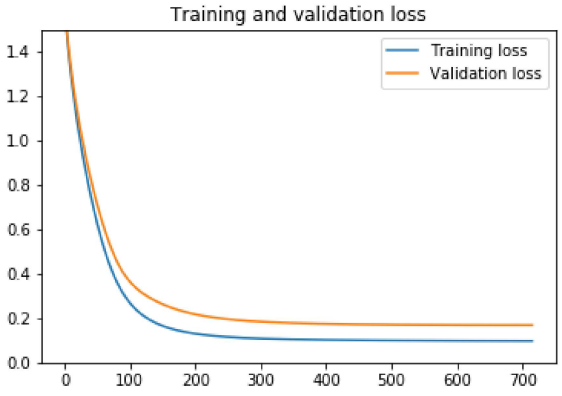

Figura 5.6 – Gráfico das funções de erro do modelo *Serviços Sociais* ao longo do treino, no *dataset* em português, usando a função ReLU.

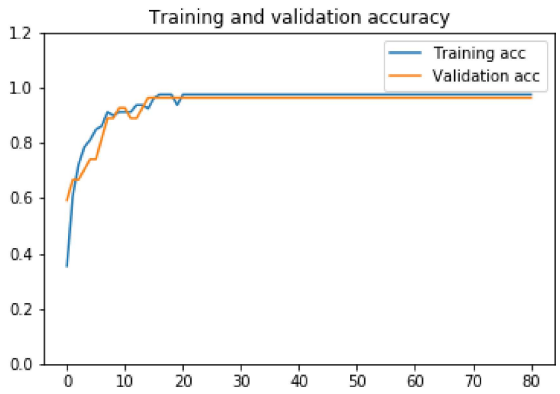

Figura 5.7 – Gráfico das funções precisão do modelo *Serviços Académicos* ao longo do treino, no *dataset* em português, usando a função ReLU.

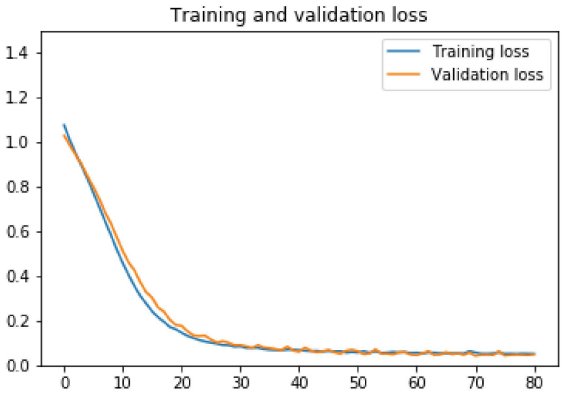

Figura 5.8 – Gráfico das funções de erro do modelo *Serviços Académicos* ao longo do treino, no *dataset* em português, usando a função ReLU.

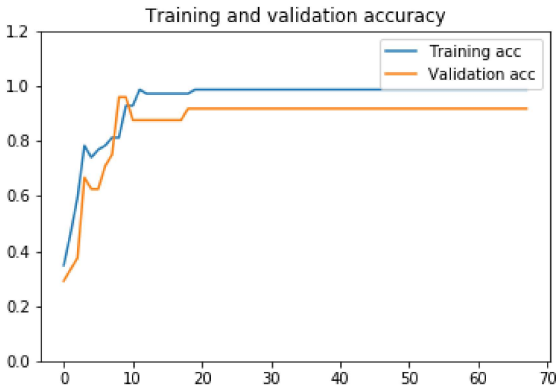

Figura 5.9 – Gráfico das funções de precisão do modelo *Secretaria* ao longo do treino no *dataset* em português, usando a função ReLU.

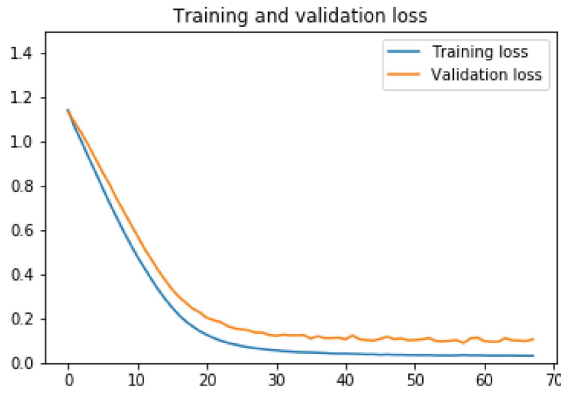

Figura 5.10 – Gráfico das funções de erro do modelo *Secretaria* ao longo do treino no *dataset* em português, usando a função ReLU.

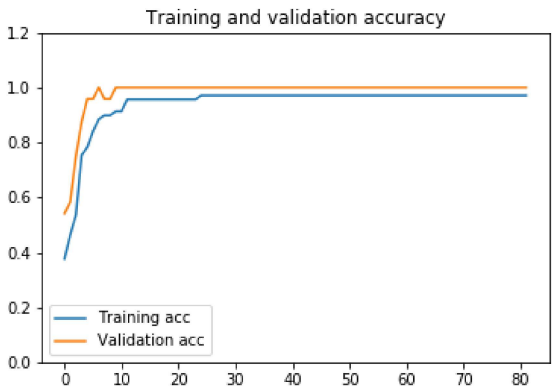

Figura 5.11 – Gráfico das funções de precisão do modelo *Biblioteca* ao longo do treino no *dataset* em português, usando a função ReLU.

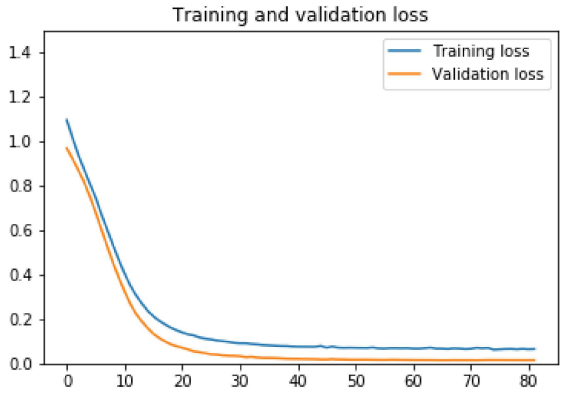

Figura 5.12 – Gráfico das funções de precisão do modelo *Biblioteca* ao longo do treino no *dataset* em português, usando a função ReLU.

Da observação dos gráficos, conclui-se que nenhum dos modelos sofreu *overfitting* ou *underfitting*. Constata-se que na maioria dos modelos, a precisão observada nos dados de treino é maior que a precisão observada nos dados de validação. O que significa que o modelo previu corretamente mais exemplos no conjunto de treino do que exemplos no conjunto de validação. Porém, no modelo *Biblioteca* tal não se verifica, ou seja, o modelo previu corretamente mais exemplos no conjunto de validação do que no conjunto de treino. Uma possível razão para este acontecimento é o facto do conjunto de treino apresentar maior variação frásica nos seus exemplos, o que torna as frases mais difíceis de classificar, resultando numa precisão menor.

De seguida, foi testado o desempenho dos modelos através da construção de matrizes de confusão e calculada a precisão alcançada pelos modelos. A matriz de confusão, ou matriz de erro, é uma matriz que permite a visualização do desempenho de um algoritmo. É uma matriz $n * n$, no qual, n é o número de classes presentes nos dados. A matriz tem duas dimensões, "Real"e "Previsto"em que, normalmente, uma coluna representa a classe "Real"e uma linha representa a classe "Previsto". Porém, também a representação inversa pode ser considerada. Por causa da estrutura da matriz, os elementos na diagonal da matriz de confusão representam classificações

corretas, ou seja, a classe prevista corresponde à classe real. Contrariamente, os elementos que não se encontram na diagonal da matriz representam as classificações incorretas, sendo que a classe prevista não corresponde à classe real (State, 2019). Para proceder a estes novos testes criaram-se novas frases, diferentes das utilizadas para treinar e validar os modelos.

A precisão, calculada com os resultados da matriz, é definida pela Equação, 5.2.

$$Precisão = \frac{Classificações \quad corretas}{Número \quad total \quad de \quad classificações} \tag{5.2}$$

A precisão é, deste modo, o quociente entre as classificações corretas de cada modelo, sobre o número total de classificações executadas.

Na Tabela A.1 é apresentada a matriz de confusão do modelo *Principal*, no *dataset* em português. As classificações corretas do modelo *Principal* encontram-se na diagonal da Tabela A.1. O número total de classificações executadas para o modelo *Principal* foi de 168, sendo que 160 classificações foram corretas, ou seja, o modelo previu corretamente a que *tag*, ou classe, pertencem 160 das 168 frases. Utilizando a Equação 5.2, a precisão do modelo *Principal* no *dataset* em português é de 95%.

A Tabela 5.3 apresenta os resultados do modelo *Cantina*, no *dataset* em português. No modelo *Cantina*, num total de 21 classificações, 19 foram corretas. A precisão calculada do modelo *Cantina*, no *dataset* em português é, por isso, de 90%.

Real / Previsto	**ementa**	**horário**	**localização**
ementa	**6**	0	0
horário	0	**6**	0
localização	1	1	**7**

Tabela 5.3 – Matriz de confusão do modelo *Cantina* no *dataset* em português.

A Tabela 5.4 apresenta os resultados do modelo *Serviços Sociais* no *dataset* em português. O modelo acertou 19 de 21 classificações. Assim a precisão observada no modelo *Serviços Sociais*, no *dataset* em português, é de 90%.

Real / Previsto	info	horário	localização
info	**7**	1	1
horário	0	**6**	0
localização	0	0	**6**

Tabela 5.4 – Matriz de confusão do modelo *Serviços Sociais* no *dataset* em português.

A Tabela 5.5 apresenta o resultado do modelo *Serviços Académicos* no *dataset* em português. De 21 classificações, 20 foram as corretas. Desta forma, a precisão do modelo *Serviços Académicos* no *dataset* em português é de 95%.

Real / Previsto	info	horário	localização
info	**7**	0	1
horário	0	**7**	0
localização	0	0	**6**

Tabela 5.5 – Matriz de confusão do modelo *Serviços Académicos* no *dataset* em português.

A Tabela 5.6 apresenta os resultados do modelo *Secretaria* no *dataset* em português. Do total de 21 classificações, 19 foram corretas. Assim, a precisão do modelo *Secretaria* no *dataset* em português é de 90%.

Real / Previsto	info	horário	localização
info	**6**	1	0
horário	0	**6**	0
localização	1	0	**7**

Tabela 5.6 – Matriz de confusão do modelo *Secretaria* no *dataset* em português.

A Tabela apresenta os 5.7 resultados do modelo *Biblioteca* no *dataset* em português. A precisão do modelo *Biblioteca* no *dataset* em português é de 95%, acertando 20 classificações dum total de 21 classificações.

Real / Previsto	info	horário	localização
info	**7**	0	1
horário	0	**7**	0
localização	0	0	**6**

Tabela 5.7 – Matriz de confusão do modelo *Biblioteca* no *dataset* em português.

Com o resultado das matrizes de confusão, é possível observar que, no mínimo, atingiu-se 90% de precisão, o que é um desempenho satisfatório. Conclui-se que os resultados são satisfatórios para os objetivos desta dissertação.

5.1.2 Treino e Testes no *Dataset* em Inglês

O mesmo método foi aplicado para treinar os modelos em inglês, ou seja, utilizou-se uma única camada interna de neurónios, um *batch size* de 5, que corresponde ao número de exemplos com que o modelo treina antes de atualizar os seus pesos. De seguida, para calcular o número de neurónios a utilizar na camada interior de cada rede neuronal, considerou-se novamente a junção de todos os seguintes pontos:

- O número de neurónios na camada interior deve ser entre o tamanho da camada de entrada e o tamanho da camada de saída.
- O número de neurónios na camada interior deve ser 2/3 do tamanho da camada de entrada mais o tamanho da camada de saída.
- O número de neurónios na camada interior deve ser menos de metade do tamanho da camada de entrada.

De forma semelhante, foram também testadas duas abordagens, uma utilizando a tangente hiperbólica e a outra ReLU, para se encontrar uma solução satisfatória, no qual, se procurou minimizar o valor da função de erro e maximizar o valor da precisão observado nos dados de treino e validação.

Teste com a função tangente hiperbólica

Com a utilização da função tangente hiperbólica, espera-se por isso, um valor de precisão nos dados de validação (Val.ACC) o mais próximo de 1 possível, e um valor da função de erro (Val.LOSS) o mais próximo de 0 possível. Os resultados obtidos podem ser observados na Tabela 5.8.

Modelo	Nº de Neurónios	Train ACC	Train LOSS	Val. ACC	Val. LOSS
Principal	**74**	**0.99**	**0.03**	**0.98**	**0.06**
Cantina	**27**	**0.92**	**0.19**	**0.92**	**0.28**
Serviços Académicos	**28**	**0.93**	**0.06**	**0.88**	**0.19**
Serviços Sociais	**33**	**0.95**	**0.12**	**0.91**	**0.15**
Secretaria	**27**	**0.97**	**0.08**	**0.95**	**0.18**
Biblioteca	**30**	**0.97**	**0.06**	**0.99**	**0.01**

Tabela 5.8 – Tabela dos resultados finais usando a função tangente hiperbólica no *dataset* em inglês.

Teste com a função ReLU

Com a utilização da função ReLU, espera-se um valor de precisão nos dados de validação (Val.ACC) o mais próximo de 1 possível, e um valor da função de erro (Val.LOSS) o mais próximo de 0 possível. Os resultados obtidos podem ser observados na Tabela 5.9.

Modelo	Nº de Neurónios	Train ACC	Train LOSS	Val. ACC	Val. LOSS
Principal	**74**	**0.99**	**0.02**	**0.96**	**0.12**
Cantina	**27**	**0.91**	**0.19**	**0.88**	**0.21**
Serviços Académicos	**28**	**0.95**	**0.10**	**0.92**	**0.25**
Serviços Sociais	**33**	**0.95**	**0.11**	**0.91**	**0.26**
Secretaria	**27**	**0.94**	**0.07**	**0.99**	**0.10**
Biblioteca	**30**	**0.97**	**0.06**	**0.99**	**0.01**

Tabela 5.9 – Tabela dos resultados finais usando a função ReLU no *dataset* em inglês.

Observou-se que utilizando a função de ativação tangente hiperbólica, ocorre um aumento no valor da função de erro nos modelos *Principal*, *Serviços académicos* e *Serviços Sociais*. Quando comparando as duas funções de ativação, os valores de erro verificados no conjunto de treino (Train LOSS) e validação (Val. LOSS) são, na generalidade, mais altos utilizando a função ReLU, o que indica que esta função não se ajustou tão bem aos dados como a função tangente hiperbólica. Este facto contrasta com o sucedido no *dataset* em português, no qual, a função ReLU ajustou-se melhor aos dados. Apesar de haver alguma diferença nos valores finais de precisão e erro nos dados de validação e de treino, não é possível só com esses dados concluir qual a melhor função a utilizar neste caso.

Durante as experiências, observou-se também que a função tangente hiperbólica apresentou valores mais consistentes que a função ReLU que variou muito os seus valores ao longo das experiências. Por esta razão, a tangente hiperbólica foi escolhida como função de ativação para treinar as redes neuronais no *dataset* em inglês.

São apresentados os gráficos correspondem aos resultados de cada rede neuronal treinada utilizando a tangente hiperbólica no *dataset* em inglês.

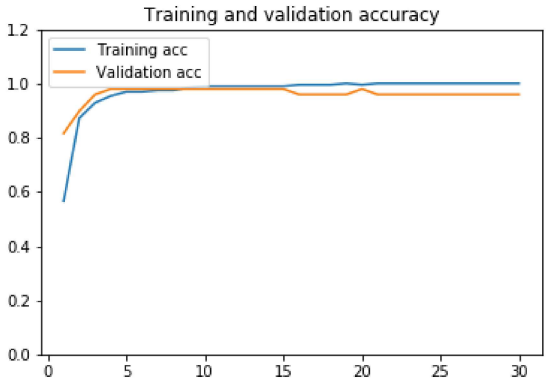

Figura 5.13 – Gráfico das funções de precisão do modelo *Principal* ao longo do treino, no *dataset* em inglês, usando a função tanh.

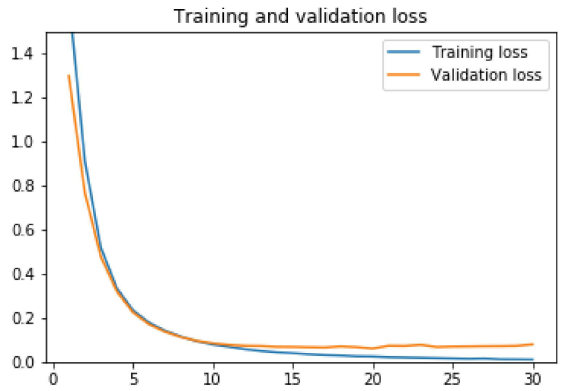

Figura 5.14 – Gráfico das funções de erro no modelo *Principal* ao longo do treino, no *dataset* em inglês, usando a função tanh.

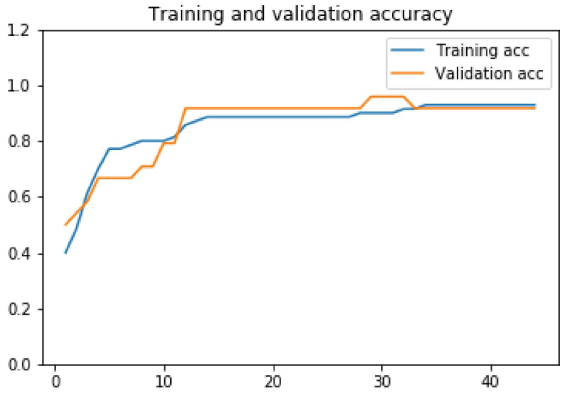

Figura 5.15 – Gráfico das funções de precisão do modelo *Cantina* ao longo do treino, no *dataset* em inglês, usando a função tanh.

Figura 5.16 – Gráfico das funções de erro no modelo *Cantina* ao longo do treino, no *dataset* em inglês, usando a função tanh.

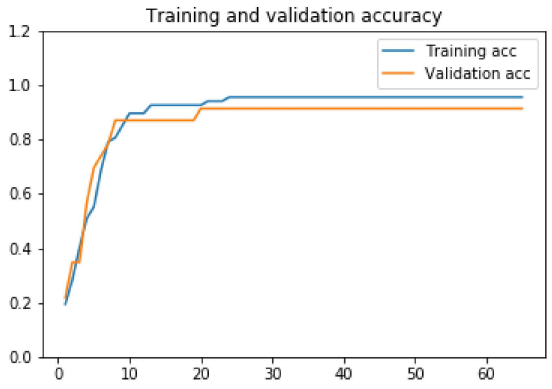

Figura 5.17 – Gráfico das funções de precisão do modelo *Serviços Sociais* ao longo do treino, no *dataset* em inglês, usando a função tanh.

Figura 5.18 – Gráfico das funções de erro do modelo *Serviços Sociais* ao longo do treino, no *dataset* em inglês, usando a função tanh.

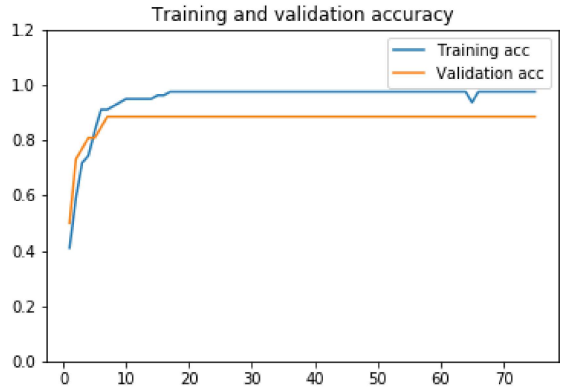

Figura 5.19 – Gráfico das funções de precisão do modelo *Serviços Académicos* ao longo do treino, no *dataset* em inglês, usando a função tanh.

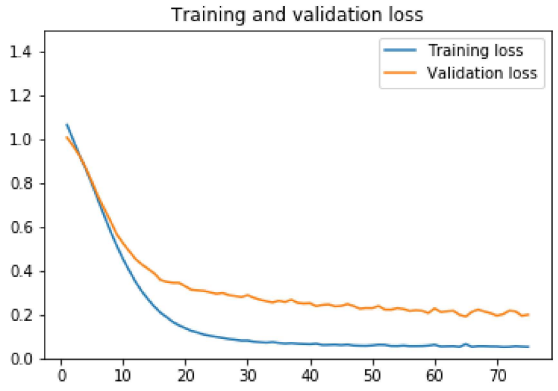

Figura 5.20 – Gráfico das funções de erro do modelo *Serviços Académicos* ao longo do treino, no *dataset* em inglês, usando a função tanh.

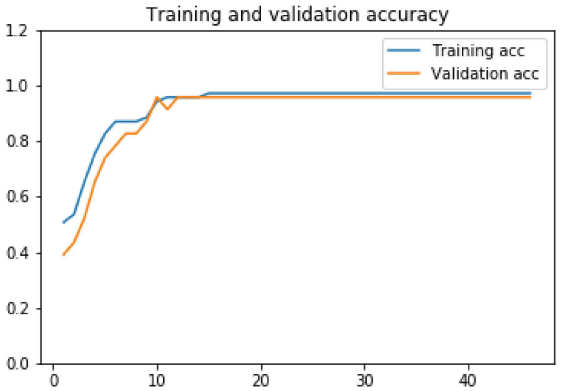

Figura 5.21 – Gráfico das funções de precisão do modelo *Secretaria* ao longo do treino, no *dataset* em inglês, usando a função tanh.

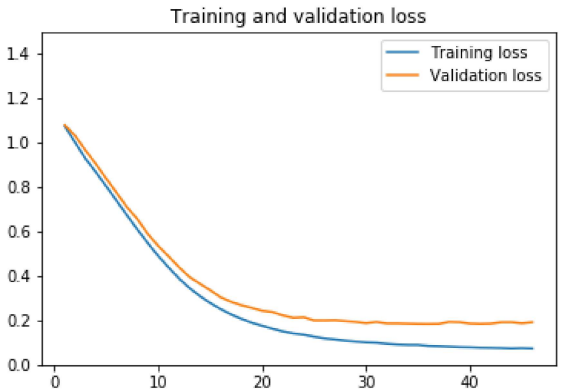

Figura 5.22 – Gráfico da função de erro do modelo *Secretaria* ao longo do treino, no *dataset* em inglês, usando a função tanh.

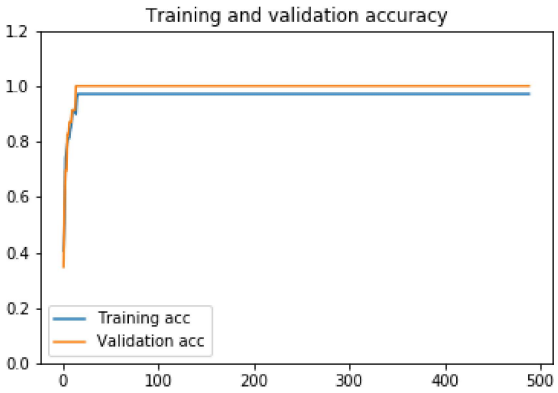

Figura 5.23 – Gráfico das funções de precisão do modelo *Biblioteca* ao longo do treino, no *dataset* em inglês, usando a função tanh.

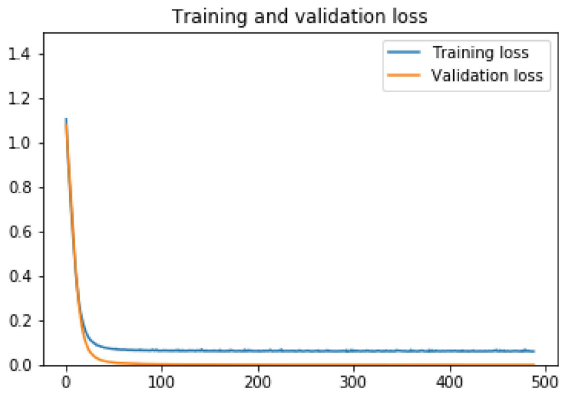

Figura 5.24 – Gráfico da função de erro do modelo *Biblioteca* ao longo do treino, no *dataset* em inglês, usando a função tanh.

Da observação dos gráficos, conclui-se que os modelos apresentam resultados satisfatórios, ou seja, não existe *underfitting* nem *overfitting*. É de notar, no entanto, que o modelo *Cantina* apresentou uma pequena tendência para sofrer *overfitting*, pois a partir da iteração 20, o valor da função de erro de validação começa a aumentar e o valor da função de erro de treino continua a diminuir. Para prevenir o *overfitting* utilizou-se uma técnica designada de Early Stopping, que consiste em parar o processo de treino antes que ocorra *overfitting*.

Mais uma vez, o modelo *Biblioteca* apresentou um valor de precisão no conjunto de validação maior que no conjunto de treino tal como aconteceu para o *dataset* em português. Uma possível explicação é porque o modelo tem mais facilidade em classificar os dados de treino que os dados de validação, devido a existir maior diversidade frásica dos exemplos presentes no conjunto de treino do que nos exemplos presentes no conjunto de validação, resultando numa precisão menor.

Em suma, conclui-se que os modelos treinados no *dataset* em inglês têm um desempenho pior que os modelos treinados no *dataset* em português. Esta diferença pode estar associada ao pré-processamento de dados, no qual está a ser retirada demasiada informação dos dados. Existem duas causas:

- O algoritmo de *stemming*, *Snowball*, está a retirar demasiada informação de

palavras essenciais no *dataset* em inglês.

- Remoção de *stop-words* muito intensiva. Em 4.1.2 explica-se como funciona o processo de remoção de *stop-words*, mas, resumidamente, remove-se palavras que são consideradas não essenciais para a classificação. No entanto, nesta dissertação utilizou-se um conjunto de *stop-words* fixo, disponibilizado pela biblioteca NLTK, ou seja, pode estar a remover-se palavras que são essenciais neste *dataset* em inglês.

De seguida, testou-se o desempenho do *dataset* em inglês recorrendo a matrizes de confusão para todos os modelos criados. A Tabela A.2 apresenta a matriz de confusão do modelo *Principal* no *dataset* em inglês. O modelo de 168 classificações feitas, 157 foram corretas. Deste modo, utilizando a Equação 5.2, a precisão observada do modelo *Principal* no *dataset* em inglês é de 93%.

Na Tabela 5.10 é apresentada a matriz de confusão do modelo *Cantina* no *dataset* em inglês. O modelo acertou 19 de 21 classificações alcançando, por isso, uma precisão de 90%.

Previsto \ Real	**Menu**	**Schedule**	**Location**
Menu	**6**	1	0
Schedule	0	**6**	0
Location	1	0	**7**

Tabela 5.10 – Matriz de confusão de *Cantina* no *dataset* em inglês.

A Tabela 5.11 mostra matriz de confusão do modelo *Serviços Sociais*. Todas as 21 classificações feitas pelo modelo foram corretas, o que, de acordo com a Equação 5.2 indica que a precisão do modelo *Serviços Sociais* é de precisão de 100%. Naturalmente, este valor baixa se mais classificações forem feitas.

Previsto \ Real	Info	Schedule	Location
Info	**7**	0	0
Schedule	0	**7**	0
Location	0	0	**7**

Tabela 5.11 – Matriz de confusão de *Serviços Sociais* no *dataset* em inglês.

A Tabela 5.12 mostra matriz de confusão do modelo *Serviços Académicos* no *dataset* em inglês. O modelo acertou 19 de 21 classificações. O que se traduz numa precisão de 90%.

Previsto \ Real	Info	Schedule	Location
Info	**6**	0	1
Schedule	0	**7**	0
Location	1	0	**6**

Tabela 5.12 – Matriz de confusão dos *Serviços Académicos* no *dataset* em inglês.

A Tabela 5.13 mostra matriz de confusão do modelo *Secretaria* no *dataset* em inglês. O modelo acertou 16 classificações de 21. Assim, a precisão observada no modelo *Secretaria* é de 76%.

Previsto \ Real	Info	Schedule	Location
Info	**7**	3	2
Schedule	0	**4**	0
Location	0	0	**5**

Tabela 5.13 – Matriz de confusão de *Secretaria* no *dataset* em inglês.

A Tabela 5.14 mostra matriz de confusão do modelo *Biblioteca* no *dataset* em inglês. O modelo acertou 19 de 21 classificações e, por isso, a precisão observada foi de 90%.

Previsto \ Real	Info	Schedule	Location
Info	**7**	1	1
Schedule	0	**6**	0
Location	0	0	**6**

Tabela 5.14 – Matriz de confusão da *Biblioteca* no *dataset* em inglês.

O pior resultado obtido nas tabelas de confusão foi o do modelo *Secretaria*, na Tabela 5.13, com uma precisão de 76%. Conclui-se, que o resultado dos modelos, baseado em todos os testes executados, é satisfatório para os objetivos desta dissertação.

5.2 Testes ao Funcionamento da Aplicação

Nesta secção são abordados os testes e resultados ao funcionamento da aplicação na plataforma *Messenger*.

Como dito anteriormente, quando um utilizador ainda não está inserido na base de dados uma *quick reply* é enviada pelo *chatbot*, para solicitar ao utilizador permissão para ser armazenado o seu PSID e introduzir as funcionalidades do *chatbot*, Figura 5.25. A primeira interação com o utilizador é feita em inglês, pois permite que mais alunos percebam o propósito do *chatbot*. Ainda na primeira *quick reply* recebida é especificado como o utilizador pode alternar entre os idiomas disponíveis, entre português e inglês.

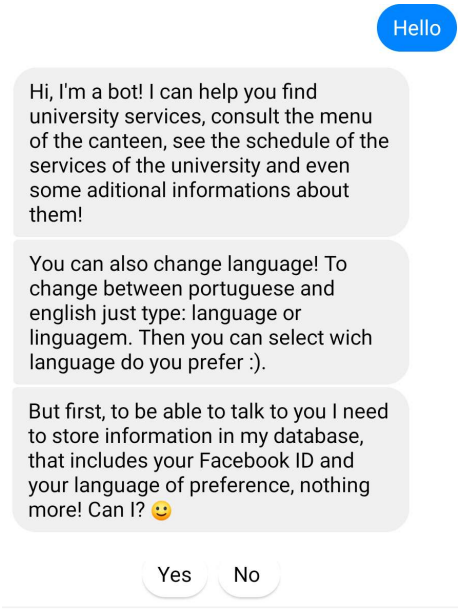

Figura 5.25 – Primeira *Quick Reply* enviada ao utilizador.

De seguida se houver concordância do utilizador em armazenar o seu PSID, é iniciada a conversa, conforme a Figura 5.26.

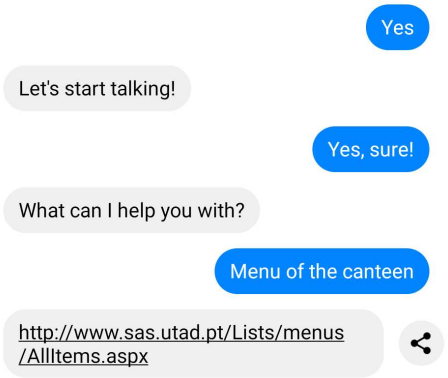

Figura 5.26 – Exemplo do início da conversação com utilizador.

Uma funcionalidade do *chatbot* é a possibilidade de mudar o idioma de conversação. A Figura 5.27 apresenta o *template* recebido pelo utilizador, quando esta introduz a palavra *"language"* ou "linguagem"na conversa.

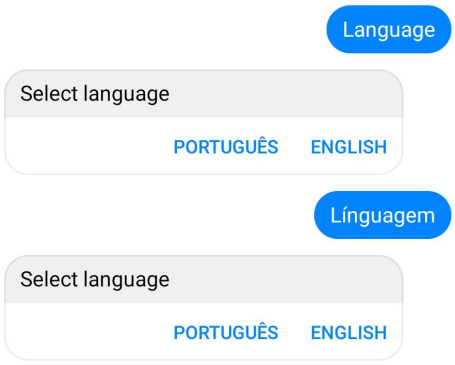

Figura 5.27 – *Template* usado para mudar o idioma.

Durante a conversa, o *chatbot* faz o processamento e classificação de cada frase. As Figuras 5.28 e 5.29, apresentam um exemplo de uma conversa, em português e inglês respetivamente. Pode-se observar das conversas apresentadas que o *chatbot* apresenta um desempenho bastante satisfatório, respondendo corretamente a todas as perguntas apresentadas em ambos os idiomas.

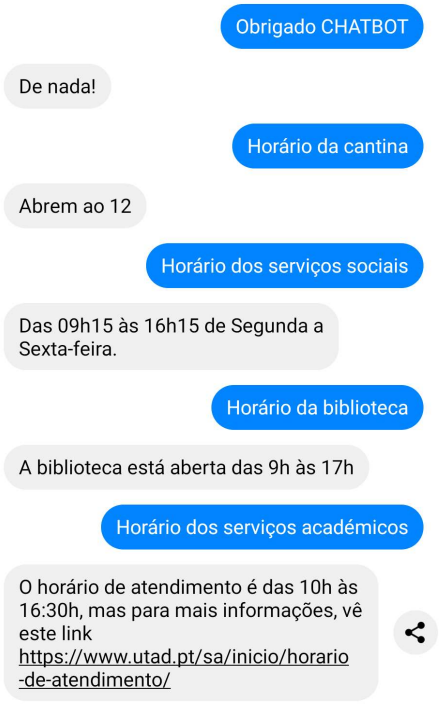

Figura 5.28 – Exemplo de uma conversa com o *chatbot* em português.

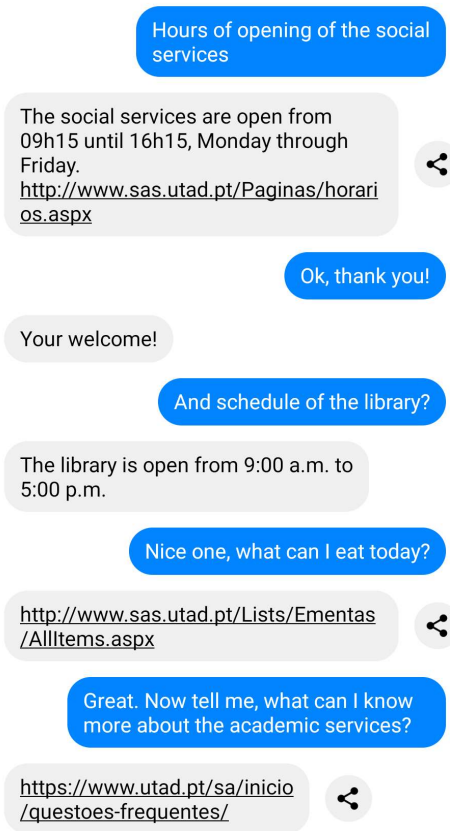

Figura 5.29 – Exemplo de uma conversa com o *chatbot* em inglês.

Observa-se que é possível manter uma conversa natural com o *chatbot*, porém é percetível que todas as interações são *one-shot queries*, ou seja, o utilizador e o *chatbot* participam numa conversa baseada em turnos. Assim, numa situação em que o utilizador insere mais que uma mensagem consecutiva, o *chatbot* apenas consegue abordar a última mensagem.

No caso em que as duas *tags* retiradas duma frase são, por exemplo, "secretaria", correspondente à primeira classificação e a "localização"correspondente à segunda classificação, um pedido de partilha de localização deve ser feito e um *template* com as direções deve ser enviado, como se observa na Figura 5.30.

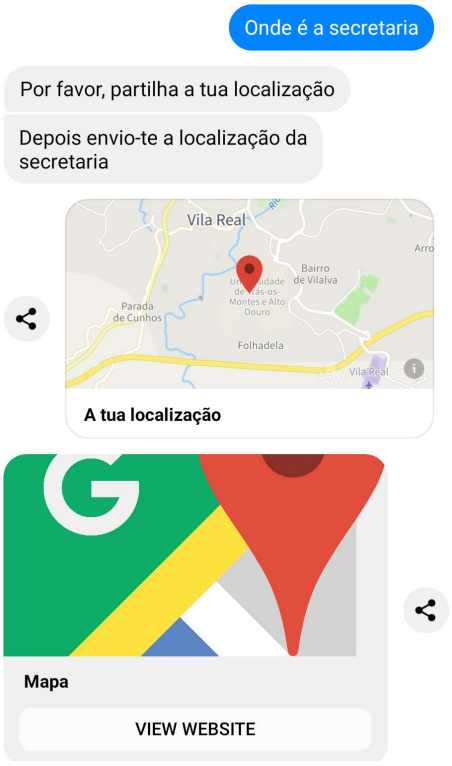

Figura 5.30 – *Template* com as direções no mapa enviado ao utilizador.

Existem, no entanto, situações nas quais o utilizador pergunta algo fora do contexto do *chatbot*, neste caso é recebida a mensagem apresentada nas Figuras 5.31 e 5.32.

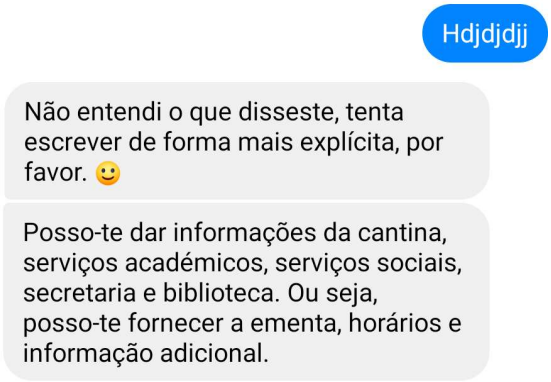

Figura 5.31 – Frase fora do contexto do *chatbot* e respetiva resposta em português.

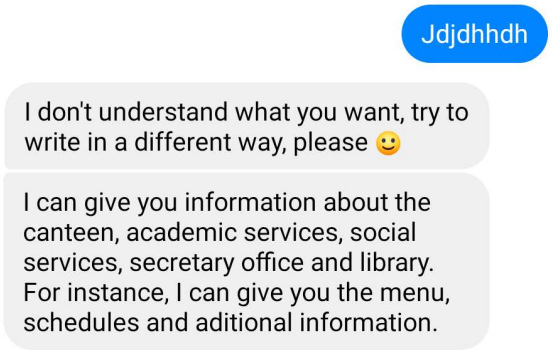

Figura 5.32 – Frase fora do contexto do *chatbot* e respetiva resposta em inglês.

O *chatbot* funciona como esperado, porém, existe um problema relacionado com a

comunicação entre o Heroku e o Facebook Messenger. Quando uma mensagem é escrita no *chat*, o servidor do Facebook Messenger envia um pedido POST para o servidor que contém o *chatbot*, neste caso, o servidor do Heroku. Se o servidor do Facebook Messenger não obtém uma resposta, reenvia o pedido. Existem certas limitações em utilizar o Heroku gratuitamente, como a quantidade de memória disponível, a velocidade do serviço e também a disponibilidade de resposta do *chatbot*. Por esta razão, se não ocorrer uma interação entre o *chatbot* e qualquer utilizador durante 30 minutos o *chatbot* entra num modo *sleep*, que implica que quando um utilizador entra novamente em contacto com o *chatbot* são necessários 15 segundos para reiniciar novamente todos os seus serviços. Enquanto o servidor do Heroku reinicia o *chatbot* já foram recebidos 3 pedidos provenientes do servidor do Facebook Messenger e, como consequência, nesta situação são recebidas, pelo utilizador, 3 respostas do *chatbot*, Figura 5.33.

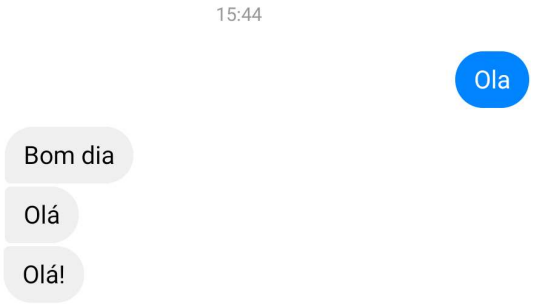

Figura 5.33 – Mensagens repetidas recebidas pelo utilizador.

6 Conclusão e trabalho futuro

Nesta dissertação foi projetado um *chatbot* com a capacidade de reconhecer uma intenção da frase recebida e fornecer informações adequadas sobre a UTAD. É principalmente dirigido a novos alunos, porém, qualquer pessoa visitante da Universidade pode encontrar valor no seu uso. Devido à sua arquitetura, é possível treiná-lo com outras frases, noutros contextos, daí poder ser utilizado noutras universidades, ou noutro tipo de serviços de apoio ao cliente. O *chatbot* utiliza técnicas de NLP e *Machine learning* para classificar uma frase e retirar uma intenção de forma a fornecer a melhor resposta possível.

Neste capítulo são apresentadas as conclusões retiradas ao longo do desenvolvimento desta dissertação, bem como algumas melhorias e propostas de trabalho futuro que poderão melhorar a solução proposta.

6.1 Conclusões

A grande afluência de novos alunos provoca um grande congestionamento nos serviços da UTAD, tornando o tempo de espera muito demorado em certos serviços. Desta forma, foi necessário criar uma solução para aliviar os funcionários da universidade

de questões de carácter simples mas muito importantes para os novos alunos e que podem ser respondidas por um sistema autónomo.

O sistema implementado tem que ser capaz de responder a qualquer hora, de ser intuitivo e fácil de usar, não implicar muito esforço por parte do aluno para poder utilizar o sistema e por fim, tem que ser fiável. Um *chatbot* que implementa tecnologias de NLP e *Machine learning* foi a proposta dada.

Inicialmente, foi feita uma revisão à evolução dos *chatbots* ao longo dos últimos anos, à tecnologia em que se baseiam e os seus diferentes tipos de implementações. Realizou-se uma análise a *chatbots* implementados por outros autores que resolvem problemas de carácter semelhante, ou seja, problemas de apoio ao utilizador. Porém, estas implementações estudadas divergiram bastante na tecnologia utilizada, no qual, em certos trabalhos utilizaram plataformas de criação de *chatbots* em noutros *Machine learning*.

No capítutlo 3 procedeu-se à revisão da teoria de NLP e *Machine learning* e foram explicados os conceitos necessários para a implementação do *chatbot*. De seguida, foi efetuada a apresentação das tecnologias utilizadas nesta dissertação, como bibliotecas, apresentação da plataforma Facebook Messenger, do serviço Heroku e da base de dados MongoDB.

Após a apresentação das tecnologias utilizadas, abordou-se a conceção e implementação do *chatbot*, no qual foram identificados os componentes do *chatbot*. O *chatbot* proposto, denominado Flask_Chatbot permite uma interação através do Facebook Messenger em que a cada mensagem de texto recebida é feito o seu pré-processamento, utilizando técnicas de NLP e, de seguida, realiza-se a sua classificação. Esta é feita através de redes neuronais, que permitem escolher a resposta adequada. O *chatbot* foi treinado em dois *datasets* um com frases portuguesas e outro com frases inglesas. Cada *dataset* contém frases relativas à cantina, aos serviços sociais, aos serviços académicos, à secretaria e à biblioteca e consegue fornecer informações sobre os horários, localização e ementa. Para preparar ambos *datasets* analisou-se a distribuição dos dados das diferentes páginas de Excel, abordou-se a extração e pré-processamento destes dados e, por fim, procedeu-se à criação dos

modelos que classificam qualquer frase introduzida pelo utilizador. Após explicado o funcionamento do *chatbot*, os processos de tratamento dos dados e o treino dos modelos, explicou-se as etapas necessárias para a integração do *chatbot* na plataforma Facebook Messenger, explicou-se a função da base de dados implementada em MongoDB e, por fim, o funcionamento propriamente dito da aplicação.

Para finalizar, foram realizados alguns testes aos modelos utilizados para a classificação e às funcionalidades do *chatbot*, com o objetivo de observar os valores da precisão e perda dos modelos, de forma a alcançar um desempenho aceitável. Por isso, analisou-se os valores de treino e validação testando as duas funções de ativação ReLU e tangente hiperbólica. Conclui-se que no *dataset* português, o melhor resultado obtido foi utilizando a função de ativação ReLU e, no *dataset* inglês, o melhor resultado obtido foi utilizando a tangente hiperbólica. Depois de se verificar a melhor funcão de ativação para cada *dataset*, construiram-se matrizes de confusão para testar o desempenho dos modelos com novas frases. Das matrizes de confusão observou-se que a menor precisão obtida no *dataset* português foi de 90% pelos modelos *Secretaria*, *Cantina* e *Serviços Sociais*. Já no *dataset* inglês, o menor valor de precisão observado nas matrizes de confusão foi de 76%, obtido no modelo *Secretaria*.

Pode-se afirmar que os modelos apresentaram todos um desempenho satisfatório e que, mesmo mantendo a mesma arquitetura, o desempenho de um modelo é muito influenciado pelo *dataset* utilizado para treinar. Por fim, observou-se que a mudança de função de ativação também influencia o desempenho final de uma rede neuronal e que, quando esta apresenta melhor desempenho é porque a função de ativação ajustou-se melhor aos dados utilizados para treinar.

Relativamente ao funcionamento do *chatbot* pode-se afirmar que funciona bem e cumpre com os objetivos da dissertação, ou seja, é capaz de interagir com o utilizador por meio de uma conversa no Facebook Messenger, é capaz de conversar em dois idiomas e fornece a localização dos edifícios, ementa e horários. Em suma, conclui-se que todos os objetivos da dissertação foram concluídos porém, existem melhorias que podem ser feitas que são apresentadas de seguida.

6.2 Trabalho Futuro

Após a conclusão deste trabalho é de salientar que muitos aspetos podem ser melhorados, para obter uma abordagem com um desempenho mais elevado e mais robusta.

- O *chatbot* ser capaz de manter contexto de uma conversa. No momento presente, o *chatbot* é capaz de classificar uma frase. Porém, se a frase seguinte de alguma forma mantém um contexto relacionado com a frase anterior, o *chatbot* não classifica corretamente, por exemplo, as frases:

  "Onde são os serviços sociais?"

  "E a que horas abrem?"

  Estas duas frases estão relacionadas, qualquer pessoa consegue perceber que a segunda frase se refere aos serviços sociais, porém, na atual implementação, o *chatbot* não é capaz de as relacionar.

- Implementar o *chatbot* noutras plataformas, como Telegram, WhatsApp e Skype. Apesar do Facebook Messenger ser a plataforma atualmente de *messaging* mais utilizada em Portugal, outras plataformas estão a emergir e a ter alguma relevância, desta forma, é sensato apresentar uma solução capaz de funcionar em múltiplas plataformas.

- Utilizar mais dados para treinar os modelos. Pela análise dos modelos, observou-se que se ajustaram bem aos dados, porém, são modelos bastante simples que, por sua vez, limita a capacidade de generalizar. Assim, com uma quantidade maior de dados é possível tornar o *chatbot* mais capaz de classificar vários tipos de frase.

- Melhorar o processo de remoção de *stop-words*. A implementação efetuada recorre ao conjunto fixo de *stop-words* tanto para inglês como para português disponibilizado pela biblioteca NLTK, no entanto, esta solução não é ideal pois pode-se remover palavras que podem ser de interesse.

- Utilizar outro algoritmo de *stemming*. Nesta dissertação utilizou-se o algoritmo de *stemming* Snowball. Porém, existem outras alternativas que podem oferecer melhor desempenho, como o algoritmo Lancaster.

- O *chatbot* ser capaz de guardar a localização do utilizador por um período curto de tempo. Desta forma, quando duas questões consecutivas são colocadas ao *chatbot* sobre a localização de edifícios, este não necessita de pedir a localização do utilizador duas vezes.

- Hospedar o *chatbot* num servidor próprio certificado. Utilizou-se uma subscrição gratuita no serviço Heroku, sendo suficiente para a implementação do protótipo, porém, limita o desenvolvimento de qualquer aplicação que seja utilizada em grande escala.

- Fornecer a localização de salas e auditórios.

- Guardar a sessão do utilizador para não ser necessário a cada mensagem fazer uma *query* à base de dados para confirmar que utilizar se trata e o seu idioma.

Referências bibliográficas

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. 59

Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*. 42

AISB (2018). Loebner prize chatbot. https://web.archive.org/web/20181226234858/http://www.aisb.org.uk/events/loebner-prize#finals2018. [Online, consultado a 30 de setembro de 2019]. 9

Alves, F. (2018). Chatbot para serviços bancários. 11, 12, 19, 21

Andrej Krenker, J. B. and Kos, A. (2011). *Introduction to the Artificial Neural Networks- Methodological Advances and Biomedical Applications*. 38, 39, 41, 46

Bergholm, R. (2018). Development of a Facebook Messenger chatbot application for social media event discovery. 61

Brownlee, J. (2019a). Difference between a batch and an epoch in a neural network. 80

Brownlee, J. (2019b). Gentle introduction to the adam optimization algorithm for deep learning. 80

Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004). Special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6. 67

Chollet, F. et al. (2015a). Keras. https://keras.io/optimizers/. [Online, consultado a 11 de setembro de 2019]. 80

Chollet, F. et al. (2015b). Keras. https://keras.io/activations/. [Online, consultado a 12 de setembro de 2019]. 81

Chollet, F. et al. (2019). Keras. https://keras.io. [Online, consultado a 14 de agosto de 2019]. 60

Cilimkovic, M. (2015). Neural networks and back propagation algorithm. *Institute of Technology Blanchardstown, Blanchardstown Road North Dublin*, 15. 46

Clark, M. (2016). A chatbot framework. https://info.contactsolutions.com/digital-engagement-blog/a-chatbot-framework. [Online, consultado a 18 de julho de 2019]. 15

Dominic (2019). Understand tensorflow. https://medium.com/@d3lm/understand-tensorflow-by-mimicking-its-api-from-scratch-faa55787170d. [Online, consultado 3 de dezembro de 2019]. 59, 60

El Assaf, A., Zaidi, S., Affes, S., and Kandil, N. (2016). Robust anns-based wsn localization in the presence of anisotropic signal attenuation. *IEEE Wireless Communications Letters*, 5(5):504–507. 46

Facebook (2019a). Button. https://developers.facebook.com/docs/messenger-platform/send-messages/template/button. [Online, consultado 20 de agosto de 2019]. 62, 63

Facebook (2019b). Conversation. https://developers.facebook.com/docs/messenger-platform/introduction/conversation-components. [Online, consultado 19 de Agosto de 2019]. 53, 62, 63

Facebook (2019c). Generic. https://developers.facebook.com/docs/messenger-platform/send-messages/template/generic. [Online, consultado 20 de Agosto de 2019]. 62

Facebook (2019d). Introduction. https://developers.facebook.com/docs/messenger-platform/introduction. [Online, consultado 18 de agosto de 2019]. 73

Facebook (2019e). Messages. https://developers.facebook.com/docs/messenger-platform/reference/webhook-events/messages. [Online, consultado 19 de Agosto de 2019]. 63

Facebook (2019f). Postbacks. https://developers.facebook.com/docs/messenger-platform/reference/webhook-events/messaging_postbacks. [Online, consultado 19 de agosto de 2019]. 63

Facebook (2019g). Setting up your aplication. https://developers.facebook.com/docs/messenger-platform/getting-started/app-setup. [Online, consultado 21 de agosto de 2019]. 75, 76

Facebook (2019h). Setting up your webhook. https://developers.facebook.com/docs/messenger-platform/getting-started/webhook-setup. [Online, consultado 19 de agosto de 2019]. 74, 75

Facebook (2019i). Webhook events. https://developers.facebook.com/docs/messenger-platform/reference/webhook-events. [Online, consultado 19 de agosto de 2019]. 63

George K, S. and Joseph, S. (2014). Text classification by augmenting bag of words (bow) representation with co-occurrence feature. *IOSR Journal of Computer Engineering*, 16:34–38. 30

Google (2013). Word2vec. https://code.google.com/archive/p/word2vec/. 48

Goyal, P., Pandey, S., and Jain, K. (2018). *Deep Learning for Natural Language Processing Creating Neural Networks*. 46, 48

Graziani, T. (2016). 5 wechat accounts using artificial intelligence. https://walkthechat.com/5-wechat-accounts-using-artificial-intelligence/. [Online, consultado a 15 de outubro de 2019]. 12

Gregor, K., Danihelka, I., Graves, A., Rezende, D. J., and Wierstra, D. (2015). Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*. 49

Gurney, K. (2014). *An introduction to neural networks*. CRC press. 38

Haddi, E., Liu, X., and Shi, Y. (2013). The role of text pre-processing in sentiment analysis. *Procedia Computer Science*, 17:26–32. 27

Hahnloser, R., Sarpeshkar, R., Mahowald, M., Douglas, R., and Seung, H. (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405:947–51. 41

Heroku (2019). Heroku. https://www.heroku.com. [Online, consultado 22 de agosto de 2019]. 74

Heroku (2020). Configuration and config vars. https://devcenter.heroku.com/articles/config-vars. [Online, consultado 29 de abril de 2020]. 75

Herrup, R. W. W. and Karl (1988). The control of neuron number. 37

Hijazi, S., Kumar, R., and Rowen, C. (2015). Using convolutional neural networks for image recognition. *Cadence Design Systems Inc.: San Jose, CA, USA*. 47, 48

Ikonomakis, M., Kotsiantis, S., and Tampakas, V. (2005). Text Classification Using Machine Learning Techniques. 4(8):966–974. 33

Islam, M. and Morimoto, T. (2015). Performance prediction of solar collector adsorber tube temperature using a nonlinear autoregressive model with exogenous input. *International Journal of Computer Applications*, 114:24–32. 80

Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the Sixteenth International Conference on Machine Learning*, ICML '99, pages 200–209, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc. 29, 33

Jordan, J. (2018). Evaluating a machine learning model. https://www.jeremyjordan.me/evaluating-a-machine-learning-model/. [Online, consultado 2 de novembro de 2019]. 80, 83

Karlik, B. and Olgac, A. V. (2011). Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4):111–122. 41

Kim, Y. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*. 48

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980. 80

Kuhn, T. and Buitenhek, R. (2017). Designing a Dutch financial chatbot Applying natural language processing and machine learning techniques to retrieval-based question answering. 7, 10, 14, 25, 26, 28, 29, 30, 31, 32, 33, 35, 36, 43

Kuyven, N., Vanzin, V., Antunes, C., Cemin, A., da Silva, J. L. T., and Tarouco, L. M. R. (2018). Mapeamento de problemas trigonométricos usando deep learning. In *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)*, volume 29, page 1513. 8

Landsteiner, N. (2005). Eliza. https://www.masswerk.at/elizabot/eliza.html. [Online, consultado a 23 de outubro de 2019]. 10

Lecun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. 36, 47, 48

Liddy, E. D. (2001). Natural language processing. In Marcel Decker, Inc., editor, *Natural Language Processing*. Encyclopedia of Library and Information Science, NY, 2nd edition. 28, 30, 32

Lilleberg, J. and Yanqing Zhang, Y. Z. (2015). Support Vector Machines and Word2vec for Text Classification with Semantic Features. pages 136–140. xix, 33, 34

Longadge, R. and Dongre, S. (2013). Class imbalance problem in data mining review. *CoRR*, abs/1305.1707. 67

Loper, E. and Bird, S. (2019). Nltk. https://www.nltk.org/. [Online, consultado a 22 de agosto de 2019]. 20, 70

Lovins, J. B. (1968). Development of a stemming algorithm. *Mech. Translat. & Comp. Linguistics*, 11(1-2):22–31. 29

Marquez, R., Pedro, H. T., and Coimbra, C. F. (2013). Hybrid solar forecasting method uses satellite imaging and ground telemetry as inputs to anns. *Solar Energy*, 92:176–188. 47

Martins, T. M. et al. (2018). Aplicação de técnicas de inteligência artificial na detecção do comportamento tipo-depressivo em camundongos através do teste de suspensão pela cauda. 36

Mathworks (2019). Tanh function. https://www.mathworks.com/help/matlab/ref/tanh.html. [Online, consultado a 16 de Setembro de 2019]. 41

Mauldin, M. L. (1994). Chatterbots, Tinymuds, and the Turing Test Entering the Loebner Prize Competition. *Aaai*, 94:16–21. 9, 10

Mctear, M. (2018). Conversational Modelling for Chatbots : Current Approaches and Future Directions. Technical report, Ulster University, Ireland. 8, 16, 17

Mikheev, A. (2003). Text segmentation. In *The Oxford Handbook of Computational Linguistics 2nd edition*. 29

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119. 33

Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2018). *Foundations of machine learning*. MIT press. 35, 36, 37

MongoDB (2006). Mongodb. https://www.mongodb.com/what-is-mongodb. [Online, consultado 14 de agosto de 2019]. 76

NLTK (2019a). Snowball stemmer. https://www.nltk.org/_modules/nltk/stem/snowball.html. [Online, consultado 27 de agosto de 2019]. 70

NLTK (2019b). Tokenizer. https://www.nltk.org/api/nltk.tokenize.html. [Online, consultado 27 de agosto de 2019]. 70

Nwankpa, C., Ijomah, W., Gachagan, A., and Marshall, S. (2018). Activation functions: Comparison of trends in practice and research for deep learning. *CoRR*, abs/1811.03378. 41

Ognjanovski, G. (2019). Everything you need to know about neural networks and backpropagation - machine learning easy and fun. https://towardsdatascience.com/everything-you-need-to-know-about-neural-networks-and-backpropagation-machine-learning-made-easy-e5285bc2be3a. 43

Oliphant, T. (2006). Numpy. https://www.numpy.org/. [Online, consultado a 14 de agosto de 2019]. 71

Pang, W. (2018). A Machine Learning Approach for Aspect-based Sentiment Analysis on Social Media. 33, 37, 47, 59, 61

Peltarion (2019). Categorical crossentropy. https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy. 80

Peters, F. (2018). Design and implementation of a chatbot in the context of customer support. 7, 10, 11, 22, 23, 24

Porter, M. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137. 29

Porter, M. (2001). http://snowball.tartarus.org/texts/introduction.html. [Online, consultado 11 de outubro de 2019]. 29

Porto Editora (2018). chatbot in dicionário infopédia da língua portuguesa. https://www.infopedia.pt/dicionarios/lingua-portuguesa/chatbot. [Online, consultado a 22 de abril de 2019]. 7

Rebala, G., Ravi, A., and Churiwala, S. (2019). *Natural Language Processing*, pages 117–125. 31, 32, 33

Ronacher, A. (2019). Flask. https://flask.palletsprojects.com/en/1.1.x/. [Online, consultado a 22 de agosto de 2019]. 74

Rush, A. M., Weston, J., and Chopra, S. (2015). A Neural Attention Model for Sentence Summarization. (September):379–389. 37

Silva, C. and Ribeiro, B. (2003). The importance of stop word removal on recall values in text categorization. In *Proceedings of the International Joint Conference on Neural Networks, 2003.*, volume 3, pages 1661–1666 vol.3. 29, 30

Silva, J. (2018). Interactive bot to support the use of the UPTEC intranet. 8, 13, 14, 18, 19

Simeone, O. et al. (2018). A brief introduction to machine learning for engineers. *Foundations and Trends in Signal Processing*, 12(3-4):200–431. 35, 36, 37

SimilarWeb (2018). Similarweb. https://www.similarweb.com/blog/mobile-messaging-app-map-january-2019. [Online, consultado 18 de julho de 2019]. 4

Simon Haykin (2009). *Neural Networks and Learning Machines Third Edition*. 37, 38, 40, 45

State, G. H. (2019). Error matrix. http://gsp.humboldt.edu/OLM/Courses/GSP_216_Online/lesson6-2/error-matrix.html. [Online, consultado 3 de Novembro de 2019]. 87

statista (2019). statista. https://www.statista.com/statistics/483255/number-of-mobile-messaging-users-worldwide/. [Online, consultado 18 de julho de 2019]. 3

Stoner, D. J., Ford, L., and Ricci, M. (2004). Simulating military radio communications using speech recognition and chat-bot technology. *The Titan Corporation, Orlando*. 12, 13

Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024. 48, 49

Svozil, D., Kvasnicka, V., and Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62. 45

Turing, A. (1950). *Computing Machinery and Intelligence*. Number Mind 49. 9

Unruh, A. (2019). What is the tensorflow machine intelligence platform? https://opensource.com/article/17/11/intro-tensorflow. [Online, consultado 3 de dezembro de 2019]. 61

UTAD (2017). Utad. https://www.utad.pt/universidade/utad-em-numeros-2017/. [Online, consultado 17 de julho de 2019]. 2

UTAD (2018). Utad. https://noticias.utad.pt/blog/2018/09/28/historicos-entrada-estudantes/. [Online, consultado 17 de julho de 2019]. 2

UTAD (2019). Utad. https://noticias.utad.pt/blog/2019/09/09/utad-aumentar-numero-destudantes/. [Online, consultado 18 de setembro de 2019]. 2

Vlad, S. (2005). On the prediction methods using neural networks. In *6th International Symposium of Hungarian Researchers on Computational Intelligence, Budapest*. Citeseer. 46

Weizenbaum, J. (1966). ELIZA A Computer Program For the Study of Natural Language Communication Between Man And Machine. *Communications of ACM*, 9(January 1966):36–25. 10

Wiggins, A. (2017). Twelve factor app. https://12factor.net/. [Online, consultado 20 de Abril de 2020]. 75

Wu, Y., Li, Z., Wu, W., and Zhou, M. (2016). Response selection with topic clues for retrieval-based chatbots. *Neurocomputing*. 13

Zhang, X., Zhao, J., and LeCun, Y. (2015). Character-level Convolutional Networks for Text Classification. pages 649–657. 34

Apêndice

A Tabela A.1 apresenta a matriz de confusão do modelo *Principal* no *dataset* português.

A Tabela A.2 apresenta a matriz de confusão do modelo *Principal* no *dataset* português.

Real / Previsto	cantina	serviços académicos	serviços sociais	secretaria	biblioteca	saudação	agradecimento	despedida
cantina	**21**	0	0	0	0	2	1	2
serviços académicos	0	**21**	2	0	0	0	0	0
serviços sociais	0	0	**19**	0	0	0	0	0
secretaria	0	0	0	**21**	0	0	0	0
biblioteca	0	0	0	0	**21**	0	0	0
saudação	0	0	0	0	0	**19**	0	0
agradecimento	0	0	0	0	0	0	**19**	0
despedida	0	0	0	0	0	0	1	**19**

Tabela A.1 – Matriz de confusão do modelo *Principal* no *dataset* em português.

Previsto \ Real	canteen	academic services	social services	secretary	library	salutation	thanks	farewell
canteen	21	0	0	0	0	0	0	0
academic services	0	**21**	0	0	0	0	0	0
social services	0	0	**21**	0	0	0	5	3
secretary	0	0	0	**21**	0	0	0	0
library	0	0	0	0	**21**	0	0	0
salutation	0	0	0	0	0	**20**	0	0
thanks	0	0	0	0	0	1	**16**	2
farewell	0	0	0	0	0	0	0	**16**

Tabela A.2 – Matriz de confusão do modelo *Principal* no *dataset* em inglês.