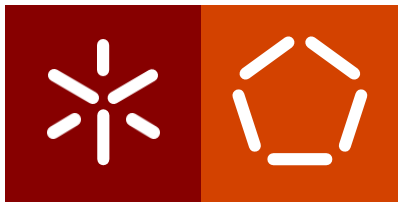


Universidade do Minho

Escola de Engenharia

André Portela de Lima Dias

Desenvolvimento de um Chatbot para apoio clínico

Outubro 2018

Universidade do Minho
Escola de Engenharia

André Portela de Lima Dias

Desenvolvimento de um Chatbot para apoio clínico

Mestrado Integrado em Engenharia Biomédica
Dissertação de Mestrado em Informática Médica

Trabalho efetuado sob orientação do
Professor Doutor Victor Alves

Outubro 2018

DECLARAÇÃO

Nome: André Portela de Lima Dias
Título dissertação: Desenvolvimento de um Chatbot para apoio clínico
Orientador: Professor Doutor Victor Alves
Supervisor na empresa: Engenheiro Martinho Gonçalves Antunes Braga
Ano de conclusão: 2018
Designação do Mestrado: Mestrado Integrado em Engenharia Biomédica
Área de Especialização: Informática Médica
Escola: de Engenharia
Departamento: de Informática

Universidade do Minho, _____/_____/_____

Assinatura: _____

AGRADECIMENTOS

Agora que chegou a etapa final do meu percurso académico, olhando para os cinco anos que passaram, são inúmeras as pessoas que, de maneira direta ou indireta, fizeram com que este fosse concluído com sucesso.

Antes de tudo, queria agradecer à minha família, sobretudo aos meus pais, Aurora e Jorge, e à minha irmã, Maria João, cujo suporte, força e paciência foram sempre inigualáveis. Muito obrigado por todos os valores e ensinamentos que me transmitiram, fazendo sempre tudo que estava ao vosso alcance para que nunca me faltasse nada.

Um agradecimento especial ao meu orientador, Professor Victor Alves, que sempre se mostrou disponível e prestável durante a realização desta dissertação. Foi também graças ao professor que realizei um estágio curricular na *Tlantic*, empresa onde fui recebido da melhor maneira possível por todos os meus supervisores, nomeadamente pelo Engenheiro Martinho Braga, pela Engenheira Isabel Pires Pimentel e pela Engenheira Isabel Faria. A todos vocês, muito obrigado! A vossa disponibilidade e conhecimento transmitido, fez-me crescer, quer a nível pessoal, quer a nível profissional.

A todos os meus amigos, que felizmente são muitos, tornando impossível a sua individualização, deixo um particular reconhecimento. Devido ao facto de sempre se mostrarem disponíveis e principalmente, compreensíveis, tornaram esta jornada mais agradável. Muito obrigado por todos os momentos que me proporcionaram e vão continuar a proporcionar !

Dedico a presente dissertação aos meus pais e à minha irmã.

RESUMO

Os avanços recentes das tecnologias de inteligência artificial e de processamento de dados mudaram radicalmente o paradigma do setor de saúde, dando origem a soluções digitais que prometem transformar os vários processos clínicos, permitindo um aumento da sua eficiência e qualidade enquanto capazes de reduzir os custos a eles associados.

Com os profissionais de saúde a enfrentarem diariamente o problema de possuírem recursos limitados, fazendo com que não sejam capazes de monitorzar e apoiar diariamente todos os seus pacientes, cada vez mais se torna importante o desenvolvimento de alternativas válidas e fidignas, capazes de ajudar os vários pacientes, no mínimo tempo possível.

Uma das soluções mais adotadas de maneira a solucionar o problema referido, reside no desenvolvimento de sistemas conversacionais, comumente chamados de Chatbots, que se assumem como capazes de efetuar o esclarecimento de questões de âmbito clínico, incorporando uma função semelhante a um assistente virtual e preenchendo, desta forma, a lacuna existente na comunicação entre os vários pacientes e os profissionais de saúde.

Esta dissertação tem como foco, a sugestão de uma arquitetura relativa a um sistema conversacional com o objetivo de efetuar aconselhamento psiquiátrico. O Chatbot proposto tem o nome de YEC, acrónimo em inglês para *"Your Everyday Companion"*, representativo de uma abordagem híbrida, pela combinação de técnicas de processamento de linguagem natural e de um modelo *Deep Learning*, para a geração da sua resposta. O sistema é desenhado para efetuar tratamento diferenciado por utilizador, permitindo desta forma a inferência do seu estado emocional, bem como do estabelecimento de um grau elevado de confiança e proximidade.

De forma a provar que o sistema apresentado na teoria, representa uma solução prática viável, procedeu-se ao desenvolvimento de uma fase primária do motor conversacional presente no sistema, expondo as diferentes abordagens realizadas de forma a que, pela análise dos seus resultados, fosse possível inferir sobre a sua melhor implementação.

A realização desta dissertação permitiu assim concluír acerca do poder inerente à combinação de técnicas de DL e de NLP para modelação conversacional, aferindo assim da sua capacidade para ajudar a resolver os diferentes problemas clínicos observados nos dias de hoje, sendo no entanto necessário mais investigação de modo a enfatizar esta afirmação.

Abstract

The recent advances of technologies for artificial intelligence and data processing have radically changed the healthcare industry, giving rise to digital healthcare solutions, promising to transform the whole healthcare process to become more efficient, less expensive and with higher quality.

Nowadays, Health professionals have to deal with the lack of resources, not being able to personally monitor and support patients in their everyday life, so it's becoming more and more important, to find and develop alternative ways to instantaneously help patients, corresponding to their needs.

One of the solutions for the problem above, resides in the form of dialogue systems, called Chatbots, that could play a leading role, by embodying the function of a virtual assistant and bridging the gap between patients and clinicians.

This thesis focus on the suggestion of a Chatbot architecture, for both psychiatric counseling and elderly monitoring, combining methodologies to emotion recognition and intent understanding. The system proposed is called YEC, acronym for "Your Everyday Companion", and represents a hybrid approach that integrates both NLP techniques and an encoder-decoder Deep Learning model, in order to generate the appropriated response. The system is designed to perform a differentiated treatment for every user in the system, thus allowing the establishment of trust and confidence between them.

To prove that the proposed approach is a viable solution to the presented problem, it is demonstrated the practical implementation of the YEC model, for an initial phase of the system, demonstrating how the use of different recurrent neural networks in our model results in dissimilar performance results for our Chatbot, allowing us to infer which one is better suitable.

This work allowed to realize the potential of combining Deep Learning and Natural language Processing for conversational modeling and their capability for solving some real life problems associated with healthcare, being nevertheless necessary more future work to give emphasis to this affirmation.

ÍNDICE

LISTA DE FIGURAS

NOTAÇÃO, ACRÓNIMOS E GLOSSÁRIO

NOTAÇÃO GERAL

A notação ao longo do documento segue a seguinte convenção:

- **Texto em itálico** – para palavras em língua estrangeira (ex., Inglês, Latim, Francês), equações e fórmulas matemáticas. Também utilizado para dar ênfase a um determinado termo ou expressão e para destacar nomes próprios.
- **Texto em negrito** – utilizado para realçar um conceito ou palavra.
- **Texto com o tipo de letra "Consolas"** – para excertos e exemplos de código.

A presente dissertação foi elaborada ao abrigo do novo acordo ortográfico. Ao longo do presente documento são utilizados termos em inglês em situações em que esta língua é universalmente aceite.

ACRÓNIMOS

ANN *Artificial Neural Networks* (Redes Neuronais Artificiais)

API *Application Program Interface*

DL *Deep Learning*

HTML *Hypertext Markup Language*

IA Inteligência Artificial

IM *Instant Messaging*

JSON *JavaScript Object Notation*

LSTM *Long Short Term Memory*

ML *Machine Learning*

NLP *Natural Language Processing*

NMT *Neural Machine Translation*

RNN *Recurrent Neural Networks* (Redes Neuronais Recorrentes)

GLOSSÁRIO

Batch size – Número de casos de treino que são utilizados em cada *epoch*.

Backend – Lógica de um serviço, fornecido por uma determinada interface.

Corpus linguístico – Conjunto de textos escritos numa determinada língua e que serve como base de análise.

Dataset – Conjunto de dados.

Deep Learning – Subárea de *Machine Learning*, responsável pela aprendizagem de vários níveis de representação e abstração, que permitam fazer com que o conjunto de dados obtenha um sentido, utilizando para tal redes neuronais artificiais.

Epochs – Número de ciclos completos efetuados na rede, para todos os casos de treino

Input – Expressão da língua inglesa que significa entrada de dados.

Inteligência Artificial – Subcampo da ciência da computação responsável pela elaboração de métodos e dispositivos que simulem a capacidade humana de raciocinar, perceber, tomar decisões e resolver problemas.

Machine Learning – Subárea de IA baseada na ideia de que os sistemas computacionais podem aprender com dados, procurando métodos de identificação de padrões e tomada de decisão, com o mínimo de intervenção humana possível.

Output – Expressão da língua inglesa que significa saída de dados.

Parse Tree – Árvore de análise sintática.

XML Tag – Marcação do significado de um determinado bloco de dados, sendo determinada por uma palavra dentro dos sinais "< >".

Unicode – Padrão universal de codificação de caracteres, utilizado para suportar caracteres em scripts não-ASCII, constituido por 128 caracteres.

Capítulo 1
Introdução

A presente dissertação descreve uma proposta de desenvolvimento de um Chatbot para apoio clínico. O projeto surge no âmbito da realização da dissertação do Mestrado Integrado em Engenharia Biomédica da Universidade do Minho. Neste capítulo é apresentada uma contextualização ao tema, a motivação para a elaboração do mesmo e a sua aplicabilidade. O capítulo é finalizado com a apresentação da estrutura do documento.

1.1 INTRODUÇÃO

A ideia por detrás do fornecimento da capacidade conversacional a uma máquina, começou há cerca de 60 anos, desde que o famoso matemático *Alan Turing* questionou na sua famosa obra, *"Computing Machinery and Intelligence"*, se "Podem as máquinas pensar?", conceptualizando o problema no que foi chamado de jogo de imitação, atualmente amplamente conhecido como teste de *Turing* [1].

Hoje em dia, o discurso ou a troca de mensagens entre um ser humano e um computador está a ganhar cada vez mais popularidade, com a comunicação virtual a ocupar uma grande porção do nosso quoatidiano, quer a nível pessoal quer a nivel profissional [2]. A título de exemplo, quase de maneira abstrata, colocamos a cargo e confiamos em métodos de comunicação virtual para efetuar a transferência de grandes quantidades de informação entre sistemas - uma tarefa comum, nos dias de hoje, a todos os serviços e grandes empresas. Outro exemplo desta transformação, é a aprovação de documentação, e posterior notificação desta via e-mail, que permitiu otimizar e elevar a eficiência de negócio para outro patamar, inexistente antes desta revolução tecnológica [3], [4].

A automatização destes processos é demonstrativa do poder adjacente a esta nova abordagem comunicacional, que levou a que a realização de certas tarefas que tradicionalmente, seriam efetuadas de forma manual, fossem entregues a métodos tecnológicos [3], [5].

Como demonstra a figura 1.1, podemos distinguir dois grandes tipos de comunicação virtual, quanto ao seu conceito de troca de mensagens: Troca de mensagens de maneira **instantânea** (em inglês, *Instant Messaging*) e troca de mensagens de maneira **não instantânea** (em inglês, *Non-Instant Messaging*) [4].

A troca de mensagens instantânea diz respeito a uma forma de comunicação em tempo real, sob a forma de texto ou voz, entre dois ou mais sistemas em rede, onde se incluem aplicações amplamente conhecidas como o *Whatsapp* ou o *Facebook Messenger,* estando também incluídos neste grupo, os sistemas que representam o foco desta dissertação: os diversos Chatbots desenvolvidos e implementados nas diferentes áreas [6].

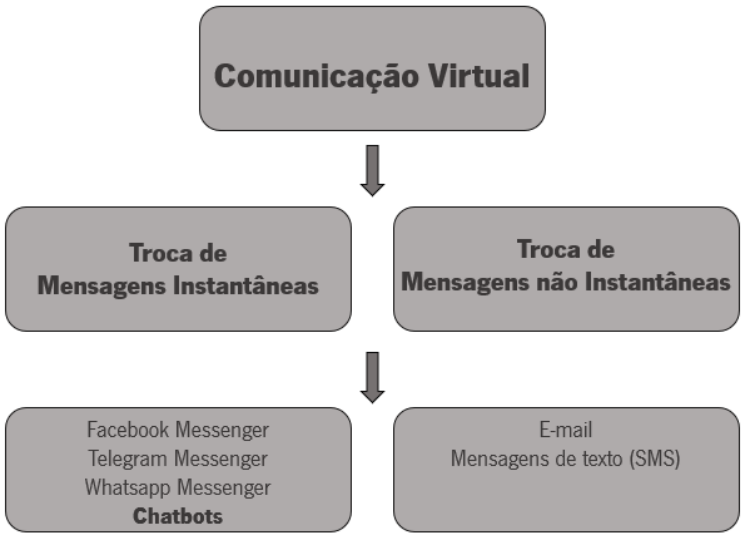


Figura 1.1 - Tipos de comunicação virtual e exemplos.

A maior parte dos seus utilizadores, utillizavam estas aplicações de IM no seu computador pessoal, sendo que, nos últimos anos, estes sistemas tornaram-se cada vez mais frequentes noutro tipo de dispositivos, maioritariamente móveis, levando assim ao seu crescimento aplicacional [4].

Esta transposição da comunicação virtual para a componente móvel, trouxe consigo inúmeras vantagens para os seus utilizadores, sendo as mais importantes, apresentadas de seguida, como sugerido por [7]:

→ **Conveniência** – Possibilidade de interacção entre diferentes pessoas, independentemente do tempo em que é realizada e do local em que se encontram.

→ **Poupança de tempo** – Dado que a comunicação é feita com recurso a um dispositivo móvel, não existe necessidade de encontro físico, podendo diferentes de aplicações de IM suportar o envio da informação que se pretende transmitir.

→ **Redução de custos** – Redução de custos de transporte, permitindo à empresa ou serviço minimizar os custos operacionais.

No entanto, como nem tudo se traduz em vantagens, um dos problemas que a introdução desta nova forma de comunicar trouxe foi o facto de, com o passar do tempo, esta garantia de comunicação em tempo real, ter levado a que os utilizadores, como consumidores, possuissem a expectativa de que as empresas ou serviços devessem estar sempre disponíveis, acessíveis e com assistência imediata. Estas três "exigências" revelavam-se difíceis de ultrapassar e portanto, não eram fornecidas por todos os serviços devido à falta de recursos exigidos pela sua implementação.

Foi aqui que o desenvolvimento de programas computacionais, disponíveis 24 horas por dia, 7 dias por semana, capazes de simular a conversa humana, entrou em acção [8]. Esses programas são chamados de Chatbots, e a sua introdução permitiu a estas, responderem às exigências dos consumidores por respostas imediatas, sem drenar os recursos e as receitas resultantes dos seus serviços. Podemos de certa forma, afirmar sob o ponto de vista negocial, que os Chatbots são, na verdade, uma resposta ao nosso problema de impaciência [8].

O crescimento nos últimos anos da área de inteligência artificial, levou a que o desenvolvimento deste tipo de sistemas fosse para outro patamar, mudando totalmente o paradigma de como um serviço é disponibilizado.

O sistema conversacional *Siri,* da tão famosa marca *Apple* ou o sistema *Cortana* fornecido pela *Microsoft* são dois exemplos de Chatbots construídos com recurso a inteligência artificial, que se assumem como assistentes pessoais e que são representativos da enorme potencialidade deste tipo de sistemas no nosso quoatidiano - a título de exemplo, centenas de milhares de seres humanos saúdam um destes dois companheiros virtuais todos os dias, sendo que meio milhão já declarou o seu amor por uma destas [6].

É esta cumplicidade e facilidade de uso que faz com que a inteligência artificial seja uma arma poderosa para o desenvolvimento de novos sistemas nas diversas áreas, sendo o ambiente clínico, apenas mais uma área com inúmeras potencialidades a serem exploradas por este ramo tecnológico [9]. Tendo como base a figura 1.2, que ilustra os resultados de um estudo efetuado no Reino Unido , para determinar as aplicações na área clínica mais requisitadas pelos consumidores, verificamos que realização de marcações, a gestão de medicamentos, o esclarecimento de questões clínicas genéricas ou o suporte a doentes são tudo tarefas e serviços que podem ser implementados sob a forma de um chatbot - as hipóteses são ínfimas e é a capacidade do sistema em se moldar ao propósito da sua projecção, que o torna atualmente, uma ferramenta poderosa [10].

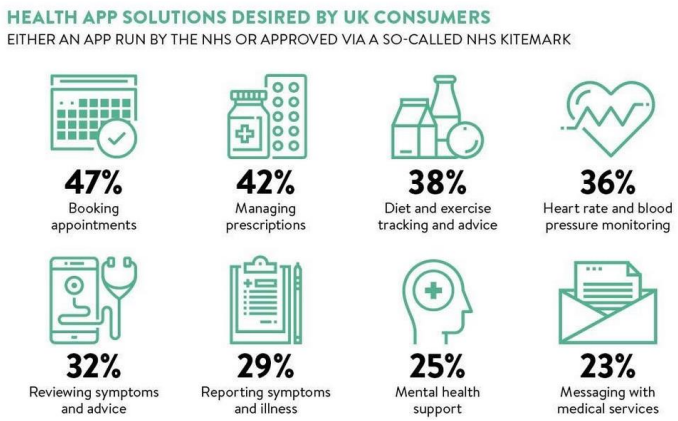


Figura 1.2 - Resultados de um estudo realizado no Reino Unido, referente às aplicações médicas mais requisitadas pelos consumidores (retirado de [10]).

1.2 MOTIVAÇÃO

A nossa saúde e bem-estar são os elementos mais importantes no nosso dia-a-dia. No entanto, muitas das vezes, para uma grande percentagem da população, o acesso a estes dois bens pode-se revelar uma tarefa árdua devido a certas razões que podiam ser ultrapassadas com o desenvolvimento de um sistema conversacional [11].

Por exemplo, existem inúmeros relatos de utentes que direcionam culpas ao sistema público de saúde, por este não possuir a capacidade de satisfazer toda a sua procura no tempo útil ou desejado pelos utentes. Isto pode ser verificado pelo simples facto de diariamente existirem diversos casos de utentes, que se deslocam a centros hospitalares por questões simples, que provavelmente não requeriam a atenção total ou presença física de um profissional de saúde [12], [13]. No entanto, apesar de serem questões de caractér não urgente, se estas não forem esclarecidas, provocam na maior parte dos casos, do lado do utente, um aumento do receio e nervosismo, levando este a um estado de procupação que, em muitos dos casos, é lhe prejudicial e facilmente evitável. Adicionalmente a este problema, uma certa percentagem afirma que não pode recorrer ao sistema de saúde privado, devido ao facto deste acarretar custos superiores daqueles que estes podem suportar.

Em última instância, existem ainda casos, em que o problema não reside no tempo de espera do serviço público nem no custo mais elevado do serviço privado, mas sim na falta de tempo para se deslocar a centros hospitalares. Uma grande parte destes, como solução, opta por efetuar pesquisas na internet de maneira a encontrar a melhor resposta, o que se tem revelado um desafio para muitos, devido ao facto de não conseguem filtrar e aceder a informação de qualidade, sendo muitas das vezes induzidos em erro [14]. Um estudo efetuado pela entidade *PewResearch*, revela que a procura de informação relativa à saúde, é, numa fase inicial, efetuada por aproximadamente 72% dos utentes, devido ao facto destes desejarem informações de forma rápida, mas que no entanto se revelam pouco confiáveis [15].

Atualmente, todas estas adversidades referidas, conjugadas com a falta de fiabilidade na auto-pesquisa efetuada pelos pacientes, deixa-lhes como única opção existente, dirigirem-se pessoalmente ao centro hospitalar, onde se incluem os casos, em que num curto espaço de tempo , obtêm a resposta à sua simples pergunta, mesmo que para isso tenham de ter ficado à espera um tempo muito superior, do que aquele que seria expéctavel.

Será conveniente para um paciente, deslocar-se para um centro hospitalar, por cada questão de carácter não urgente que lhe surja [16]?

A resposta mais natural a esta questão será que não, pois muitas das vezes acarreta custos desnecessários para o paciente para além de fazer aumentar a espera nos centros hospitalares. O objetivo não passa por definir o que é ou não urgente, ou aquilo que necessita da presença física de um profissional de saúde, pois todos os utentes, possuem naturalmente, o mesmo direito de acesso à informação e tratamento.

O que se procura nesta dissertação é tentar fornecer uma maneira mais conveniente para o paciente, de obter a ajuda e as respostas que necessita, a qualquer momento, através do desenvolvimento de um Chatbot para apoio clínico. A melhoria substancial a que se vem assistindo nas áreas de inteligência artificial, mais precisamente na área de *Deep Learning*, conjugado com os avanços presentes nas áreas de compreensão e processamento linguístico, permitiram efetuar uma nova abordagem ao desenvolvimento destes sistemas conversacionais, fornecendo-lhe, na verdadeira essência da palavra, inteligência, algo que era transmitido como ilusão até à data [17].

Perante todos estes avanços e desenvolvimentos, a questão principal a ser respondida é a seguinte:

Com base nas tecnologias de DL e NLP, até que ponto pode a vida dos utilizadores-alvo identificados nesta dissertação, melhorar, usando um Chatbot devidamente treinado para o efeito?

1.3 OBJETIVOS

A presente dissertação possui como objetivo explorar o atual estado de arte em termos de *design*, técnicas adotadas e finalidades, aquando da construção de um Chatbot, expondo toda a informação necessária para a compreensão do seu funcionamento.

Para além da componente teórica, é tido como finalidade da dissertação, a avaliação da possibilidade de desenvolvimento de um destes sistemas para apoío clínico, efetuando a construção da arquitetura do sistema proposto, bem como se possível, a implementação prática deste.

Tendo em conta os objetivos expostos acima, estes podem ser divididos num conjunto de questões de investigação, propostas a serem respondidas ao longo do documento:

- → Que lacunas são encontradas na construção de um Chatbot baseado em *Deep Learning* ?
- → Qual o propósito de desenvolver um chatbot para a área da saúde ?
- → Será possível na prática efetuar o desenvolvimento de um Chatbot baseado em métodos de *Deep Learning*, com o objetivo deste se assumir como um companheiro virtual ?
- → Será o sistema proposto capaz de ser aceite e adaptado pelos seus utilizadores-alvo identificados?

1.4 ESTRUTURA DO DOCUMENTO

Para além da introdução já apresentada anteriormente, esta dissertação é composta por mais cinco capítulos:

No **capítulo dois** descrevem-se os principais conceitos associados a um Chatbot, relevantes para a compreensão do presente trabalho. Neste está contida toda a revisão de literatura, onde se inclui a sua definição e os seus diferentes tipos de implementação, terminando o capítulo com o seu estado de arte, onde é descrito o funcionamento e a lógica implementada em todos os Chatbots referenciados, permitindo inferir acerca da evolução assistida no desenvolvimento deste tipo de sistemas.

O **capítulo três** fornece uma apresentação teórica sobre as tecnologias utilizadas para o desenvolvimento do tipo de Chatbot projectado nesta dissertação, sendo também formulado um sub-capítulo sobre o processamento de linguagem natural, onde se apresenta no que consiste esta área bem como as suas técnicas mais importantes.

De seguida, no **capítulo quatro**, é exposta a solução proposta, apresentado para tal, a arquitetura sugerida para o desenvolvimento de um Chatbot para apoio clínico. Aqui são descritos todos os componentes que constituem o sistema, e a maneira como estes se interligam, para no final , constituírem um Chatbot diferenciado dos atuais, e capaz de realizar o objetivo proposto.

No **capítulo cinco** é efetuada a implementação prática da arquitetura proposta anteriormente. Neste são apresentados os resultados obtidos a partir do desenvolvimento do Chatbot, discutindo e avaliando o seu comportamento final. A solução é avaliada a partir da aceitação do utilizador, perante a usabilidade e adaptação por parte deste, conseguindo desta maneira verificar se a solução implementada, é ou não viável.

A dissertação termina no **capítulo seis**, onde são apresentadas as conclusões e onde é efetuada uma análise a um trabalho futuro.

A figura 1.3 ilustra a ordem de leitura dos capítulos, bem como a sua nomenclatura.

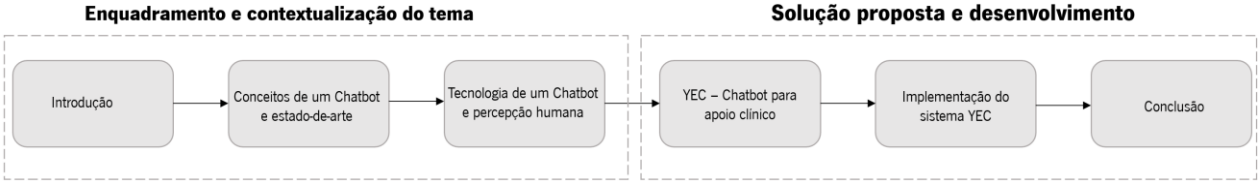


Figura 1.3 - Diagrama da estruturação dos capítulos presentes neste dissertação.

CAPÍTULO 2
CONCEITOS DE UM CHATBOT E ESTADO-DE-ARTE

No presente capítulo apresenta-se uma revisão de todos os conceitos associados ao termo Chatbot, iniciando-se com a sua definição, funcionalidade e propósito. De seguida são apresentadas as características diferenciadoras deste tipo de sistemas, expondo para tal, os seus diferentes tipos de projecção. O capítulo termina com a apresentação do seu estado de arte.

2.1 FUNDAMENTOS DE UM CHATBOT

Hoje em dia existem diversos termos para nos referirmos a um Chatbot: agentes ou sistemas conversacionais, *chatterbots*, *chat robots* ou simplesmente *bots*, sendo que, todos estes termos são tratados como equivalentes devido ao facto do seu núcleo de funcionamento e propósito ser o mesmo: simular a habilidade conversacional humana [18].

Devido à complexidade destes sistemas, é dificil encontrar uma definição universal para o termo *Chatbot*, podendo estes serem projetados de inúmeras maneiras e com diferentes propósitos, dependendo sempre da visão do seu desenvolvedor. Para o efeito desta dissertação, definimos **chatbot** como sendo [18]:

"Um agente conversacional baseado em regras e/ou métodos de inteligência artificial, com o objetivo de simular uma conversa humana real, através do uso de linguagem natural para comunicar com os seus utilizadores".

Apesar destes sistemas conversacionais poderem ser arquitetados de uma vasta variedade de formas, as suas funções expéctaveis e características base são idênticas em todos eles, dado que todos convergem para o mesmo objetivo final.

Sansonnet et. al. em [19], fornece uma estrutura básica comportamental, indicativa daquilo que é expectável num Chatbot moderno:

→ **Agente de aprendizagem dialógica** – Deve entender o utilizador, isto é, deve fornecer a função de compreensão através da sua capacidade em extrair o contexto e o significado acoplado ás diferentes entradas fornecidas pelo utilizador.

→ **Agente racional** – Deve ter acesso ou deve ser alimentado por uma base de conhecimento externa e representativa do senso comum e conhecimento geral, de maneira a fornecer um certo tipo de competência aquando da formulação da resposta a devolver ao utilizador. Esta base de conhecimento pode ser projectada de diversas maneira como iremos verificar ao longo desta dissertação, podendo esta ser estática ou dinâmica, representativa de conhecimento específico ou genérico.

→ **Agente personificado/corporificado** – Deve fornecer a função de presença, sendo uma das maiores características que um Chatbot deve possuir. A concepção de uma personalidade por parte de um sistema conversacional é um grande desafio mas que se revela importante no estabelecimento da confiança entre o utilizador e o sistema, sendo o seu objetivo final a percepção deste por parte do utilizador como uma pessoa real. A atribuição de um nome ao sistema desenvolvido é um exemplo da sua personificação.

Adicionalmente a esta estrutura caracterizadora do que um Chatbot deve incorporar aquando da sua projecção, *McTear et al.* [20], fornece uma estrutura operacional, demonstrativa do seu funcionamento genérico:

→ **Reconhecer** o texto enviado pelo utilizador.
→ **Interpretar** as palavras nesse texto, extraíndo o seu significado e contexto.
→ **Formular** a resposta a devolver e ter a capacidade de, se a mensagem recebida do utilizador não for clara, interagir com este para obter esclarecimento.
→ **Construír** a resposta, podendo esta ser somente na forma de palavras, ou, conjugar outro tipo de dados como imagens ou vídeos.
→ **Devolver** a resposta ao utilizador.

É este o princípio adjacente a cada Chatbot: Interagir com os seus utilizadores, comportar-se de forma adequada e ter a capacidade de compreender a conversa no qual está inserido, devolvendo uma resposta apropriada para cada interacção a ele fornecido.

Mas qual a razão para tanto foco e para este súbito interesse neste tipo de sistemas, nos dias de hoje? Apesar dos primeiros chatbots terem surgido já nos anos 60, só passado meia centena de anos é que podemos afirmar que o mundo se encontra preparado para a sua implementação na vida real. Para além do constante crescimento assistido na área de inteligência artificial e de processamento de linguagem natural, recentemente houve um aumento do interesse nestes sistemas devido maioritariamente a duas razões [21]:

A primeira razão deve-se ao crescimento dos serviços de troca de mensagem, que inclusive já se tornaram mais populares e mais utilizados quando comparado com as redes sociais. A título de curiosidade, as quatro principais aplicações de troca de mensagens (*Messenger*, *WhatsApp*, *WeChat* e *Viber*) possuem mais utilizadores ativos por mês que qualquer umas das quatro maiores redes socais (*Facebook*, *Twitter*, *Instagram* e *Google+*) [21], [22].

Este facto leva-nos a concluir que, se os utilizadores se encontram a virar de maneira tão ativa para as aplicações de troca de mensagens textuais, devia tornar-se natural para as diferentes marcas e serviços, procurarem desenvolver e incorporar os seus próprios serviços neste tipo de sistemas conversacionais, de maneira a estarem presentes onde os utilizadores desejam estar.

A outra razão, reside no facto dos utilizadores, no papel de consumidores, não acreditarem em horários de funcionamento para os serviços que desejam usufruir. A figura 2.1 ilustra os resultados de um estudo efetuado no ano de 2016, intitulado *The 2016 Mobile Messaging Report,* onde foi demonstrado que 51% dos utilizadores clamam que um serviço devia estar disponível 24 horas por dia, 7 dias por semana, sendo que por esta altura, esse número já deve ter aumentado substancialmente [23].

Esta é uma das motivações principais no desenvolvimento de um Chatbot, a eliminação de hórario de funcionamento, permitindo a interação dos utilizadores com o serviço a ele acoplado, independentemente da hora e local onde esta interacção é realizada.

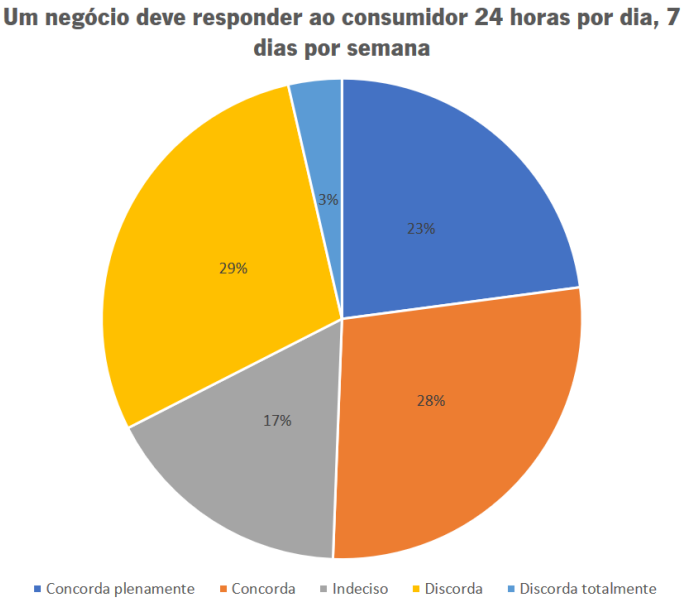


Figura 2.1 - Resultados do estudo "The 2016 Mobile Messaging Report", aquando do questionamento acerca de qual devia ser o horário de funcionamento de um determinado serviço .

2.2 OS DIFERENTES TIPOS DE CHATBOT

Como já referido, a dificuldade em encontrar uma definição única e universalmente aceite para um Chatbot, prende-se pelo facto de este poder ser desenvolvido de diversas maneiras, tendo sempre como base, a visão de quem projeta o sistema a desenvolver.

Perante este facto, esta sub-secção tem como propósito apresentar as principais diferenças e decisões a tomar, aquando do desenvolvimento de um Chatbot, identificando os princípais aspetos a ter em conta durante a sua projecção. A sub-secção termina com um olhar geral acerca das diferentes combinações arquitectónicas de um Chatbot, acoplando o grau de dificuldade a estas adjacente.

2.2.1 Domínio aberto e domínio fechado

A primeira característica de um Chatbot capaz de ser distinguível, relaciona-se com o seu domínio conversacional: **aberto** ou **fechado** [24].

Um sistema de domínio aberto é usualmente mais dificil de projetar e implementar. Neste tipo de sistemas, o utilizador pode interagir com o Chatbot acerca de qualquer tópico, não existindo um objetivo ou propósito definido aquando da sua construção [25].

O número infinito de tópicos e o facto deste necessitar de um conhecimento geral, capaz de abranger um vocabulário extenso que permita a criação de respostas apropriadas perante qualquer tipo de interacção oriunda dos seus utilizadores, tornam este tipo de sistemas, um problema difícil em termos de implementação prática [26]. A realização de conversas nas diferentes redes sociais como o *Twitter,* o *Reddit* ou o *Facebook* são exemplos de interacções em domínio aberto – estas podem tomar qualquer direcção [26].

Contrariamente ao sistema de domínio aberto, os sistemas projetados para um domínio de conversa fechado são mais fáceis de implementar devido à limitação do universo de entradas e saídas – o sistema converge para um objetivo específico [25]. Sistemas de apoio ao cliente ou assistentes virtuais associados a uma determinada empresa ou serviço são exemplos de sistemas que possuem o seu domínio conversacional fechado.

Estes sistemas não possuem a capacidade de responder a temas que não estejam no seu universo de discurso, possuindo uma certa tendência para a devolução de respostas genéricas para tudo aquilo que se encontra deslocado do seu espectro conversacional.

2.2.2 Modelos baseados na extracção de informação e modelos generativos

A distinção mais importante a realizar e a ter em conta na projecção de um Chatbot reside no tipo de modelo implementado para a devolução da resposta para o utilizador: se o sistema é baseado na **extracção de informação** ou se é um **modelo generativo** [27].

Os modelos baseados na extracção de informação possuem como base funcional, o processamento da entrada fornecida pelo utilizador, de forma a retornar uma resposta existente num determinado conjunto

previamente definido para o sistema, onde se encontram as diversas relações entre as entradas recebidas e as respostas a serem devolvidas [24], [27].

A sua heurística pode ser tão simples como a correspondência baseada em regras previamente formuladas, ou tão complexa como a utilização de algoritmos de classificação baseados em métodos de *Machine Learning* [27].

As consequências da utilização deste modelo residem no facto de se tornar necessário antecipar todos os casos passíveis de serem transmitidos para o Chatbot, sendo imperativo para tal, a construção de um conjunto de dados que permita responder a todos estes casos – o nosso domínio conversacional é fechado, restrito perante aquilo que declaramos no nosso conjunto de respostas.

Um aspeto positivo na utilização deste modelo é o facto de conseguirmos controlar o que se encontra na sua base de conhecimento, evitando assim a ocorrência de erros gramaticais e de más construções frásicas, garantindo desta forma uma elevada qualidade nas respostas devolvidas pelo sistema.

Contrariamente aos modelos baseados em extracção, os modelos generativos são como o nome indica, capazes de gerar novas frases para a sua resposta, não possuíndo nenhum domínio de resposta pré-estabelecido [24]. A ideia por detrás destes modelos passa pelo fornecimento de grandes quantidades de dados relativos a conversas humanas, para que o sistema seja capaz de aprender como modelar a linguagem, tornando-se suficientemente autónomo para gerar e devolver uma resposta, sem nenhum tipo de regras a si associado [25], [27].

Este tipo de modelos é tipicamente implementado com recurso a técnicas de inteligência artificial e de processamento de linguagem natural, residindo o seu motor conversacional nas ferramentas e modelos de *Deep Learning* baseados em redes neuronais.

Em contraste com os sistemas baseados na extracção de informação, estes modelos generativos são capazes de assumir um comportamento semelhante ao comportamento humano real, adaptando-se a questões complexas oriundas do utilizador e devolvendo uma maior diversidade de respostas devido à incorporação de um maior leque de vocabulário [28].

O problema existente para já nestes tipo de modelos reside no facto das respostas devolvidas por estes tenderem por vezes a ser genéricas, irrelevantes, inconsistentes ou mal formuladas gramáticamente, sendo necessário um elevado número de dados conversacionais para permitir ao sistema adaptar-se e adquirir a capacidade de formular a sua própria resposta com base no vocabulário a ele fornecido [26].

A figura 2.2 ilustra a estrutura presente em cada um destes tipos de sistema, apresentando o funcionamento de cada um dos modelos apresentados anteriormente.

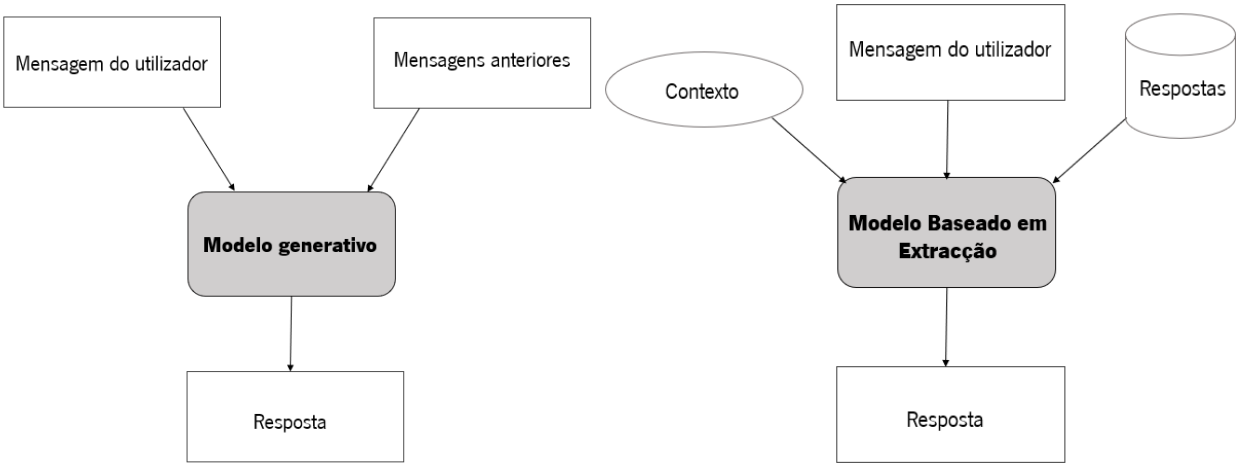


Figura 2.2 - Estrutura dos modelos generativos e baseados em extracção, passíveis de serem implmentados num Chatbot.

2.2.3 CHATBOTS CONVERSACIONAIS E CHATBOTS ORIENTADOS A UMA TAREFA

Outra distinção frequente nos Chatbots, reside no seu propósito: se este é um sistema meramente **conversacional** ou se é construído de maneira a realizar uma determinada **tarefa**, i.e. com o intuíto de atingir um certo objetivo [24].

Os Chatbots conversacionais têm como objetivo entreter o utilizador, simulando uma conversa geral, sem que esta possua um propósito ou meta final. Neste tipo de Chatbots, a longevidade da conversa é um indicador importante do bom funcionamento do sistema construido, devido ao facto de ser uma métrica demonstrativa da capacidade deste em gerar respostas adequadas e com qualidade, para as diferentes entradas fornecidas pelo utilizador ao longo do tempo [29].

Por outro lado, um Chatbot orientado a uma dada tarefa, tendencialmente projectado como um sistema de domínio fechado, possui como propósito a interpretação e a devolução, o mais rapidamente possível, da resposta à interacção fornecida pelo utilizador.

Estes são na sua maioria, Chatbots implementados por empresas e/ou serviços, capazes de efetuar conexões a fontes estáticas de informação a si associadas, como por exemplo à sua página de FAQ (acrónimo da expressão inglesa, *frequently asked questions*) ou à sua base de dados, tendo em vista a devolução de informação específica requisitada pelo utilizador [25], [30].

Nestes casos, o objetivo não é entreter ou conseguir manter uma conversa coerente e estruturada, mas sim ter a capacidade de interpretar e devolver a resposta correta ao utilizador num espaço de tempo aceitável e que satisfaça as suas expectativas [30], [31]. Não é suposto serem companheiros do utilizador, mas sim receber a informação necessária e transmiti-la para este.

Podem possuir personalidade e ser amigáveis, conseguindo até lembrar-se de informação prévia acerca do utilizador aquando de uma nova conversa, mas não é expectável que o façam [30], [31].

2.2.4 RESPOSTAS CURTAS E RESPOSTAS LONGAS

Finalmente, a última distinção usualmente realizada num Chatbot baseia-se no **comprimento frásico** das suas interacções. Quanto mais longa é uma conversa, mais dificil é a sua automatização. Aquando da elaboração da arquitetura de um Chatbot é comum depararmo-nos com o problema do quão extensas serão as frases envolvidas no diálogo a simular, pois estas podem influenciar por si só, toda a adesão e qualidade associada a um Chatbot [32].

As conversas de texto curto, com poucos termos em cada interacção, são mais fáceis de implementar, sendo o objetivo destes Chatbots, criar uma resposta única e objetiva para cada entrada recebida. Neste tipo, o sistema devolve uma resposta sem andar em detalhe nem efetuar mais perguntas acerca da questão recebida. As respostas são baseadas em objectividade, sendo este o tipo de implementação mais escolhido numa fase inicial de desenvolvimento devido ao facto de permitir ao sistema moldar e interligar os termos mais facilmente [32], [33].

Depois existem sistemas de conversas extensas, mais difíceis de implementar devido ao facto de necessitarem algum mecanismo de memória a si associados. Esta capacidade de memorização resulta do estabelecimento por parte do sistema de um certo tipo de questionário acerca da entrada fornecida pelo utilizador, de maneira a que este consiga obter informação suficiente para retornar a resposta mais adequada, sendo necessário para tal que o sistema se consiga manter a par do que está a ser referido pelo utilizador. Os sistemas de apoio ao cliente são normalmente um exemplo de sistemas que utilizam este tipo de lógica [32], [33].

O tamanho da mensagem a ser devolvida pelo Chatbot deve ser um parâmetro a ter em especial consideração durante a sua projecção pois este na maior parte dos casostem a capacidade de ditar o sucesso deste através do grau de envolvimento do utilizador. É necessário verificar o contexto em que o Chatbot se encontra inserido e moldar as suas respostas de acordo com este:

Por exemplo, para o caso de um sistema orientado a uma determinada tarefa, será expectável a obtenção de respostas curtas e objetivas, de forma a realizar o desejo do utilizador no menor número de interacções possível. Por outro lado, no caso de um sistema conversacional, este parâmetro pode ser ajustado de forma a mostrar algum comprometimento e interesse por parte do sistema em relação ao utilizador, devendo para tal, fornecer diversas interacções sobre o tópico conversacional em que se encontra, de forma a extrair cada vez mais informação do utilizador.

2.2.5 PERSPECTIVA GERAL DA ESTRUTURA DE UM CHATBOT

Como descrito nas sub-secções acima, existem diferentes abordagens que um desenvolvedor pode adotar aquando da construção e projecção de um Chatbot, devendo a sua escolha se basear no **propósito** do sistema a elaborar.

As arquiteturas correspondentes aos sistemas conversacionais podem ser diferenciadas principalmente pela sua esfera operacional (domínio conversacional) e pelo método de geração da resposta a devolver ao utilizador. Diferentes arquiteturas possuem diferentes graus de dificuldade associados à sua implementação, sendo ilustradas na figura 2.3, as combinações passíveis de serem implementadas num Chatbot com base nos dois atributos referidos anteriormente.

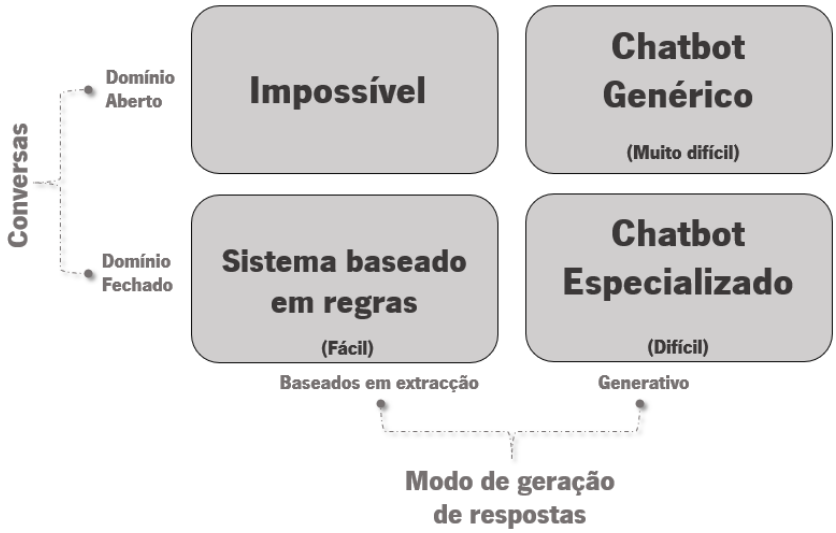


Figura 2.3 - Graus de dificuldade associados ao desenvolvimento dos diferentes tipos de sistemas conversacionais.

Naturalmente que um Chatbot que possuí um modelo baseado em extracção como lógica de devolução de resposta, nunca poderá implementar um domínio conversacional aberto, sendo este totalmente restrito aquilo que é definido pelo utilizador. Este tipo de Chatbots são usualmente chamados de sistemas baseados em regras devido ao facto de se basearem na correspondência entre aquilo que foi estabelecido no seu espectro conversacional e as diferentes entradas dos utilizadores. Este tipo de Chatbots é maioritariamente implementado em empresas ou serviços, estando associados a dois domínios distintos de informação: Um domínio de resposta e um domínio de informação sobre o serviço/empresa para o qual foi projetado.

Nestes casos, desenvolver um Chatbot de domínio aberto, revela-se algo desnecessário devido ao facto da sua intenção se resumir a que este seja apenas capaz de responder e esclarecer os temas envolvidos e relacionados com esse serviço ou empresa, devendo portanto ser projetado como um sistema de domínio fechado [34].

Quanto aos Chatbots incorporadores de modelos generativos para a devolução de resposta, estes representam uma tarefa bastante mais árdua de ser concretizada quando contrastados com os sistemas baseados em extracção. Um modelo generativo ser projetado com um domínio fechado é algo contraditório devido ao facto das respostas devolvidas pelo sistema não poderem ser limitadas, sendo este um caminho que não se deve tomar nestes casos. Em último caso temos os sistemas que representam a verdadeira essência de inteligência num Chatbot: os sistemas genéricos, incorporadores de modelos generativos para a formulação de resposta e sem qualquer restrição de área conversacional.

São o tipo mais difícil de implementar e representam o caso de estudo desta dissertação: Apesar de associarmos ao sistema nela proposto, um determinado propósito, o facto de não restringirmos o seu domínio de conversação faz com que este se enquadre num tipo de Chatbot genérico, capaz de estabelecer qualquer tipo de relacionamento independentemente do contexto em que se encontra.

Dado que o objetivo é que este forneça apoio psicológico para os seus utilizadores, quanto mais "humano" o sistema parecer, mais confiável este será aos olhos destes. Por estas razões a sua projecção e implementação é classificada como uma tarefa muito dificíl, advindo esta também da complexidade associada às técnicas de *Deep Learning* e de NLP utilizadas para o seu desenvolvimento.

2.3 ESTADO DE ARTE DOS CHATBOTS

Chatbots com variados graus de inteligência têm vindo a ser desenvolvidos desde os anos 60: Os primeiros eram assentados em correspondências simples, baseando-se no reconhecimento de padrões e de certas palavras chave para a geração da resposta a ela associada. Com o passar dos anos, assistiu-se a um crescimento exponencial na área de sistemas conversacionais inteligentes, levando a que fossem desenvolvidos diversos tipos de Chatbots, cada um adoptando uma técnica própria e inovadora, desde o uso de técnicas de processamento de linguagem natural até à utilização de metodologias de *Machine Learning* e de *Deep Learning*.

A presente sub-secção tem então como objetivo, expor a história por detrás da construção dos sistemas conversacionais, apresentado e descrevendo os Chatbots mais revolucionários e mais importantes, desde a sua criação até aos dias de hoje.

Nesta são descritas as qualidades, características e falhas de cada um destes sistemas, permitindo desta forma, o estabelecimento de uma certa comparação entre os desenvolvimentos efetuados nos diferentes espaços temporais, demonstrando assim, a evolução a que estes sistemas conversacionais assistiram até aos dias de hoje.

2.3.1 O JOGO DA IMITAÇÃO – TESTE DE *TURING*

Em 1950, Alan Turing publicou um artigo no qual colocou a famosa questão: "Podem as máquinas pensar?" [1]. Devido à imprecisão ajdacente aos termos "*máquina"* e *"pensar"*, *Turing* sugere no seu artigo, um teste empírico no qual o uso da linguagem por parte de um computador seria a base para determinar se de facto as máquinas podem pensar.

O teste, ou jogo, hoje em dia conhecido como Teste de *Turing,* envolve três participantes – dois humanos e um computador – no qual um dos seres humanos possui o papel de interrogador, com o objetivo de determinar qual dos outros dois participantes é o computador. Neste, o interrogador deve fazer uma série de questões, sem limitação de tópico, sendo o papel do computador neste caso, enganar o interrogador, levando-o a pensar que este é o ser humano [1], [35]. A premissa presente no teste assenta no facto de que se o humano e a máquina forem indistinguíveis do ponto de vista interaccional, então é legítimo afirmar que as máquinas podem pensar. A figura 2.4 ilustra o funcionamento deste teste, como proposto por *Turing*.

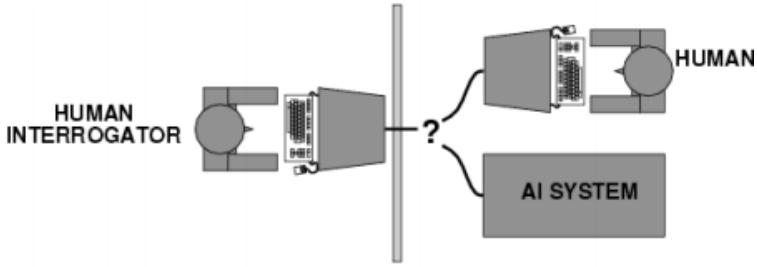


Figura 2.4 - Funcionamento operacional do jogo da imitação, proposto por Alan Turing (retirado de [36])

Devido ao facto do uso efetivo da linguagem estar interligado com as nossas habilidades cognitivas, *Turing* acreditava que a utilização desta por parte de uma máquina, representa um teste suficente para enquadrar a noção de inteligência [35]. Estas deveriam para tal, possuir as seguintes capacidades [36]:

- → Processamento de linguagem natural de forma a permitir à maquina comunicar com sucesso numa determinada língua.
- → Representação de conhecimento de maneira a armazenar a informação a ela fornecida.
- → Raciocínio automatizado que permita a devolução de respostas coerentes e a inferência de novas conclusões.

Este teste e as suas variantes, representam um dos métodos mais comuns para a avaliação da qualidade e do funcionamento de um Chatbot, e a verdade é que, passados quase 70 anos, ainda nos encontramos a tentar passar este mesmo teste, o que é demonstrativo da sua dificuldade de execução.

2.3.2 ELIZA

Em 1966, *Joseph Weizenbaum*, professor no Instituto de Tecnologia de Massachusetts (MIT), divulgou o primeiro Chatbot que chegou perto de imitar um ser humano, ao qual deu o nome de ELIZA [37].

Este sistema era capaz de fornecer aos seus utilizadores, pela primeira vez, a ilusão de que estes se encontravam de facto a comunicar com um indíviduo real. O seu criador porém, não pensou em ELIZA como um programa inteligente, mas sim num programa que apenas fornece a **ilusão de inteligência**, isto porque o sistema não era auto-capaz de formular a sua própria resposta, mas sim retornar aquilo que lhe foi pré-estabelecido, enquadrando-se num sistema baseado na extração de informação [38]. No entanto, para o utilizador, a maneira como a resposta é formulada, é algo abstrato e não relevante, daí o uso da expressão "ilusão de inteligência" para caracterizar ELIZA.

Como se verificou no desenvolvimento posterior de sistemas conversacionais, esta definição de inteligência tornou-se num fator essencial e característico da qualidade de um Chatbot, ainda que sujeito a bastante subjectividade.

A figura 2.5 ilustra um exemplo de uma interacção realizada no sistema ELIZA.

Figura 2.5 - Exemplo de interacção conversacional presente no Chatbot ELIZA

A lógica primária implementada no sistema ELIZA consistia na procura de padrões e de palavras-chave nas frases recebidas como entrada, de forma a efetuar o seu mapeamento com um conjunto de regras pré-programadas no sistema. Após efetuado o mapeamento, o sistema devolvia para o utilizador, as respostas pré-estabelecidas de acordo com essas regras [37], [38].

Por exemplo, dada uma frase do género "Eu costumava sair bastante com o meu pai durante a minha infância", o procedimento natural de ELIZA seria o seguinte, na respetiva ordem [38]:

→ Receber o *input* e armazená-lo em memória, para análise posterior.

→ Procurar por palavras-chave na frase recebida , estando estas palavras-chave já previamente definidas aquando da construção do programa. Neste exemplo podemos pensar em palavras-chave como:

  – "mãe", com uma resposta associada do tipo "diz-me mais acerca da tua mãe".

  – "pai", com uma resposta associada do tipo "diz-me mais acerca do teu pai".

→ Se as palavras presentes na frase de entrada, mapearem com as palavras-chave, o sistema devolve a resposta pré-programada para o utilizador.

→ Se não existir nenhuma correspondência clara entre as frases e o conjunto de palavras-chave, o sistema iria devolver uma resposta genérica como por exemplo "Desculpa, não sei nada sobre esse assunto!".

Em ELIZA, todas as resposta devolvidas são escolhidas pelo utilizador que projetou o sistema. Através do fornecimento de uma resposta formulada de acordo com os passos acima, é criada como já referido, uma ilusão de compreensão e inteligência, devido ao facto de esta responder e conseguir estabelecer uma conversa com um utilizador, mesmo que da perspectiva do desenvolvimento do sistema, este seja um método simples e trivial.

2.3.3 PARRY

Em 1972, *Kenneth Colby* da universidade de *Stanford*, desenvolveu um sistema conversacional ao qual deu o nome de PARRY, um Chatbot cuja personalidade correspondia a um paciente esquizofrénico [39].

Colby desenvolveu este sistema com o propósito de obter uma melhor compreensão acerca da mente de um paciente paranóico, tendo sido reportado pela sua equipa de desenvolvimento, que este sistema tinha concluído com sucesso, uma aproximação do teste de Turing apresentado anteriormente [40].

Na experimentação realizada, um grupo de psiquiatras tinha como tarefa, gerar conversas com pacientes reais e com o sistema PARRY, enquanto que outro grupo de psiquiatras analisava as conversas geradas, tendo como responsabilidade, atribuir se as conversas em questão, estariam a ser efetuadas por um ser humano real ou pelo sistema desenvolvido [40].

É importante notar que este teste representou uma aproximação ao original devido ao facto dos interrogadores apenas conversarem ou com um humano ou com o sistema computacional em cada fase, invés dos dois em simultâneo, como delineado por *Turing*.

O grupo de psiquiatras apenas conseguiu acertar corretamente 48% das conversas, ou seja, representava quase uma medida de acerto aleatória, demonstrativa da qualidade do sistema.

Apesar do facto de PARRY representar um sistema computacionalmente mais avançado, quando comparado com ELIZA, ambos tinham como base, técnicas de correspondência entre padrões e pequenas acções previamente declaradas, como por exemplo, devolver uma pequena história sem enquadramento, para os casos em que este não conseguisse efetuar nenhum mapeamento entre o seu domínio de respostas e a questão fornecida pelo utilizador [35],[40].

Em termos de abordagem conversacional, o sistema PARRY apresenta melhorias significantes quando comparado com os sistemas desenvolvidos até aquela data, introduzindo novos atríbutos como por exemplo a capacidade de mudança contextual, realizada através da devolução de respostas do tipo, "Porque é que perguntas isso?", tentando desta forma, "esconder" a sua incapacidade de resposta [41]. Adicionalmente, PARRY era capaz de admitir ignorância para os casos em que não conseguia de facto, formular uma resposta coerente para devolver ao utilizador, retornando respostas como "Eu não sei" [42].

Para além disso, PARRY apresentava um certo estado emocional para o utilizador, direccionando-o a responder com "raiva" se as diversas interacções realizadas com este o levassem para um estado de fúria [41]. Um exemplo interaccional ocorrido neste sistema é apresentado na figura 2.6 [35].

Contrariamente ao que se sucedia em ELIZA, cujo funcionamento se baseava na devolução de respostas curtas e sem grande continuidade de assunto, o sistema PARRY revela-se capaz de continuar de forma consistente, um certo tópico conversacional, tornando-o mais "humano" do ponto de vista do utilizador.

PSIQUIATRA: *Good afternoon. My name is Doctor Jones.*

PARRY: *Pleased to meet you.*

PSIQUIATRA: *How are you?*

PARRY: *All right I guess.*

PSIQUIATRA: *Could you tell me your name?*

PARRY: *Pat Smith*

PSIQUIATRA: *How old are you?*

PARRY: *28.*

PSIQUIATRA: *I thought you might have some problems.*

PARRY: *I've been upset lately.*

Figura 2.6 - Exemplo de interacção conversacional presente no Chatbot PARRY.

2.3.4 A.L.I.C.E

Em 1995, *Richard Wallace* criou A.L.I.C.E, um Chatbot significativamente mais complexo, inspirado no sistema ELIZA, e que utiliza uma extensão de XML chamada *Artificial Intelligence Markup Language* (AIML) para especificar os pares de padrões e respostas a serem utilizados pelo sistema [43]. Este foi um dos sistemas conversacionais com mais sucesso tendo sido a sua primeira versão implementada, com o passar do tempo, em outras linguagens de programação como Java, C e PHP, o que é demonstrativo da sua qualidade funcional [35].

Na arquitetura do sistema ALICE, ao contrário do que acontece nos dois sistemas anteriores, o "motor conversacional" e o "domínio de conhecimento" estão claramente separados, permitindo desta forma que a base de conhecimento possa ser alterada de maneira a conter, mais ou menos informação, sem influenciar o funcionamento geral do sistema [43], [44].

Apesar de diferentes em termos de complexidade e de funcionamento operacional, estes três Chatbots apresentados anteriormente, têm em comum o facto de se basearem na correspondência entre padrões de forma a tratar as palavras contidas na entrada fornecida pelo utilizador. ALICE introduziu uma nova abordagem para realizar esta correspondência, efetuada a partir da utilização de regras construídas em AIML. Este tipo de linguagem possui um certo tipo de *tags*, denominadas *srai*, que permitem ao sistema efetuar as correspondências entre os diversos padrões de forma recursiva, simplificando e reduzindo significativamente, a quantidade de declarações a efetuar quando comparado com os dois outros sistemas [44]. As figuras 2.7 e 2.8 ilustram exemplos de declarações AIML, para a realização de correspondências no sistema ALICE.

```
<categoria>
    <padrão> Qual é o teu nome? </padrão>
    <resposta> O meu nome é Alice </resposta>
</categoria>
```

Figura 2.7 - Exemplo de uma declaração AIML simples, presente no sistema ALICE.

```
<categoria>
    <padrão> Sabes quem é o * ? </padrão>
    <resposta><srai> Quem é  o <star/> <srai></resposta>
</categoria>
```

Figura 2.8 - Exemplo de uma declaração AIML recursiva, presente no sistema ALICE.

Se a frase fornecida como entrada corresponder à frase declarada, como demonstrada nos exemplos anteriores, a resposta associada para essa mesma pergunta seria devolvida pelo sistema, sem qualquer tipo de alteração em termos de construção frásica ou disposição gramatical, podendo variar apenas, o que se encontra declarado recursivamente com um asterisco. Por exemplo, fornecendo uma frase "Sabes quem é o André?", ALICE devolveria a resposta, "Quem é o André".

Umas das falhas apontadas a este sistema e que o levaram a falhar o teste de *Turing*, foi o facto destes sistemas baseados em AIML, falharem no estabelecimento de conversas de longa duração [41].

2.3.5 JABBERWACKY

Jabberwacky foi um sistema conversacional criado por *Rollo Carpenter* em 1997, com o objetivo de, segundo o autor, "*simular a conversa natural humana, de uma maneira interessante, divertida e bem-humorada*", ou seja, era um sistema de domínio aberto, sem qualquer personalidade adjacente e sem qualquer público-alvo previamente estabelecido [45]. Este sistema revolucionou de certa forma, a tecnologia de desenvolvimento dos Chatbots até aquela data, devido ao facto de ter sido o primeiro sistema a fugir a bases de conhecimento estáticas, procurando uma abordagem mais dinámica e generativa [46].

O sistema tinha a capacidade de colecionar e adicionar os vários pares de perguntas e respostas, efetuados entre si e os seus utilizadores, à sua própria base de conhecimento, tornando esta dinâmica e capaz de retornar diferentes respostas às mesmas perguntas, algo que nunca tinha sido realizado até à data [46]. No entanto, o termo generativo não é o mais correto para caracterizar o sistema devido ao facto de este não ser capaz de gerar respostas autónomas, mas sim apenas devolver algo pré-declarado no seu universo de conhecimento, mesmo que este seja mais amplo e diversificado quando comparado com os sistemas apresentados anteriormente.

Foi esta nova abordagem, que permitiu a este sistema vencer o prémio *Loebner*, uma competição anual, em tudo semelhante ao teste de *Turing*, responsável por premiar os sistemas conversacionais, considerados pelos jurados como, os que melhor representam a capacidade conversacional humana.

2.3.6 CHATBOTS MODERNOS

No final do ano de 2015, assistiu-se à primeira grande onda de inteligência artificial a tomar forma nos sistemas conversacionais. Plataformas sociais como o *Facebook* começaram a disponibilizar as ferramentas necessárias aos desenvolvedores de *software,* para estes criarem os seus próprios Chatbots, permitindo assim às diferentes marcas e serviços possuir o seu próprio sistema conversacional, tendo em vista a obtenção de uma melhor interacção com os seus consumidores.

A arquitetura e a estrutura destes sistemas sofreu uma grande evolução até aos dias de hoje, tendo sido formuladas diversas técnicas para a geração da sua resposta, nas quais se incluem metodologias baseadas nos resultados da realização de pesquisas *Web,* em algoritmos de classificação oriundos de técnicas de ML e na adoção de modelos generativos para construir o seu *output*, através do uso técnicas de NMT e de NLP*,* capazes de "traduzir" as frases recebidas como entrada*,* nas respostas devolvidas como saída.

Nos dias de hoje, os Chatbots já se encontram num estado muito mais avançado e são o foco de grandes empresas mundiais como demonstram por exemplo os casos dos sistemas *Echo* da *Amazon,* da *Siri* criada pela *Apple* ou do sistema desenvolvido pela *Microsoft, Cortana*. Todas estas assistentes pessoais possuem a capacidade de responder às diferentes perguntas do seu utilizador, tendo como base mecanismos de reconhecimento de aúdio e de extracção de informação na *Web*. Recentemente, a sua capacidade foi melhorada para efetuar pesquisa baseada em reconhecimento de imagem.

Na área da saúde, são inúmeros os Chatbots desenvolvidos e implementados para diversas aplicações: O Chatbot *Your.MD*, desenvolvido em 2012, é um sistema conversacional que possui associadas ao seu desenvolvimento e manutenção, cerca de 50 pessoas, o que demonstra o seu grau de complexidade [47]. O Chatbot tem como base a utilização de técnicas de IA e de ML, de maneira a fornecer informação médica genérica, bem como a divulgação de novos produtos e serviços clínicos [48]. Neste sistema, os algoritmos nele implementados são treinados sobre uma série de obras literárias medicinais previamente validadas, permitindo ao sistema aprender diversos sintomas comuns e fornecer recomendações ao seu utilizador [48].

Outro Chatbot que tem vindo a obter imenso sucesso, aquando da sua implementação e disponibização, tem o nome de *Florence* e diz respeito um sistema que tem como propósito, funcionar como enfermeira virtual [49]. A sua aplicabilidade baseia-se na capacidade do sistema em relembrar os seus utilizadores, da hora a que estes devem tomar a sua medicação, sendo portanto, um sistema bastante utilizado por utilizadores com idade acima dos 65 anos [50]. Para além desta funcionalidade, *Florence* pode ainda efetuar rastreios de saúde relativos ao peso corporal e ao estado emocional do utilizador, interagindo com este quando são verificadas oscilações comportamentais quando comparado com aquilo que é considerado normal pelo sistema [50].

Estes são apenas alguns exemplos de Chatbots desenvolvidos e já implementados nos dias de hoje, demonstrativos da evolução e da aplicabilidade destes sistemas conversacionais quer na área da saúde, quer na vida pessoal em geral, dos diferentes utilizadores.

CAPÍTULO 3
TECNOLOGIA DE UM CHATBOT E PERCEPÇÃO HUMANA

Depois de efetuada a contextualização ao conceito de um Chatbot, às suas diferentes formas de implementação e aos vários sistemas desenvolvidos ao longo dos anos, serve este capítulo para efetuar um enquadramento ao leitor, das técnicas e metodologias utilizadas para a construção nos dias de hoje, de um Chatbot do tipo generativo.

Devido ao facto da sua construção se basear na conjugação de técnicas de processamento de linguagem natural (NLP) e de *Deep Learning*, torna-se importante efetuar uma contextualização a ambos os processos, com vista um melhor entendimento por parte do leitor, nas fases mais adiantadas desta dissertação onde estes são referenciados.

Inicialmente é construída uma sub-secção para o tema de NLP, na qual se expõe no que se baseia todo o processo e apresentando as suas principais técnicas. Os conceitos referentes ao *Deep Learning* são apresentados de seguida, através da elaboração de uma sub-secção relativa a ferramentas e métodos, onde é realizada a apresentação e descrição das tecnologias e metodologias utilizadas, aquando da projecção do Chatbot proposto nesta dissertação, explicitando a sua aplicabilidade e o porquê da sua escolha.

3.1 PROCESSAMENTO DE LINGUAGEM NATURAL

A nossa comunicação interpessoal é realizada de diversas maneiras: através da fala e audição, por comunicação gestual, pela realização de sinais ou através de várias formas **textuais**. Por comunicação textual, referimo-nos a um conjunto de palavras escritas em qualquer superfície ou exibidas em algum dispositivo eletrónico, tendo em vista a sua transmissão para o receptor [51].

O processamento computacional de linguagem natural concentra-se apenas no último caso: nesta dissertação estaremos preocupados com as diversas maneiras pelas quais os sistemas computacionais podem analisar e interpretar textos, assumindo por conveniência que estes são apresentados em formato digital [51].

Como acontece em qualquer tentativa de definição, quando se procura por uma declaração universal do que é o processamento de linguagem natural, encontrar-se-á um amplo leque de diferentes propostas. Qualquer que seja porém a perspectiva em que se encare o processo de NLP (acrónimo do inglês, *Natural Language Processing*), um dos aspectos centrais desta área consiste em lidar com o conhecimento linguístico, modelando as suas regularidades e sobretudo o modo de como a forma e os significados linguísticos, se encontram associados de forma sistemática [52].

Para os efeitos desta dissertação, podemos resumir NLP como sendo uma área especifica da IA, que tem como propósito, o estudo e o desenvolvimento de técnicas que permitam a análise e compreensão da linguagem humana por parte de um sistema computacional [53].

Ao contrário do que acontece na maioria dos processadores textuais existentes, cujo funcionamento se baseia na aplicação de expressões regulares e na correspondência de padrões, as técnicas de NLP levam em consideração a estrutura hierárquica da linguagem, isto é, o facto de que **letras** formam **palavras** e palavras formam **frases**.

O NLP é visto como um problema de difícil resolução no campo da ciência computacional, devido à ambiguidade associada às diferentes linguas existentes. Quando uma máquina necessita de compreender uma língua, esta necessita não só de saber e compreender o significado das palavras, mas também os conceitos e como estes se encontram interligados, de maneira a extrair o significado inerente a um dado texto [53].

O processo de compreensão de linguagem natural, divide-se de uma forma comum, em 3 passos distintos, discutidos e apresentados ao longo desta secção:

- → Análise **Morfológica**
- → Análise **Sintática**
- → Análise **Semântica**

3.1.1 ANÁLISE MORFOLÓGICA

A análise morfológica concentra-se em analisar os elementos "dentro" das palavras, estando esta encarregue do estudo da sua estrutura, formação e classificação. A morfologia reconhece as palavras em termos das suas unidades primitivas, a que se dá o nome de morfemas, que as compõe (e.g. caçou → caç + ou) [54].

Neste primeiro passo da NLP, as palavras são individualmente extraídas do texto a processar, sendo esta acção realizada com base em delimitadores textuais como é o caso da pontuação e dos espaços em branco. Depois de individualizados, os componentes de um dado texto são analisados um a um, sendo feita a identificação da sua classe gramatical, do seu lema e do seu radical [53], [54]. A classe gramatical identifica qual é a função dessa palavra, ou seja, se esta diz respeito a um verbo, a um nome, a um pronome, etc.

De seguida são apresentados os termos e técnicas mais importantes e mais utilizadas, aquando da análise morfológica.

- **Tokenização – *Tokenization***

A tokenização é o processo no qual um dado texto, é partido e individualizado em símbolos, palavras, frases ou outros elementos textuais, aos quais se dá o nome de *tokens* [51].

Resumidamente, o seu funcionamento baseia-se na conversão da entrada textual fornecida ao sistema, em fragmentos, de modo a que estes possam ser processados e utilizados em tarefas posteriores por parte do utilizador, como por exemplo, na segunda fase do processo de NLP, relativa à análise sintática, e em conjugação com os modelos de *Deep Learning* para geração conversacional [55].

Adicionalmente, o processo de tokenização permite a identificação de palavras sem significado num determinado texto, fornecendo assim ao seu utilizador, a capacidade de remover componentes não desejados por este, reduzindo desta forma a quantidade de dados sem relevância, a serem processados.

Neste nível, as palavras não são classificadas em categorias gramaticais nem tão pouco nos permite a indicação da sua estrutura sintática, ainda assim, uma boa quantidade de informação pode ser obtida a partir de uma análise, relativamente superficial, de um texto que sofreu tokenização [53], [55].

Por exemplo, dada a frase "Quem é que hoje alinha em pescar?", o *tokenizer* (algoritmo que efetua o processo de tokenização), seria responsável por dividir o texto nos seguintes *tokens*:

→ *Quem*
→ *é*
→ *que*
→ *hoje*
→ *alinha*
→ *em*
→ *pescar*
→ *?*

- **Palavra vazia – *Stop-Word Removal***

O processo *Stop-Word Removal* é uma das etapas de pré-processamento mais utilizadas nas diferentes aplicações de NLP. A ideia a si adjacente consiste na remoção das palavras que ocorrem com maior frequência numa dada lingua.

Por exemplo, no caso mais genérico, a lingua inglesa, as *stop-word* (em português, palavras vazias) seriam palavras como "a", "the" e "are","in", enquanto que na lingua portuguesa poderiamos identificar palavras como "a","o","do","são" como sendo o nosso conjunto de palavras vazias [56].

Estes termos ocorrem com muita frequência e não podem ser utilizados para separar diferentes tópicos conversacionais presentes num dado texto, pois são utilizadas de igual forma ao longo deste. Devido ao facto de estas palavras não serem úteis na classificação e na extracção do significado acoplado a um dado excerto textual, então estas podem ser removidas durante o seu pré-processamento [57].

A inconsistência dos termos quanto à sua relevância, de domínio para domínio, torna díficil a criação e disponibilização de listas de *stop-word* genéricas, requerendo por isso, uma análise prévia por parte do utilizador do domínio textual em questão, aquando da sua formulação e declaração.

Ainda assim, este processo revela-se bastante útil devido ao facto de reduzir substancialmente o tamanho do conjunto de dados considerados relevantes, e de aumentar o desempenho na tarefa de classificação de texto [57].

- ***Stemming***

Stemming diz respeito a uma técnica de análise lexical, utilizada para diminuir a lista de palavras indexadas num dado documento. Este método permite reduzir os vários termos que se encontram em formas derivadas, para a sua base, removendo as variações das diferentes palavras como o seu plural, o seu gerúndio, prefixos, sufixos, o seu género e número, de modo a que estas fiquem só com a raiz, à qual se dá o nome de *stem* [56],[58].

Por exemplo, a ocorrência de palavras conjugadas como "viajo", "viajei", "viajando" num dado texto, podem todas ser transformadas num único *stem* – "viaj" – referente ao acto de viajar.

Esta técnica é maioritariamente utilizada para facilitar a correspondência entre diferentes documentos de texto, que possuam o mesmo conteúdo [59]. Na sua forma mais comum, existem quatro tipos distintos de algoritmos de *stemming*, apresentados na figura 3.1. De seguida é efetuada uma breve descrição de cada um destes algoritmos.

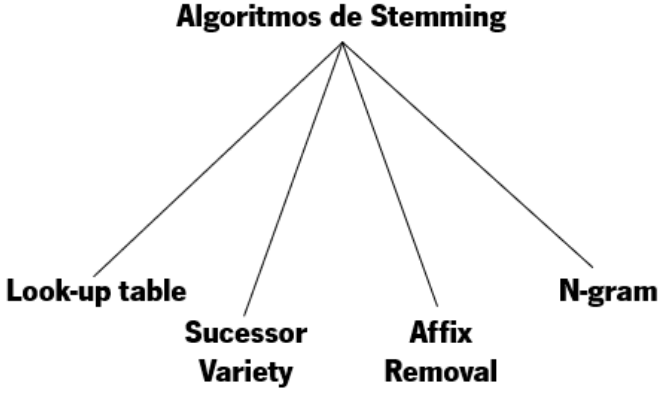


Figura 3.1 - Príncipais algortimos de Stemming

→ ***Look-up Table stemmers***

O primeiro algoritmo de *stemming* apresentado tem por base um conjunto de entidades chamadas *look-up tables*.

Este consiste no armazenamento manual, sob a forma de uma tabela, do maior número possível de palavras indexadas num dado texto, associando a cada uma destas, os seus *stems* correspondentes [59], [60]. Após formulada, os elementos são acedidos e procurados na tabela sob a forma de uma *query.*

Esta procura de palavras conjugadas na tabela é chamada de *look-up* e possui a vantagem de serem procuras extremamente rápidas, mas que por outro lado, possuem a desvantagem de necessitarem de uma capacidade de armazenamento que acompanhe o número de *stems* pretendido [57], [61].

Para além disso, o facto de serem poucas as tabelas completas, já formuladas e disponibilizadas para uso geral, com palavras conjugadas na maior parte das línguas e domínios, tornam muitas das vezes esta abordagem insuficiente quando as línguas em questão possuem um elevado grau de complexidade sintática e gramatical, levando a uma carga de trabalho extensa e muitas das vezes não recompensadora [57], [61].

→ ***Successor variety stemmers***

Os algoritmos de *stemming* do tipo *successor variety* foram propostos por *Hafer* e *Weiss* em 1974 [62], e tentam, após uma rebuscada análise da frequência e do tamanho dos prefixos ao longo das variações morfológicas de uma palavra, identificar heuristicamente o seu potencial radical [63]. O funcionamento deste algoritmo é de fácil compreensão recorrendo à ajuda de um exemplo demonstrativo [57]:

Supondo que num dado excerto textual possuímos o seguinte conjunto de palavras na lingua inglesa:

[*"walk"*, *"world"*, *"warm"* , *"like"*]

Se observarmos a variação da sucessão da letra na palavra *"warm"*, a primeira letra do termo corresponde à letra "w". Confrontando esta letra, com o resto dos termos presentes no conjunto, verificamos que esta letra é sucedida ou por a letra "a" (nos termos *"warm"* e *"walk"*) ou pela letra "o" (no termo *"world"*), permitindo então registar a frequência da sua letra sucessora como sendo igual a dois.

De seguida, realizando a contagem da frequencia relativa ao prefixo "wa", verificamos que este é seguido ora por a letra "l" (no termo *"walk"*) ora por a letra "r" (nos termos *"world"* e *"war"*), sendo a sua contagem também igual a dois. A combinação de letras *"war"*, por sua vez só é seguida por a letra "m", fazendo com que a sua contagem seja igual a um, terminando aqui o processamento para a sequência de letras em análise.

Aplicando toda esta lógica a todos os termos presente no conjunto de dados a processar, conseguimos criar uma série de contagens, demonstrativas da variação de letras ou de combinação de letras, para todos os termos presentes num dado texto.

Estas contagens são depois utilizadas para segmentar os termos nos seus *stems* correctos, baseando-se para tal na frequência de maior valor [59], [60].

→ **Algoritmos de *stemming* do tipo *n-gram***

Os algoritmos do tipo *n-gram* dividem os termos presentes num dado excerto de texto, em partes com *n* letras consecutivas, sendo estas partes depois comparadas entre todos os termos presentes num dado documento de texto [57], [59].

Por exemplo, dadas as palavras em inglês, *"walking"* e *"walker"*, estas podem ser comparadas através da criação de bigramas, isto é, para $n = 2$ (daí o termo *n-gram*), da seguinte forma [57]:

- *Walking* = [wa al lk ki in ng]
- *Walker* = [wa al lk ke er]

Como podemos observar, o termo *"walking"* pode ser desmembrado em seis bigramas únicos, enquanto que o termo *"walker"* pode ser estruturado sobre cinco bigramas distintos, partilhando os dois termos, três bigramas entre eles: ["wa","al","lk"].

Após extraídos os bigramas comuns aos dois termos, é utilizada uma medida de similaridade para efeitos comparativos entre os dois termos, à qual se dá o nome de coeficiente de *Sørensen-dice, S,* :

$$S = \frac{2C}{A + B}$$

onde A e B representam respectivamente, o número de bigramas únicos no primeiro e no segundo termo e C representa os bigramas comuns. No exemplo acima descrito, o coeficiente de *dice* seria igual a $\frac{2\times3}{6+5} = \frac{6}{11}$.

Após formulados os bigramas e calculados todos os coeficientes de *Dice*, entre os diferentes termos presentes num dado excerto textual, estes são representados sob a forma de uma matriz, à qual se dá o nome de matriz de similaridade [61]. Os limites estabelecidos para os coeficientes de *Dice*, determinam se os termos são *stemizados* sob a mesma base.

Uma vez construída a matriz de similaridade, os termos são agrupados utilizando um método de aglomeração de ligação simples *(em inglês, single link clustering method)*, como o descrito por *Croft* [64].

Este algoritmo possui a vantagem de ser independente da língua em que o texto se encontra, não interessando a complexidade da mesma, sendo portanto bastante útil em diversas aplicações[53].

A desvantagem deste, reside no facto de requerir uma quantidade significativa de memória e de capacidade de armazenamento, de maneira a criar e a armazenar todos os *n-grams* pretendidos, bem como os seus índices correspondentes, tornando-o por vezes um método de difícil aplicação na prática [53].

→ ***Stemmers* de remoção de afixos – *Affix Removal Stemmers***

Os algoritmos do tipo de remoção de afixos, como o nome indica, focam-se na remoção de sufixos ou prefixos, acoplados aos termos presentes num determinado excerto de texto [59], [65].

Por exemplo, dado um *stemmer* encarregue de remover o afixo "ndo" dos termos presentes num dado conjunto de dados textuais, palavras como "envelhecendo" seriam transformadas em "envelhece", possuíndo este a si acoplada a desvantagem de palavras como "mando" se transformarem em "ma".

Devido ao facto deste método não ser muito preciso nem ter em atenção o contexto da remoção efetuada, não é um método muito aconselhável para realizar o processo de *stemming* [57], [60].

Todos os algoritmos de *stemming* apresentados podem cometer erros e não ser 100% fiáveis, maioritariamente devido ao facto de existirem línguas, cuja aplicação do processo se revela um grande desafio.

Esses erros podem ser divididos em dois grandes tipos [65]:

- *Over-stemming* – Quando dois ou mais termos sofrem stemização para a mesma raiz, sem ser suposto, sendo chamado de falso-positivo. Por exemplo, os termos "universal" ,"universidade", e "universo" são reduzidos ao termo "univers". Este é um caso de overstemming: apesar destas três palavras estarem relacionadas etimologicamente, o seu signficado é diferente nos diferentes domínios, e portanto, tratar estes termos como sinónimos resultaria em bastantes incongruências e reduziria a fiabilidade dos resultados.
- *Under-stemming* – Ocorre quando duas palavras deveriam ser reduzidas por meio de um algoritmo de *stemming,* ao mesmo termo, mas não o são, representando falsos-negativos. Um exemplo de *understemming* é apresentado de seguida: Enquanto que os termos "refere" e "referir" seriam reduzidos ao *stem* "refer", o termo "referência" poderia seria reduzido ao termo "referên". Esta não aglomeração correta, de termos com domínios e significados idênticos, pode resultar numa falsa, ou não associação de termos, que na verdade pertencem ao mesmo domínio contextual.

- **Lematização – *Lemmatization***

O processo de lematização, semelhantemente ao que ocorre no processo de *stemming*, possui como objetivo reduzir termos conjugados à sua forma original.

No entanto, enquanto que o processo de *stemming*, tem como procedimento base habitual, retirar a parte final de uma palavra para obter o seu *stem*, não tendo qualquer conhecimento acerca do contexto, o processo de lematização utiliza um certo tipo de vocabulário, permitindo-lhe desta forma, ter em consideração a análise morfológica do termo em questão, retornando a palavra base a si associada, à qual se dá o nome de lema [59].

Um exemplo comparativo dos algoritmos de lematização e de stemming, é que o primeiro conseguiria transformar a palavra, "melhor" , para o seu lema correto, "bom", enquanto que um algoritmo de *stemming* era incapaz de a reduzir por não possuir noção do contexto da palavra.

- **Expansão automática de pesquisa – *Automatic Query Expansion***

O processo de expansão automática de pesquisa (em inglês, tratado como o fenómeno de realizar uma *query*), tem como objetivo, reformular a *query* base, de forma a aumentar o seu universo de resultados, sendo um processo bastante utilizado pelos motores de pesquisa como o *Yahoo* ou o *Google* [57]. Por exemplo, a pesquisa num destes motores por "vp de marketing", seria reformulada para:

(vp ou vice-presidente) de marketing,

permitindo desta forma, otimizar e devolver um maior conjunto de resultados quando comparada com a pesquisa inicial. Esta reformulação de *query* envolve diversas técnicas como [57]:

→ Encontrar sinónimos dos termos a serem pesquisados, acrescentando-os à pesquisa realizada.

→ Encontrar termos relacionados semânticamente, isto é, antónimos ,merónimos, hiperónimos, hipónimos, etc.

→ Encontrar todas as várias formas morfológicas das palavras , aplicando o processo de stemização a cada palavra na *query*.

→ Corrigir erros de ortografia e procurar automaticamente o termo correto, ou como alternativa, sugeri-lo nos resultados devolvidos.

3.1.2 ANÁLISE SINTÁTICA

O termo "Sintaxe" remete para a ciência responsável pelo estudo do processo de construção frásica. Nessa perpectiva, realizar a análise sintática, possui como objetivo, a identificação da função adjacente a cada elemento no contexto frásico em que este se encontra inserido, bem como da relação que este estabelece com os demais constituintes [55]. Saber que uma determinada palavra pertence a uma classe Gramatical específica, ajuda a determinar que tipo de palavras são mais prováveis de serem suas vizinhas, e.g. na Língua Portuguesa, um pronome pessoal é geralmente seguido de um verbo.

Podemos resumir a Análise Sintáctica, como sendo a conjugação da gramática (regras estruturais) com técnicas de *parsing* (análise/geração de frases, de acordo com a sua gramática). A análise sintática é vista como o segundo grande passo de NLP, no qual as sequências de palavras são transformadas em estruturas capazes de demonstrar a relação existente entre os vários termos presentes numa frase, podendo estas representarem um adjectivo, um verbo, um substantivo, etc.

- **Part-of-speech (PoS) tagging**

PoS tagging diz respeito a um processo de NLP cujo funcionamento consiste na classificação das palavras em termos de *Part-Of-Speech* (parte de discurso), indicativo da sua categoria lexical , estando desta forma relacionada com o seu comportamento sintáctico numa frase, bem como do campo semântico em que esta se encontra inserida [55]. Resumidamente, o seu objectivo consiste em associar a cada termo presente numa dada frase, a sua classe gramatical, isto é, se um dado termo presente num excerto textual diz respeito a um nome, um verbo, um adjetivo, etc, baseando-se para tal na sua definição e no contexto em que este se encontra inserido [57].

O processo *PoS tagging* revela-se tendencialmente uma tarefa complicada, devido ao facto de não ser um método genérico, sendo possível a ocorrência de casos em que uma única palavra, tenha uma categoria gramatical diferente em frases distintas, em virtude de estarem inseridas em contextos distintos, tornando desta forma impossível obter um mapeamento genérico para a atribuição da classe gramatical às diferentes palavras [66]. Um exemplo desta ambiguidade é a palavra prova , que na frase "A sua atitude prova o seu caráter" é classificada como um verbo, mas na frase "A prova foi difícil" é definida como sendo um substantivo.

Este é um processo importante no contexto de processamento de linguagem natural, nomeadamente na parte geração de resposta por um sistema conversacional, tendo como função a indicação para este, do tipo de *PoS* a ser formulado na resposta a devolver ao utilizador [67].

Adicionalmente é uma técnica de pré-processamento, bastante, utilizada para anteceder à técnica de *parsing,* apresentada de seguida.

- **Parsing**

O *parsing* frásico diz respeito ao processo de transformação de uma sequência de palavras numa determinada estrutura gramatical. Em termos computacionais, esta técnica é capaz de retornar uma árvore de análise, demonstrativa de todas as correlações sintáticas existentes entre os termos da frase recebida como entrada [55], [57].

Tendo em conta a necessidade de analisar sintáticamente e compreender a linguagem natural como o seu todo, existem duas técnicas principais para o *parsing* de textos em linguagem natural: *full parsing* (análise completa) e *shallow parsing* (análise superficial ou parcial).

As principais diferenças entre estas duas técnicas, residem no detalhe e profundidade que apresentam as *parse trees* devolvidas por estas, bem como na complexidade processual associada a cada técnica [55]. A forma de representação em àrvore, é a representação mais natural devido aos diferentes constituintes frásicos possuirem dependências hierárquicas [68].

A abordagem mais tradicional para efetuar o *parsing* de um texto em linguagem natural é recorrendo à técnica de *full parsing*, que consiste na utilização de uma *feature-based grammar* (gramática com base em recursos), capaz de efetuar a validação formal frásica, de acordo com as regras gramaticais da linguagem [55]. A análise sintáctica realizada do tipo *full parsing* resulta numa representação em árvore de toda uma frase, constituída por todos os elementos individuais que a constituem [69].

A técnica *shallow parsing*, por outro lado, procura fornecer informação sobre a estrutura de um determinado conjunto de frases, sem que para tal, construa representações hierárquicas completas destas, isto é, efetua uma análise de uma determinada frase, na qual identifica os seus constituintes (nomes ou verbos) mas não especifica a sua estrutura interna. O funcionamento desta é geralmente baseado em regras e técnicas de aprendizagem similares às utilizadas no processo de *POS Tagging* apresentado anteriormente [69]. Em vez de efectuar uma análise completa de toda uma frase, esta só é responsável pela análise de certas unidades sintácticas que são fáceis e que não introduzem ambiguidade, sendo tipicamente, geradas frases preposicionais (PP), substantivas (NP) e verbais (VP) pequenas e simples, e que ao contrário do que acontece na análise completa, não se podem interrelacionar [69].

Para conseguir este objectivo é habitual recorrer-se a um processo designado por *chunking*. Este consiste na divisão do texto em segmentos não-sobrepostos e não-recursivos, ou seja, os diversos elementos não podem estar contidos mutuamente (chamados de *chunks*).

- **Modelo saco de palavras – *Bag-of-words***

O modelo *Bag-Of-Words (BoW)* é um modelo simples de representação textual, bastante utilizado em algoritmos de *Machine Learning*. A sua abordagem consiste em retratar um texto como um conjunto visto apenas pelos elementos que o constituem, ignorando qualquer relação entre os diferentes termos bem como a parte gramatical a estes adjacente [66]. O modelo apenas representa as palavras presentes num dado texto e sua multiplicidade, sendo uma tarefa bastante utilizada por exemplo, na classificação de documentos [57]. De seguida é apresentado um exemplo da aplicação da técnica, retirado de [70], relativo a um dado excerto textual, escrito em inglês:

"It was the best of times,
it was the worst of times,
it was the age of wisdom,
it was the age of foolishness
(...)"

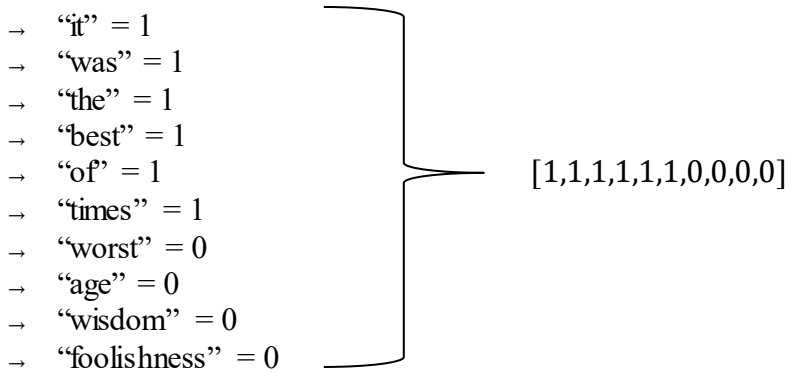
Tratando cada frase como um documento ou texto único, e as quatro frases como o nosso conjunto de documentos, o primeiro passo do modelo seria desenhar o seu vocabulário através da análise do conjunto apresentado. O vocabulário, constituído apenas por palavras únicas seria neste caso, ignorando pontuação, o seguinte:

→ *"it"*
→ *"was"*
→ *"the"*
→ *"best"*
→ *"of"*
→ *"times"*
→ *"worst"*
→ *"age"*
→ *"wisdom"*
→ *"foolishness"*

Este é composto por 10 termos únicos, retirados de um conjunto de textos constituido por 24 termos. Depois de criado o vocabulário, o passo seguinte diz respeito à criação de vetores representativos da ocorrência dos termos nos diferentes documentos: O objetivo consiste em transformar cada documento composto por texto livre, num vector único, que possa ser utilizado como entrada ou como saída, em diferentes técnicas de *Machine Learning* e de *Deep Learning*.

Dado que o tamanho do vocabulário já é conhecido, pode ser usada uma representação de documentos com tamanho fixo de 10 unidades, tentando corresponder cada palavra do vocabulário ao documento a comparar. A maneira mais usual de fazer esta correspondência é através de marcação desta com um valor boleano: 0 se a palavra do vocabulário não se encontra no texto a comparar e 1 se esta for correspondida.

Utilizando a ordenação arbitrária de palavras listadas acima como nosso vocabulário, podemos passar pelo primeiro documento ("*It was the best of times* ") e convertê-lo num vetor binário da seguinte maneira:



Os vetores correspondentes aos outros documentos seriam então representados da seguinte forma:

→ *"it was the worst of times"* = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]

→ *"it was the age of wisdom"* = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]

→ *"it was the age of foolishness"* = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]

Toda a ordem das palavras é nominalmente descartada, obtendo assim uma maneira consistente de extrair características de qualquer documento presente no nosso *corpus linguístico*, prontos para serem utilizados em modelos de *Machine Learning* e NLP.

No fundo, o modelo *BoW* é basicamente uma ferramenta de conversão de texto, para um vetor representativo de todas as suas características, a partir do qual é se torna possível inferir o seu contexto e o seu conteúdo.

- **TF-IDF – Term Frequency-Inverse Document Frequency**

O TF-IDF diz respeito a um algoritmo que tem como propósito, indicar a importância ou o peso de uma palavra contida num documento, em relação a um determinado *corpus* linguístico. É importante observar que a proposição de que todas as palavras com elevada frequência de ocorrência dizem respeito a palavras-chave, não é verdadeira, pois palavras muito comuns geralmente possuem baixa importância para o texto em questão, representando uma espécie de ruído para o sistema [71]. Para indicar o peso de uma determinada palavra, o algoritmo tem como base duas medidas importantes:

→ **Frequência do termo** (TF, *term frequency)*.

O TF (frequência do termo) de uma palavra diz respeito ao número de vezes que aparece que esta aparece num dado documento [71]. Por exemplo, quando um documento constituído por 100 palavras contém o termo "gato" 12 vezes, o TF da palavra "gato" é:

$$TF_{gato} = \frac{12}{100} \; i.e \; 0{,}12$$

→ **Inverso de frequência nos documentos** (IDF, *inverse document frequency*)

O IDF de uma palavra é a medida de quão significativo é esse termo, perante todo o *corpus* linguístico. Dado o exemplo retirado de [72], supondo que um determinado termo "gato", aparece *x* número de vezes num *corpus* linguístico, constituido por 10.000.000 de documentos .

Assumindo que existem 0,3 milhões de documentos que contêm o termo "gato", então o IDF (ou seja $\log\{DF\}$) é dado pelo número total de documentos (10.000.000) dividido pelo número de documentos contendo o termo *"gato"* (300.000) [72].

$$IDF_{gato} = \log\left(\frac{10000000}{300000}\right) = 1{,}52$$

Após conhecidas ambas as medidas, é possível calcular o seu TF-IDF, utilizado para medir a importância de um dado termo, através do número de vezes que esta aparece no documento. Mais importante ainda, refere o quão importante esta é num dado *corpus linguístico* [72]. Para tal é efetuado o produto entre as duas medidas apresentadas acima, de tal forma que:

$$\text{TFIDF} = \text{TF} * \text{IDF} = TF_{d,t} \log\left(\frac{N}{DF_t}\right),$$

em que:

→ $TF_{d,t}$ é o número de ocorrências de t num dado documento d.

→ DF_t é o número de documentos que contêm o termo t.

→ N é o número total de documentos que compõe o corpus linguístico.

Quanto maior for o valor do TFIDF, mais raro é o termo, enquanto que quanto menor o peso, mais comum é este. A titulo de curiosidade acerca da aplicabilidade desta técnica, *Berger et al.* [73], utilizaram uma versão adaptada do algoritmo TF-IDF no seu sistema conversacional. O algoritmo TF-IDF nele utilizado, possuía a capacidade de ajustar os pesos IDF de cada termo, de tal maneira que poderiam ser utilizados para maximizar a probabilidade de retornar a resposta mais correta para cada pergunta fornecida ao sistema.

3.1.3 ANÁLISE SEMÂNTICA

Até ao momento foram abordadas as etapas de análise morfológica, responsáveis pela divisão do conjunto textual em *tokens* e pela remoção de termos e caractéres indesejáveis, e as de análise sintática, nas quais as sequências de palavras são transformadas em estruturas capazes de demonstrar as relações existentes entre os vários termos. Todas as análises anteriores são importantes para a interpretação e classificação das frases, no entanto o seu significado real e o seu contexto ainda não foi referido.

Esta última tarefa é responsabilidade da análise semântica, que diz respeito ao último grande passo presente na NLP, concentrando-se em atribuir um certo significado acoplado a palavras, frases e textos, descrevendo o processo de compreensão da linguagem natural – i.e. o modo como os humanos comunicam – com base no significado e no contexto [74]. Conseguir perceber a frase "Eu vou fazer uma viagem pela Europa" não é uma tarefa simples de se realizar, uma vez que se revela necessária a tradução da linguagem natural para uma linguagem formal, sem ambiguidade, de maneira a que os sistema computacionais consigam interpretar o sentido das palavras [74].

Neste passo, são criadas estruturas com vista a representação do significado e do contexto adjacente a um dado termo ou conjunto de termos, inseridos numa determinada frase. Este é ainda um passo com bastante desenvolvimento e pesquisa, devido à complexidade inerente em encontrar soluções ótimas capazes de extrair automáticamente o significado e o contexto de um dado documento textual.

Após efetuada a análise semântica obtém-se um texto sob linguagem formal, sem ambiguidade, e passível de ser compreendido por um computador. Os métodos e as técnicas mais utilizadas na análise semântica, são apresentados e explicados ao longo desta sub secção.

- **Extração de informação**

O objetivo dos processos de Extração de Informação é representar, de uma forma automática, informação não estruturada de uma forma organizada, de maneira a que possa depois ser utilizada em métodos de *Machine Learning* e *Deep Learning*, bem como em motores de pesquisa. Esta tarefa de extracção de informação é composta por diversas técnicas, sendo a mais comum, o Reconhecimento de Entidades Mencionadas.

→ **Reconhecimento de entidades mencionadas – Named Entity Recognition (NER)**

O Reconhecimento de Entidades Nomeadas – *Named Entity recognition (NER)*, é uma das mais importantes ferramentas do Processamento de Linguagem Natural , referindo-se a uma tarefa de extração de informação, responsável por capturar as entidades presentes num um texto e classificá-las em categorias pré-definidas, tais como pessoas, empresas, locais, valores monetários, datas, entre outros, dependendo sempre do domínio da tarefa [75].

A execução da técnica NER é normalmente dividida em duas subtarefas, que ocorrem praticamente ao mesmo tempo [75]:

O primeiro passo diz respeito à identificação de entidades, sendo neste, identificadas corretamente todas as entidades presentes num dado texto não estruturado, sem efetuar a sua classificação. Normalmente estas entidades passam ser representadas sob a forma de *tokens* , ficando as palavras armazenada em vetores [75]. A técnica NER pode inclusive, ajudar na tarefa de identificação de classes gramaticais, o que lhe permite inferir desde logo algumas entidades, ajudando na classificação dos átomos vizinhos. O uso de listas pré-concebidas de entidades conhecidas, com vista a comparação de átomos, muitas vezes não é suficiente para a detecção da sua totalidade, uma vez que o conjunto de entidades possíveis vai sendo constantemente alargado [74].

O segundo passo diz respeito à classificação das entidades, onde são corretamente classificadas as entidades identificadas e armazenadas nos *tokens,* na etapa anterior.

Este algoritmo ainda está a ser bastante investigado e aprimorado, pelo que as abordagens atuais concentram-se em técnicas baseadas em *look-up tables,* na correspondência entre padrões ou através do uso de *triggers*[57], [76]. Enquanto que as técnicas baseadas no uso de *triggers* se baseiam na formulação de regras adjacentes a nomes próprios, e.g. o termo "Monte Fuji" seria reconhecido através do estabelecimento de um *trigger* para a regra [*Monte + letra maiuscula*] , nas técnicas de correspondência entre padrões, todo o padrão é construído manualmente por parte de quem o projeta, de forma a que por

exemplo, dado o padrão: *"<Nome> encontra-se em <Localização> durante <Data>"*, a frase "*André encontra-se em Nova Iorque durante 2018"* seria correspondida.

Ambas as técnicas, devido ao facto de não serem automatizáveis, tornam a sua implementação na prática, muito dificíl e dispendiosa.

No entanto esta técnica é amplamente utilizada e à qual se associa um elevando grau de importância devido às suas inúmeras aplicações práticas, tais como [57]:

- → Encontrar o nome de uma pessoa ou empresa num dado excerto de texto.
- → Saber quais pessoas ou empresas foram mencionadas num artigo, notícia ou comentário em rede social (às vezes, o interesse é analisar as reações das pessoas e da comunicação social a uma determinada campanha de marketing).
- → Possui um papel importantíssimo na construção de Chatbots, devido ao valor do contexto de uma conversa, na formulação das respostas a serem devolvidas. São inumeros os casos em que um termo pode ter mais que um significado, e portanto, a identificação do seu contexto revela-se crucial no desenvolvimento de um agente conversacional.

- **Word Sense Disambiguation (WSD)**

A Desambiguação Lexical de Sentidos (tradução do inglês *Word Sense Disambiguation,* WSD) corresponde à tarefa de determinar o sentido mais adequado de uma palavra, com base no contexto em que esta foi inserida (frase, texto, documento, etc.), através da construção de um conjunto finito de possíveis sentidos ou significados, declarados em dicionários, em ontologias, etc [77].

Esta tarefa pode ser considerada um problema de classificação, onde se pretende desambiguar palavras que podem assumir vários sentidos [74]. A partir de uma palavra e de uma lista de possíveis significados, a técncia de WSD escolhe o significado mais apropriado tendo em conta o seu contexto.

Geralmente, os métodos de WSD são aplicados em etapas de pré-processamento de dados para o desenvolvimento de sistemas conversacionais, de forma a que os problemas relacionados com ambiguidade de termos sejam eliminados ou pelo menos, amenizados, fazendo consequentemente, que melhores resultados sejam alcançados [77].

O problema abordado pela técnica WSD, é classificado como um problema completo da Inteligência Artificial, que apenas será solucionado na íntegra, quando as aplicações da IA forem capazes de "compreender" completamente a linguagem natural humana e forem capazes de assimilar conhecimento enciclopédico, isto é, capaz de evoluir com o tempo.

As abordagens atuais para solucionar este problema vão desde a construção de dicionários que contêm a informação neles embutida, passando pela utilização de algoritmos de ML supervisionados capazes de utilizar um classificador treinado para cada termo presente no *corpus linguístico* até ao uso de algoritmos não-supervisionados baseados em *Clustering,* capazes de agregar os termos que se encontram num contexto semelhante, conseguindo desta forma, inferir o sentido destes [77].

- **Latent Semantic Analysis (LSA)**

A técnica de LSA é uma técnica de NLP baseada na proposição, de que os termos com significados idênticos, ocorrem em documentos também semelhantes. A técnica baseia-se na capacidade de identificar corretamente os padrões existentes nas demais relações entre os termos e conceitos contidos num dado *corpus linguístico* [78].

- **Semantic Role Labelling (SRL)**

As técnicas de *SRL* dizem respeito aos processos capazes de rotular os elementos de uma frase, de acordo com o seu papel semântico, isto é, são capazes de identificar quem é o sujeito, qual o objetivo, o resultado, entre outros, dependendo do domínio e do contexto em que estes se encontram inseridos [79].

Os papéis semânticos identificados pela técnica de SRL, correspondem à relação semântica estabelecida entre um verbo (ou outro elemento predicativo) e os seus respetivos argumentos e complementos, correspondendo, de grosso modo, à noção de *lead*: "Quem faz o Quê, a Quem, Como, Quando e Onde" [80].

Por exemplo, dada a frase, "O André vendeu um livro ao João", a técnica de SRL seria capaz reconhecer o verbo *"vender"* como representativo do predicado/verbo da frase, *"*André*"* como o agente ou sujeito, um *"*livro*"* como o objeto e *"*João*"* como o receptor. Ter esta capacidade de reconhecer as relações existentes numa frase, torna mais fácil a extração e manipulação o significado a ela adjacente.

- **Mecanismos de análise de questões**

Na construção de Chatbots, um passo fundamental, para o sistema inferir sobre a melhor abordagem a adotar na devolução de resposta, é a interpretação e a análise meticulosa da pergunta efetuada.

De um modo geral, a análise de uma questão inícia com o recebimento por parte do sistema, da questão enviada pelo utilizador, sob a forma de texto não estruturado , começando desta forma, por identificar de forma sintática e semântica, os elementos da questão.

A abordagem passa pela aplicação de várias regras de detecção em conjunto com classificadores, de forma a identificar os elementos na questão. Existem diversos tipos de elementos a encontrar numa questão, sendo cada um destes, importante para o pós-processamento da resposta a devolver pelo sistema.

Os elementos mais comuns são o fóco *(focus)* e os tipos de resposta lexical (*lexical answer types, LATs)*. Os *LAT* dizem respeito aos termos contidos numa dada questão, indicativos do tipo de entidade que se encontra a ser questionada, enquanto que o fóco, é como o nome indica, o fóco da questão, isto é, a parte da questão que é referência para a resposta a ser devolvida [81].

Por exemplo, dada a frase, "Quem é o presidente de Portugal?" , o termo "Quem" é neste caso, o **LAT**, indicativo de que o tipo de resposta será um nome de uma pessoa enquanto que a combinação de termos, "presidente de Portugal" , diz respeito ao **foco** da questão.

O desenvolvimento de ferramentas capazes de extraír este tipo de informação, elevam o desempenho dos sistemas conversacionais para outro patamar, permitindo um aumento na coerência frásica e estrutural das suas respostas.

3.2 MÉTODOS E FERRAMENTAS UTILIZADAS NA CONSTRUÇÃO DE UM CHATBOT

Esta sub-secção tem como objetivo apresentar uma descrição teórico-técnica das tecnologias e conceitos utilizados na presente dissertação, para a projecção e construção do tipo de Chatbot nela proposto.

Primeiramente é apresentado o conceito genérico de redes neuronais, servindo como base à explicação seguinte acerca de redes neuronais recorrentes, um dos tipos de redes neuronais existentes e o utilizado neste projeto. É também efetuada uma contextualização às principais técnicas de pré-processamento e de modelação de dados, utilizadas no desenvolvimento de sistemas conversacionais, como por exemplo as técnicas de *Word Embedding* e de *One-Hot Encoding*.

Para finalizar o capítulo, é apresentado e descrito um modelo SMT: o modelo *encoder-decoder*, também conhecido por *Seq2seq*, utilizado para a construção do motor conversacional adotado no nosso sistema.

3.2.1 REDES NEURONAIS ARTIFICIAIS - *ANN*

As redes neuronais artificiais *(ANN,* do inglês *Artificial Neural Networks)* , também designadas por sistemas conexionistas, são modelos simplificados do sistema nervoso central do ser humano, integrantes de uma estrutura extremamente interconectada de unidades computacionais, frequentemente designadas por neurónios ou nodos, aos quais se encontram associados uma certa capacidade de aprendizagem [82].

Uma ANN, assume-se então como um processador paralelo, composto por simples unidades de processamento, possuíndo uma propensão natural para armazenar conhecimento empírico e torná-lo disponível para o utilizador [82].

A sua atuação assemelha-se ao comportamento humano cerebral de duas maneiras [82]:

- → O conhecimento é adquirido a partir de um determinado ambiente, através de um processo de aprendizagem.
- → O conhecimento é armazenado nas conexões da rede, também designadas por ligações ou sinapses entre nodos.

Durante o processo de aprendizagem, realizado através um determinado algoritmo de aprendizagem, também denominado como algoritmo de treino, o peso (ou força) das conexões presentes na rede, é ajustada de forma a ser atingido um determinado objetivo ou um estado de conhecimento específico na rede [83].

A unidade de processamento chave para a operação funcional de uma ANN é o seu nodo. Apesar de existirem diversos tipos de nodos para implementação em redes ANN, o seu comportamento é usualmente, transversal entre si [82]:

Este funciona como um comparador, capaz de produzir uma saída, quando o efeito cumulativo das entradas, excede um determinado valor máximo. Se fizessemos *zoom* num nodo, o que seria encontrado é ilustrado na figura 3.2:

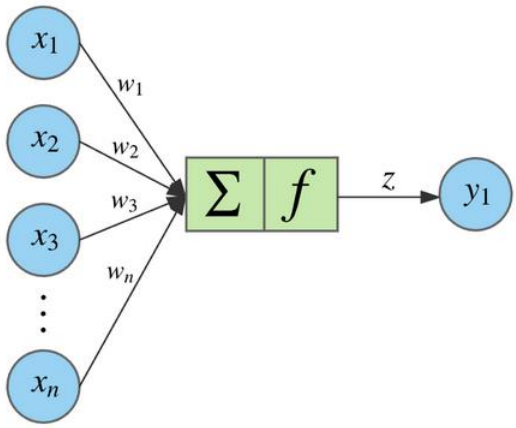


Figura 3.2 - Processamento existente num nó de uma Rede Neuronal Artificial.

Por observação da figura, verificamos que um neurónio presente na rede, é constituído pelos seguintes elementos :

→ Um **conjunto de conexões**, cada uma associada a um determinado peso (w_n), isto é, um número real ou binário capaz de possuir um efeito estimulatório (quando para valores positivos) ou inibitório (valores negativos) para a rede. Desta forma, o sinal ou estímulo (x_n) obtido como entrada da conexão, é multiplicado pelo peso (w_n) do nó em questão. Pode ainda existir uma conexão extra, não representada na figura, denominada de *bias*, cuja entrada toma o seu valor adicionado por uma unidade (+1), estabelecendo desta forma, uma certa tendência no processo computacional.

→ Um **integrador**, que possui como função reduzir os n argumetos de entrada a um único valor. Este integrador é representado de forma comum por uma função de adição ($\sum$), capaz de pesar todas as entradas numa combinação linear.

→ Uma **função de ativação (f)**, capaz de condicionar o sinal de saída, através da introdução de uma componentente não-linear nas rede, o que faz com que estas tenham a capacidade de aprender mais do que simples relações lineares entre as variáveis dependentes e independentes. Se não for aplicada uma função de ativação, então o sinal de saída seria uma simples função linear. Uma função linear é como sabido, um polinómio de grau 1, de fácil resolução mas bastante limitado em complexidade, possuindo pouca capacidade de aprendizagem aquando de mapeamentos complexos entre as diferentes conexões da rede.

A forma como estes nós se interligam para formar uma determinada estrutura, é denominada por arquitectura de rede, podendo ser, na sua forma mais comum, uma de três tipos: *Feed-forward* de uma só camada, *Feed-forward multicamada* ou *recorrentes* [82], [83].

Devido ao facto de as redes recorrentes serem uma derivação dos tipos de camada anteriores e por representarem o tipo de redes aplicada na componente prática desta dissertação, a sua explicação encontra-se numa sub-secção própria, deixando apenas expostos os conceitos relativos aos dois primeiros tipos de rede, na presente sub-secção.

As redes do tipo *feed-forward* de uma só camada (em inglês, *single layer feed-forward network*), são na sua forma mais simples, compostas por uma camada de entrada, cujos valores de saída são fixados externamente, e por uma camada de saída, estando desta forma, os nodos de entrada diretamente conectados aos nodos de saída, como ilustra a figura 3.3.

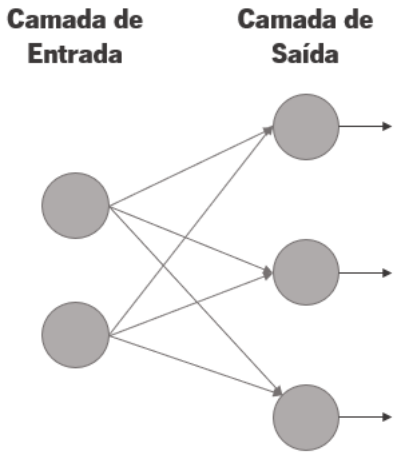


Figura 3.3 - Estrutura genérica de uma rede feed-forward de uma só camada.

O segundo tipo de arquitetura de rede dentro da classe *feed-forward*, distingue-se do primeiro na medida em que, para além de possuir as camadas de entrada e saída, estas não se encontram interligadas diretamente devido ao facto de conterem um novo tipo de camadas, às quais se dá o nome de camadas intermédias, constituídas por nodos intermédios [82], [84].

A adição de uma ou mais camadas intermédias, permite a elevação da capacidade da rede em modelar funções de maior complexidade (i.e, dados mais complexos como imagens, videos, etc.), nas quais o número de nodos à entrada é elevado. A contrapartida destas redes, é o facto da sua aprendizagem levar um maior espaço de tempo, dado o aumento de complexidade a elas inerente [82], [84].

A figura 3.4 ilustra a estrutura tradicional desde tipo de redes.

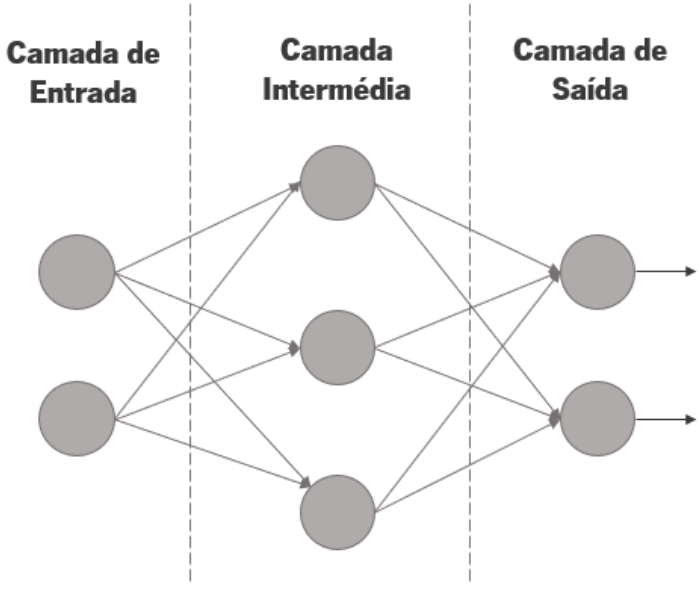


Figura 3.4 - Estrutura genérica de uma rede feed-forward multi-camada.

O sinal de entrada propaga-se ao longo da rede, camada a camada, sem a existência de ciclos. A não linearidade, a existência de nodos intermédios e o seu alto grau de conectividade entre nodos, tornam a arquitetura de *feed-forward* em multicamada, muito poderosa para o fenómeno de aprendizagem.

A propriedade mais importante das ANN apresentadas talvez resida na sua capacidade de aprendizagem a partir do seu ambiente, isto é, do conjunto de dados representado nas diferentes entradas, a elas fornecido. Esta acção de aprendizagem é realizada a partir de um conjunto de regras representadas sob a forma de um algoritmo, sendo esta acção também denominada de **treino**, distinguindo-se os diversos algoritmos pelas **regras** e **tipos** de aprendizagem, nele implementadas [82], [85].

Fundamentalmente, existem três grandes tipos de aprendizagem [82], [86],[87]:

→ **Supervisionada** (em inglês, *Supervised Learning)*

Este tipo de aprendizagem pode ser interpretado como a realização de uma dada tarefa sobre supervisão de alguma entidade, responsável por julgar se esta está a ser realizada de forma correta. Trata-se de um paradigma que envolve um "professor", isto é, são fornecidas as respostas correctas a dele, sendo esta capaz de aprender a partir de um conjunto de padrões, onde cada padrão é chamado de caso de treino, constituído por um vector de entrada e um vector de saída/resposta. Durante o processo de aprendizagem, é realizada uma comparação entre o valor desejado com o valor de saída da rede, originando um dado valor de **erro**. Este erro, resultante das comparações efetuadas é utilizado para ajustar os pesos de cada nodo da rede, de forma a que o valor deste seja constantemente reduzido, realizando-se para tal, constantes iterações ao algoritmo de treino. Este é o tipo de aprendizagem mais comum para tarefas de aprendizagem, via redes neuronais, sendo inclusive, o tipo de aprendizagem adoptada na presente dissertação.

→ **Reforço** (em inglês, *Reinforcement Learning)*

A aprendizagem do tipo *Reinforcement*, assume também a presença de uma entidade externa, neste contexto novamente tomada como um "professor", mas contrariamente ao sucedido no tipo de aprendizagem supervisionada, a resposta correta não é apresentada para a rede, fornecendo apenas pistas ou indicações se a resposta ou saída atual desta, é ou não correta. A rede utiliza então esta informação fornecida para melhorar a sua eficácia, reforçando os pesos em determinados nodos quando a resposta devolvida pela rede é a correta e penalizando-os quando acontece o contrário.

Este tipo de aprendizagem é bastante utilizado no desenvolvimento de videojogos, na medida em que são fornecidas várias dicas e recompensas à medida que se vai avançando neste, melhorando o seu

desempenho no próximo nível. Sem este *feedback*, os utilizadores iriam actuar de forma aleatória na esperança de passar para o nível seguinte.

→ **Não Supervisionada** (em inglês, *Unsupervised Learning)*

Este último tipo de aprendizagem em nada se relaciona com os dois tipos anteriormente descritos. Neste caso não existe nenhuma entidade responsável pelo fornecimento de indicações acerca da resposta correta, baseando-se a sua aprendizagem na descoberta de padrões e características presentes nos dados de entrada. Dependendo do problema a realizar sobre este tipo de aprendizagem, o algoritmo pode organizar os dados em diferentes maneiras, como por exemplo o recurso a técnicas de *Clustering* ou de detecção de anomalias.

Na distinção de algoritmos de aprendizagem, para além do seu tipo, é também comum efetuar-se uma diferenciação quanto ao tipo de regras nestes implementado. As regras de aprendizagem dividem-se em cinco tipos básicos: *Hebbian,* competitiva, estocástica, baseada em memória e no tipo mais conhecido e utilizado nos algoritmos desta dissertação, gradiente descendente [82].

Como referido anteriormente, na aprendizagem do tipo supervisionada, pretende-se reduzir o erro entre o valor a alcançar e o valor de saída da rede. Desta forma, o erro atua como um mecanismo de controlo, onde os seus vários ajustamentos resultam em melhores respostas. O objetivo consiste na minimização de uma função de custo, $\boldsymbol{\xi}$, definida em termos do sinal de erro e^P [82], [88]. Esta regra de aprendizagem, também conhecida como a regra delta ou *Widrow-Hoff,* baseia-se na resolução da seguinte equação:

$$\Delta w = \eta \nabla \xi$$

onde η representa uma constante positiva, denominada taxa de aprendizagem e $\nabla \xi$ representa o gradiente da função de custo utilizada. Esta remete para um tipo de aprendizagem amplamente utilizado e conhecido, associada ao popular algoritmo de *Back-Propagation*, o mais conhecido algoritmo dentro do conjunto de algortimos supervisionados [82], [89].

O seu sucesso baseia-se na sua representação como um método eficiente de computação para o treino de Redes Neuronais, procurando o mínimo da função de erro no espaço temporal de procura dos pesos associados ás diversas conexões, baseiando-se em métodos de gradiente descendente. A combinação dos pesos que minimiza a função de erro é considerada a solução para o problema de aprendizagem. Dado que este método exige o cálculo do gradiente, é obrigatório que a função de erro seja continua e diferenciável, algo que se sucede aquando da utilização de funções de ativação diferenciáveis [82], [89].

3.2.2 REDES NEURONAIS RECORRENTES – *RNN*

O último grande tipo de arquitetura de redes neuronais artificiais, não explicado na secção anterior, possui o nome de Redes Neuronais Recorrentes. A recorrência ocorre neste tipo de redes devido ao facto de uma saída de um elemento na rede, influenciar de alguma maneira, a entrada para esse mesmo elemento, criando-se um género de circuito fechado a partir da disposição de como os nodos se encontram conectados [82], [85].

A formulação de conexões ciclícas introduz na rede uma componente não-linear, de natureza espacial e/ou temporal, que pode ser vista como um tipo de memória associativa e/ou temporal, permitindo desta forma o processamento de dados do tipo sequencial, como por exemplo ficheiros de texto ou audio [85].

Se o tipo de dados a ser tratado e processado, variar numa escala temporal, um único ponto de dados isolado, não é inteiramente útil devido ao facto de carecer de atributos importantes, como por exemplo se ocorreu uma mudança no conjunto de dados, se este aumentou ou diminuiu em termos dimensionais, para além do não conhecimento do contexto em que o ponto de dados se encontra inserido [90].

Por exemplo, no caso da dissertação apresentada, dado que a tarefa a realizar envolve o processamento de dados textuais com o objetivo de simular uma conversa humana, aquando da construção frásica, torna-se necessário saber quais as palavras já formuladas na frase a devolver, devido ao facto do **contexto** ser crucial, tarefa que se revela impossivel de realizar através do uso de outras de redes *Feed-Forward* [90].

A figura 3.5 ilustra a diferença entre o fluxo de informação e processamento nos nodos presentes numa rede RNN e numa rede do tipo *Feed-Forward.*

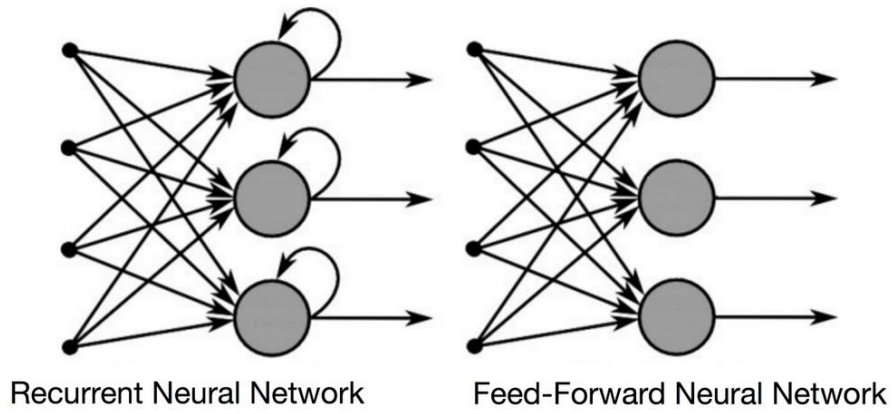


Figura 3.5 - Diferença entre uma RNN e uma rede do tipo Feed-Forward. (Retirado de [86]).

Numa rede neuronal recorrente, a informação circula num *loop* interno da rede, possuíndo a capacidade de manter memória interna com um determinado mecanismo de *feedback* a si associado, permitindo-lhe assim, suportar e compreender a variação do conjunto de dados num dado espaço de tempo.

Quando a rede toma uma decisão num dado nó, tem em consideração a entrada desse nodo, bem como toda a informação e aprendizagem efetuada pela rede até aquele instante.

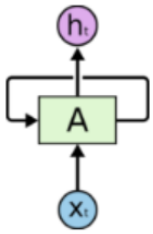


Figura 3.6 - Estrutura do mecanismo presente numa RNN (Retirado de [87]).

A figura 3.6 demonstra uma porção de uma dada rede neuronal A, que para um dado *input* x_i devolve um determinado *output* representado por h_i. Um mecanismo de *loop* permite que a informação seja passada de uma dada iteração na aprendizagem da rede para a iteração seguinte [91].

Apesar da introdução de *looping* na rede, as RNN não são assim tão diferentes quando comparadas com as redes neuronais tradicionais:

Uma RNN pode ser vista como um conjunto constituído por múltiplas cópias da mesma rede, a que se dá o nome de células, cada uma transmitindo uma mensagem ao seu sucessor na rede [91].

Desdobrando a rede apresentada na imagem 3.6, o mecanismo de *looping* pode ser visto da maneira representada na figura 3.7.

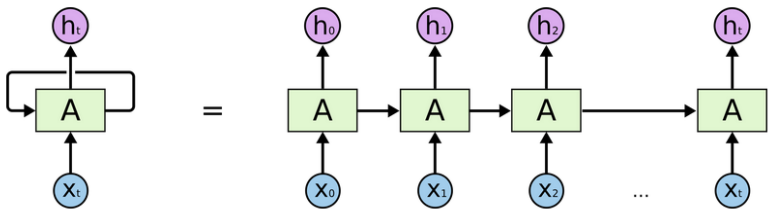


Figura 3.7 - Tipologia de uma rede neuronal recorrente (Retirado de [86]).

Esta natureza, de processamento em cadeia, demonstra que as RNN se encontram intimamente relacionadas com sequências e listas, e daí serem a arquitetura natural de redes neuronais a utilizar para este tipo de dados. Este tipo de redes tem tido bastante sucesso na realização de tarefas como modelação de linguagem, reconhecimento de discurso, traduções, entre outras tarefas que englobem um tipo de dados sequencial.

Essencial para esse sucesso é a utilização de dois tipos específicos de redes neuronais recorrentes: as redes *LSTM* e as redes *Gru-RNN*. São estes dois tipos de redes utilizadas na componente prática desta dissertação, devido ao facto de possuirem um melhor desempenho e por resolverem alguns problemas presentes na versão *standard*, como é o caso do problema de dependências a longo-prazo [92]:

O problema de depências a longo-prazo remete para uma adversidade comum na estrutura original das RNN:

Quando o desfasamento entre processamento entre os nodos da rede é elevado, as RNN revelam alguma dificuldade em tratar estas dependências a longo-prazo, isto é, não conseguem transmitir a informação localizada num espaço de tempo muito anterior na rede, para o nodo onde esta é atualmente necessária [91], [92].

Por exemplo, no caso da tarefa relativa ao processamento textual, perante a frase "as nuvens encontram-se no céu", se o objetivo atribuído à rede fosse prever a última palavra presente na frase, não se revela necessária mais nenhuma contextualização ou informação transmitida para esta, dado que é óbvio e facilmente por ela inferido, que a palavra a devolver seria o termo "céu".

No entanto existem casos em que o fornecimento de mais contexto e informação se revela necessário para a rede conseguir realizar a sua aprendizagem, e.g. supondo que as seguintes frases foram retiradas de um excerto de texto, em que entre elas existe um grande número de frases e informação:

"Eu nasci e cresci em Portugal"

(...)

"Eu falo português".

Por análise da última frase, a rede é capaz de deduzir que a próxima palavra nela presente seria referente a uma dada linguagem, no entanto para esta inferir qual é, revela-se necessário o conhecimento textual referente a "Portugal", que se encontra localizado mais atrás no espaço temporal de aprendizagem atual, podendo o desfasamento entre esta informação considerada relevante para o contexto, ser demasiado grande para a RNN interligar a informação, inibindo esta de efetuar corretamente a tarefa de previsão. O problema de dependências a longo-prazo é demonstrado na figura 3.8.

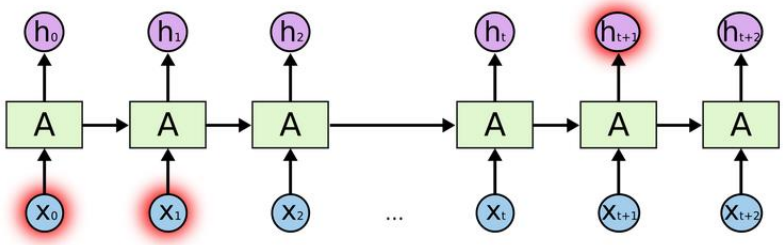


Figura 3.8 - Problema de depêndencias a longo-prazo (Retirado de [87].

→ **Redes LSTM – Long Short Term Memory Networks**

As redes LSTM são um tipo específico de redes neuronais recorrentes, capazes de efetuar uma aprendizagem a *longo-prazo*, isto é, são capazes de se lembrar dos *inputs* sobre um longo período de tempo, eliminado o problema de dependências a longo prazo. Isto deriva do facto das redes LSTM armazenarem a informação sob uma forma específica de memória, semelhante à memória de um computador, dado que este tipo de redes pode ler, escrever e apagar informação da sua memória [91], [93].

Uma rede LSTM é tipicamente constitúida por diferentes blocos de memória a que se dá o nome de células (representado pelos retângulos na figura 3.9).

Cada célula na rede recebe num dado momento dois estados/entradas: o estado da célula atual e o seu estado intermédio. A figura 3.9 ilustra a estrutura típica de uma rede LSTM, sendo cada célula nela presente representada por A.

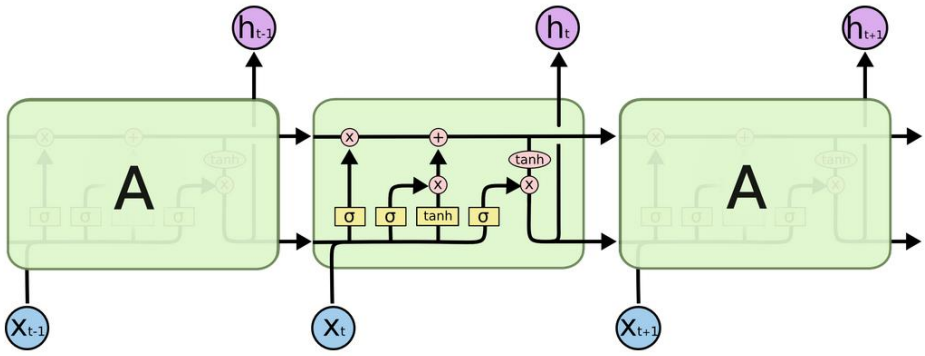


Figura 3.9 - Exemplo de uma estrutura de uma rede LSTM (Retirado de [87]).

Esta memória presente numa rede LSTM pode ser vista como uma célula constituída por *portas lógicas*, responsáveis por decidir se a informação a receber naquele momento, é para armazenar ou para remover, baseando-se para tal no peso ou preponderância acoplado à informação recebida como entrada naquele momento [91], [93].

Cada célula presente numa rede LSTM é constituída por três portas lógicas: *input gate*, *forget gate* e uma *output gate*. Estas portas são responsáveis por determinar se devem ou não, deixar entrar uma nova entrada numa dada célula (responsabilidade da *input gate)*, remover a informação devido a esta não ser importante *(forget gate)* ou permitir à nova entrada influenciar e possuir impacto na saída a devolver naquele momento *(output gate)* [91], [93]. Cada um destes mecanismo tem uma função e lógica distinta entre si, explicadas de seguida.

A primeira porta lógica a ser abordada tem como nome *forget gate*. Tomando como exemplo um problema de previsão num dado excerto de texto, assumindo que uma dada rede LSTM recebe a seguinte frase como entrada:

"O André é boa pessoa. O João por outro lado, é mau."

Assim que o ponto final é encontrado na primeira frase, a *forget gate*, tem a capacidade de se aperceber, que poderá haver uma mudança de contexto na frase seguinte. Como resultado deste facto, o *sujeito* da primeira frase é *esquecido*, ficando o lugar para o sujeito vago na célula. Quando a rede, recebe a informação acerca da segunda frase, a posição de sujeito é alocada para o "João" [93].

Este processo de esquecimento do sujeito aquando da mudança de contexto frásica, é responsabilidade da *forget gate*, sendo a sua estrutura apresentada na figura 3.10.

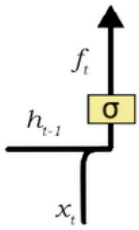


Figura 3.10 - Estrutura comum de uma output gate, presente numa dada célula de rede LSTM (Retirado de [88]).

A informação que se revela, através da sua preponderância, já não ser necessária para a rede LSTM entender o contexto, ou simplesmente por ser desnecessária para a sua aprendizagem, é removida através da multiplicação de um filtro (normalmente sob a forma de uma função sigmoide) na rede, necessária para otimizar o desempenho desta [93].

Esta porta como demonstrado na figura 3.10 recebe dois *inputs*:

- h_{t-1} – Referente ao estado intermédio, isto é, o estado resultante da célula anterior.
- x_t – Referente à entrada da célula naquele momento.

O segundo tipo de portas adotadas numa rede LSTM possuem o nome de *input gates*. Continuando a utilizar o processamento de texto como tarefa, para efeitos exemplificativos, supondo que, num dado momento, a rede LSTM se encontra a analisar a seguinte frase: "O Rui sabe nadar. Ele disse-me por telefone que serviu no exército por 4 longos anos!".

Por observação e análise, deduz-se que a informação importante a reter sobre a frase acima apresentada, é que o "Rui" sabe nadar e que frequentou o exército durante quatro anos.

Esta informação pode ser adicionada ao estado da célula naquele preciso momento, pois revela-se como informação importante a reter, para a aprendizagem futura da rede [93]. No entanto, o facto de esta informação ter sido transmitida por telefone pode ser considerado um acontecimento menos preponderante, podendo como tal ser ignorado por esta. Este processo de adicionar nova informação pode ser efetuado através da *input gate*. A sua estrutura pode ser observada na imagem 3.11.

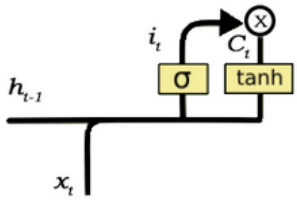


Figura 3.11 - Estrutura comum de uma input gate, presente numa dada célula de uma rede LSTM (Retirado de [88]).

Esta adição de informação na rede é efetuada através de um processo, que consiste em três passos distintos, como observável na imagem 3.11:

Primeiro, é necessário efetuar um controlo e um acerto dos valores que necessitam de ser adicionados à celula, através da aplicação de uma função sigmoide. Este passo é muito semelhante ao que acontece na *output gate,* sendo que esta actua como filtro da informação recebida como entrada, nomeadamente h_{t-1} e x_t [90], [91].

De seguida, é criado um vetor que contém todos os valores passíveis de serem adicionados (derivados de h_{t-1} e x_t) ao estado da célula naquele momento. Este passo é realizado através da aplicação da função tangente hiperbólica, representada na figura pela sua nomenclatura usual, $tanh$.

O passo final consiste na multiplicação dos resultados extraídos dos dois passos anteriores (valor da aplicação do filtro regulador no passo 1 e do vetor resultante do passo 2), adicionando depois a informação ao estado da célula através de uma operação básica de adição. Uma vez realizado o terceiro passo, garante-se que apenas informação importante e não redundante é adicionada à célula [91], [93].

A última porta lógica contida numa rede LSTM tem o nome de *output gate*. Nem toda a informação que percorre e está contida no estado da célula atual, é adequada para ser devolvida como saída num dado momento. Tendo como exemplo a seguinte frase:

"O Rui combateu sozinho o inimigo, sacrificando-se pelo seu país. Pelas suas contribuição, podemos dizer, bravo ___".

Nesta frase, podem existir inúmeras opções para preencher o espaço vazio. A rede sabe que a última entrada presente na célula, "bravo", é um adjetivo que é utilizado para descrever um nome, e então,

qualquer palavra que o suceda é identificada na rede como tendo forte tendência a ser um nome [93]. Consequentemente, o termo "Rui", seria neste caso, uma saída válida devolvida pela rede.

A tarefa de selecionar informação útil do estado atual da célula e devolvê-la como saída da rede, é efetuada através do *output gate*. A sua estrutura é demonstrada na figura 3.12:

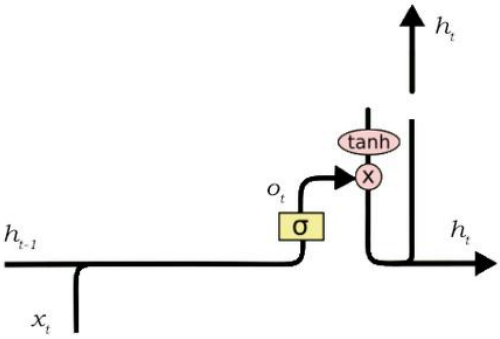


Figura 3.12 - Estrutura comum de um output gate, numa dada célula de uma rede LSTM (Retirado de [88]).

Semelhantemente ao que ocorre numa *input gate*, o funcionamento de uma *output gate*, pode ser dividido em três passos distintos [91], [93]:

Primeiramente ocorre a criação de um vetor por aplicação da função $tanh$ ao estado da célula, escalando desta forma os valores desta, entre -1 e +1.

De seguida é aplicada novamente, como efetuado nas outras duas portas apresentadas, a função sigmoide, para aplicação de um filtro com o objetivo de efetuar o acerto dos valores que advêm de h_{t-1} e x_t recebidos como *input*. O passo final consiste na multiplicação do valor resultante da aplicação do filtro no passo 2, pelo vetor criado no passo 1, enviando o resultado como *output* e também como estado intermédio para a célula seguinte, sendo este resultado representado por h_{t-1} na célula seguinte.

→ Redes Gru-RNN – Gated Recurrent Unit

As redes Gru-RNN representam uma variação das redes LSTM apresentadas anteriormente, introduzidas por Cho, et al [94]. A sua estrutura é bastante semelhante à apresentada numa rede LSTM, com a diferença que neste tipo de redes, existem duas portas lógicas e não três: a *output gate* e a *input gate*, são combinadas numa única porta lógica, a que se dá o nome de *update gate* [91], [95]. Para além da *update gate*, uma rede *Gru* possui ainda uma *reset gate*.

Basicamente, a *reset gate* determina como combinar a entrada numa dada célula na rede, com a memória que advém da célula anterior, ficando a cargo da *update gate*, definir qual a memória relativa à informação a reter, no estado atual da célula, para formular a sua saída, não existindo desta forma, nenhum

estado persistente na célula, como ocorre nas células LSTM [91], [95]. A estrutura presente numa célula de uma rede do tipo Gru-RNN é representada na figura 3.13.

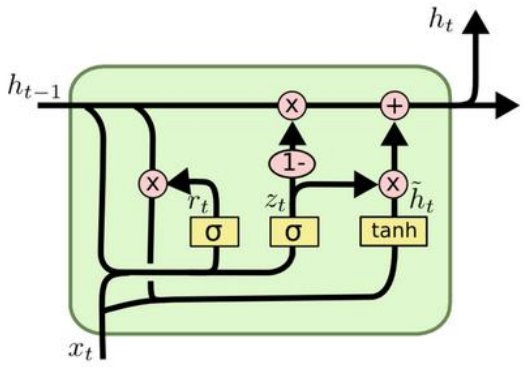


Figura 3.13 - Estrutura de uma célula constituinte da rede Gru-RNN (Retirado de [87])

Tanto as redes LSTM como as redes Gru são capazes de capturar e reter informação a curto e a longo prazo, isto é, a informação não é perdida à medida que o espaço temporal avança na rede, com os sucessivos treinos que ocorrem nesta.

No entanto, devido ao facto das redes Gru envolverem menos parâmetros, derivado do facto de possuirem menos portas lógicas, estas são muitas das vezes utilizadas como alternativa às redes LSTM por possuirem melhores resultados, serem mais rápidas de treinar e por utilizarem menos memória aquando do seu processamento.

3.2.3 MÉTODOS DE REPRESENTAÇÃO DE DADOS CATEGÓRICOS

As redes neuronais artificiais apresentadas anteriormente apenas conseguem processar informação de entrada e de saída, se estas possuírem um valor de tamanho fixo, armazenado sob a forma de um vetor.

É necessário portanto, uma certa transformação, usualmente conhecida por *encoding* , que se baseia no pré-processamento de um determinado conjunto de dados categóricos ou nominais, tendo como objetivo a representação destes, sob forma a que possam ser compreendidos e processados pela rede [85]. Como os dados categóricos não são representados sob forma numérica, o seu número possível de valores no conjunto é limitado, tornando possível efetuar a diferenciação entre cada um dos valores a ele pertencente [96].

Tendo como exemplo os seguintes conjuntos de dados categóricos:

- Uma variável animal, com os valores “cão” , “gato ” e “águia”.
- Uma variável cor, com os valores “vermelho”,”verde” e ”preto” a si associada.
- Uma variável classificações, com os valores “primeiro”,”segundo” e “terceiro”.

Cada variável acima apresentada representa uma categoria distinta, e podemos por observação, inferir que algumas destas podem possuir alguma relação entre elas, sendo a sua ordenação em certos casos, um factor relevante a ter em consideração, como no demonstrado na variável classificação apresentada acima [97], [98].

Devido à existência de casos em que o agrupamento de dados relativos a uma determinada categoria, poder ser ou não influenciado pela sua ordenação, foram desenvolvidas duas técnicas distintas para tratar cada um dos casos, sendo estas chamadas de *Label Encoding* e *One Hot Encoding*.

A técnica de *Label Encoding* é muito simples, baseando-se apenas na atribuição de um número inteiro único a cada valor presente num conjunto de dados categóricos. Por exemplo, supondo que o conjunto de dados a utilizar na rede, seria o conjunto de dados relativo a animais, apresentado anteriormente, a técnica de *Label Encoding* seria responsável por efetuar o mapeamento para os diferentes valores a ele pertencente, como por exemplo: [“águia” = 1], [“cão”=2] e [“gato”=3] [98].

No entanto, este mapeamento numérico incremental, levanta um problema para uma grande parte de algoritmos de ML e de redes neuronais, devido ao facto de os valores numéricos, por si só, possuirem uma relação de ordenação natural, podendo ser entendidos pela rede como mais relevantes, atribuíndo esta mais peso aos nodos representados por variáveis associadas a números mais elevados [99].

Desta forma, o nodo de entrada relativo ao termo “gato” seria entendido para a rede, como mais preponderante quando comparado com o termo “águia”, situação que levaria a uma má aprendizagem e na inferência de conclusões falaciosas por parte da rede.

Este tipo de *encoding* seria aplicável por exemplo ao caso do conjunto de dados referentes a classificações, onde de facto a ordenação interessa, para atribuír preponderância aos seus diferentes valores. Para ultrapassar o problema exposto acima, é utilizado um diferente tipo de *encoding*, que possui o nome de *one-hot encoding*.

Nesta técnica, o processo de atribuição numérica incremental aos diferentes valores de um conjunto de dados é substituido por uma variável binária (daí o nome *one-hot*), única para cada valor nele presente [98].

A figura 3.14 ilustra as diferenças na transformação presente num certo conjunto de dados categóricos, utilizando cada uma das técnicas apresentadas.

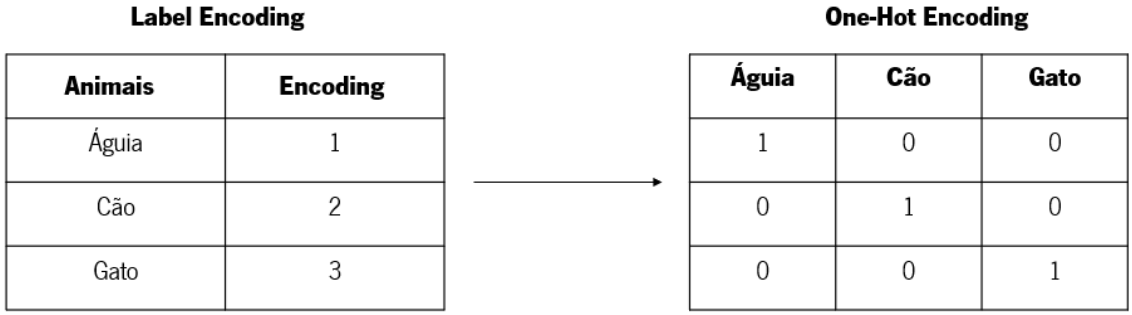


Figura 3.14 - Diferenças na aplicação das técnicas de Label e One-Hot Encoding.

Através da transformação dos dados categóricos em vectores do tipo *one-hot*, a rede adquire a capacidade de identificar corretamente os dados a serem processados em cada nodo num dado momento, não atribuíndo a este, nenhuma preponderância adicional para a sua aprendizagem.

Na tarefa de processamento textual explorada nesta dissertação, as frases fornecidas necessitam, para efetuar a aprendizagem na rede, de ser representadas sob vetores de tamanho fixo, sem qualquer ordenação em termos de relevância entre os termos nela presentes. Devido a este facto, no desenvolvimento de sistemas conversacionais, é comum a utilização de técnicas de *encoding* do tipo *one-hot*.

3.2.4 WORD EMBEDDING

Na sub-secção anterior foram apresentadas duas técnicas para a representação de dados categóricos sob a forma de vectores, com vista o seu processamento para a entrada e saída de informação na rede. Na presente sub-secção, é apresentado uma nova técnica para a representação de informação categórica na rede, chamada *Word Embedding*, remetente para um tipo de representação de dados sob um novo tipo de vectores, chamados *word vectors*.

Word Vectors são de uma forma genérica, vectores constituídos por pesos, capazes de representar o significado de uma dada palavra num conjunto de dados textuais [100]. A representação de todos os termos pertencentes a um conjunto de dados, sob a forma de vetores do tipo *one-hot*, não permite o estabelecimento de medidas de similaridade entre estes, dado que a dimensão do seu vector será sempre igual ao número de termos distintos a serem processados pela rede, sendo cada vector declarado de maneira diferencial e sem qualquer tipo de relação perante os demais [100].

Supondo que o nosso vocabulário é constituído pelos termos [‘Rei’,’Rainha’,’Princesa’,’Mulher’], a representação do termo “Rainha”, sob o *encoding* do tipo *one-hot*, seria efetuada da maneira ilustrada pela figura 3.15:

Mulher	Princesa	Rei	Rainha
0	0	0	1

Figura 3.15 - Representação do termo "Rainha", presente no vocabulário de entrada, sob um vetor do tipo one-hot

O problema na utilização deste tipo de representação é o facto de, exceptuando testes de igualdade, não existir qualquer comparação significativa que possa ser efetuada entre os diferentes vetores.

Os *Word Vectors*, vieram ultrapassar este problema, sendo estes adoptivos de um tipo de representação distribuída sob a palavra a ser transformada. Neste, cada termo é representado por uma distribuição do seu peso contra o vocabulário de entrada no qual este se encontra inserido [100]. Então, invés de ser realizado um mapeamento um-para-um, entre um elemento do vetor e o seu termo, a representação deste é realizada através de uma distribuição sob todos os elementos no vetor, contribuíndo cada elemento para a definição de vários termos [100].

Tomando como exemplo, um conjunto de dimensões contextuais, pré-declaradas (obviamente não existem dimensões contextuais pré-declaradas na rede) para efeitos demonstrativos, os vetores do vocabulário anterior seriam formulados da maneira ilustrada na figura 3.16.

		Mulher	Princesa	Rei	Rainha
Realeza →		0.05	0.98	0.99	0.99
Masculino →		0.01	0.02	0.99	0.05
Feminino →		0.99	0.94	0.05	0.93
Velhice →		0.5	0.1	0.7	0.6
(....)	(...)	(...)	(...)	(...)	(...)

Figura 3.16 - Representação de vocabulário sob a forma de word vectors.

Em suma, um *Word Vector* corresponde a uma linha constituída por N números reais, onde cada ponto captura a dimensão contextual do termo, e onde termos semânticamente similares possuem vectores também semelhantes. Isto significa que palavras como “Rei” e “Rainha”, devido à semelhança do seu significado, devem possuir *Word Vectors* idênticos ao termo “Realeza”, estando estes localizados proximamente no espaço vectorial do vocabulário [101].

Devido ao facto dos termos serem representados como vectores, permitem a aplicação de operações aritméticas onde é permitido subtrair e adicionar significados de diferentes termos, permitindo desta forma a extracção de novo contexto, como por exemplo [101]:

$$rei - homem + mulher = rainha$$

3.2.5 MODELO SEQUENCE TO SEQUENCE

Um modelo que se encontra a ganhar cada vez mais popularidade para efetuar a aprendizagem e previsão textual é o modelo *sequence-to-sequence* apresentado por [86], também chamado de modelo *encoder-decoder*.

Este modelo é utilizado para a execução de diversas tarefas complexas como *Machine Translation*, auto-descrição de imagens, modelação e geração conversacional, sendo portanto amplamente utilizado na construção de Chatbots.

O formato mais simples e inicial do modelo, é baseado em duas redes RNN: uma rede a que se dá o nome de **encoder** e outra a que se dá o nome de **decoder**, podendo estas ser de dois tipos de RNN distintos: redes LSTM ou redes Gru-RNN [86]. Uma das redes tem como função efetuar o *encoding* dos dados de entrada na rede, numa representação vetorial capaz de capturar o significado e informação relevante a eles associado [86].

A outra rede é responsável por receber como entrada, o vetor formulado na primeira rede, e usá-lo para produzir a frase devolvida como saída da rede. Este processo pode ser observado na figura 3.17, na qual cada retângulo diz respeito a uma célula de uma rede RNN.

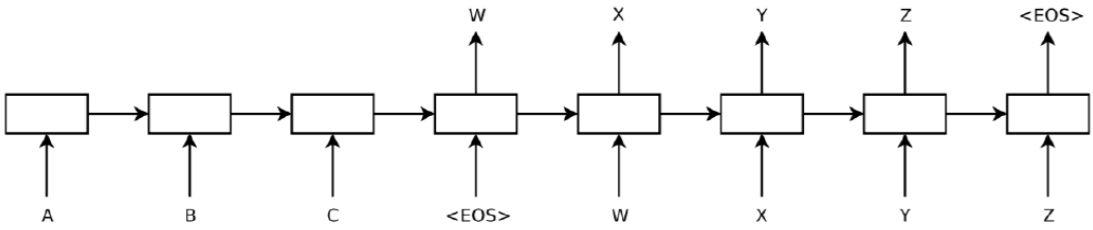


Figura 3.17 - Um modelo seq2seq genérico, onde (A;B;C) é a frase de input, <EOS> é um símbolo utilizado para delinear o final da frase e (W,X,Y,Z) é a frase devolvida como saída pela rede (Retirado de [86]).

Na figura, o *encoder* processa a entrada recebida, *token* a *token*, até atingir um *token* especial, representado na imagem por <EOS>, relativo ao fim da informação recebida como entrada [102]. Com base no contexto gerado pelo *encoder*, o *decoder* gera a informação a devolver como saída da rede, *token* a *token,* até atingir novamente, um *token* de paragem [102].

A partir da primeira entrada na rede *decoder* (na figura representada por W) a rede gera uma distribuição probabilistica sob todos os *tokens* já conhecidos. A partir dessa distribuição, um *token* é escolhido (na figura representado por X) e realimentado na rede *decoder* para a geração de uma nova distribuição probabilística para a inferência do próximo *token*, e assim sucessivamente. São os *tokens* escolhidos, representados por X, Y e Z que formam a frase a ser devolvida.

Existem duas maneiras para efetuar a escolha por parte da rede do *token* de sáida em cada célula: a selecção do *token* mais provável em cada espaço temporal, ao qual se dá o nome de abordagem gananciosa (em inglês, *greedy manner*) ou a análise dos x *tokens* mais prováveis, sendo esta segunda abordagem chamada de *beam search*, permitindo encontrar a mais provável sequência de *tokens* a serem devolvidos.

O objetivo do modelo *encoder-decoder* pode ser resumido à estimação da probabilidade condicional $p(y_1, \dots, y_{N'} | x_1, \dots, x_N)$, onde $x_1, \dots, x_N$ diz respeito à frase de recebida como entrada na rede e $y_1, \dots, y_{N'}$ corresponde à frase devolvida pela rede, como saída [102]. Baseando-se na figura 3.17, esta probabilidade seria escrita como $p(X, Y, Z | A, B, C)$. Dado que duas RNNs distintas, são utilizadas para as processamento das frases recebidas como *input* e devolvidas como *output*, o comprimento destas, designado na equação acima por N e N', pode também ser diferente.

CAPÍTULO 4
YEC – CHATBOT PARA APOIO CLÍNICO

No presente capítulo é descrita a elaboração e a divisão do problema apresentado, bem como a forma como este foi explorado segundo as tecnologias anteriormente mencionadas.

O sistema idealizado é apresentado nas diferentes secções seguintes, componente a componente, de acordo com a lógica funcional de cada um destes. O capítulo termina com a apresentação da arquitetura global de desenvolvimento do sistema, explicíta da maneira de como os diferentes componentes descritos, quando interligados, resultam no desenvolvimento do sistema proposto.

4.1 PROJECÇÃO E PERSPECTIVA GERAL DO SISTEMA

Como já referido, a presente dissertação consiste na projecção e no desenvolvimento de um Chatbot para apoio clínico, sendo o sistema a desenvolver, de uma maneira específica, designado a prestar auxílio e fornecer companhia a um determinado grupo-alvo de utilizadores, nomeadamente idosos e pessoas que padeçam de disturbíos psicológicos, como autismo ou depressões, permitindo ao sistema funcionar ora como "companheiro" ora como "psicólogo" virtual.

O sistema é arquitectado no presente capítulo de forma a possuír a capacidade de efetuar a distinção acima referida, para além de garantir o estabelecimento de um certo nível de proximidade e confiança entre si e os seus diferentes utilizadores, característica que se revela muito importante para o sucesso e aceitação do sistema por parte destes.

Esta confiança resulta da adopção de uma abordagem distinta e inovadora, quando comparada com a maioria dos sistemas conversacionais desenvolvidos e apresentados: A incorporação de um mecanismo que permita ao sistema efetuar um tratamento único e **diferenciado** em cada conversa. Contrariamente ao que se sucede na maior parte dos sistemas do género, o modelo que permite ao sistema gerar a resposta final, varia de utilizador para utilizador, permitindo desta forma a adição de um certo tipo de memória virtual.

Apesar de ter sido identificado um certo grupo inicial de utilizadores alvo para o sistema a desenvolver, o domínio conversacional deste será aberto, não havendo um tópico mandatório aquando da realização das conversas. Para além de possuir domínio aberto, o sistema é caracterizado pela incorporação de um modelo generativo para a formulação das suas respostas, conjugando técnicas de processamento de linguagem natural e de *Deep Learning* apresentadas no capítulos anteriores. Quanto ao nome do Chatbot, este, como apresentado no título do capítulo, possui o nome de **YEC**– sigla inglesa para "*Your everyday companion"*, em português "A tua companhia diária"*.* Do ponto de vista do utilizador, com um elevado nível de abstração, o funcionamento do sistema é apresentado na figura 4.1.

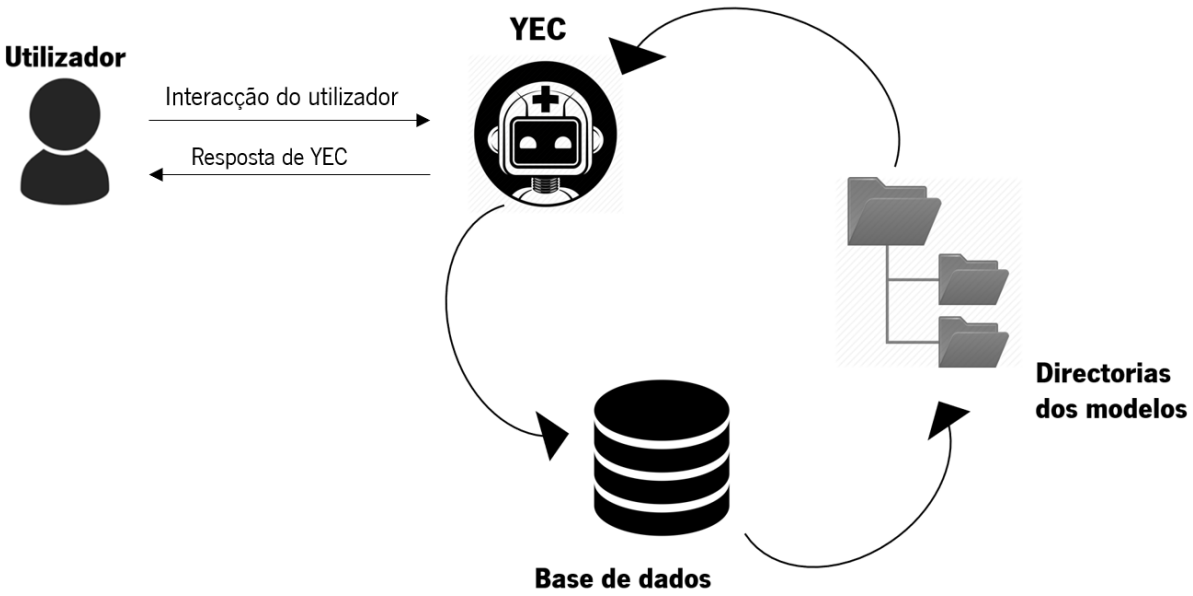


Figura 4.1 - Fluxo de conversa existente em YEC Bot.

A entidade relativa ao utilizador é sempre autónoma, não sendo objeto de qualquer desenvolvimento nem tratamento especial. Todos os utilizadores são tratados de igual forma pelo sistema numa fase inicial, possuíndo este a capacidade de se moldar ao utilizador, à medida que efetua conversas contínuas com este.

O modelo implementado no sistema YEC, é responsável pela geração da resposta devolvida pelo sistema de acordo com a entrada fornecida pelo utilizador, sendo naturalmente o núcleo de todo o funcionamento, envolvendo como tal diferentes passos durante o seu desenvolvimento.

Para uma leitura mais fácil e compreensiva de todos os componentes e da maneira de como esses componentes interligados, resultam na construção de YEC, o desenvolvimento do sistema pode ser visto como a realização dos seguintes passos:

1. Extracção e tratamento inicial de dados.
2. Armazenamento do histórico de conversas e do treino dos modelos, diferenciados por utilizador.
3. Desenvolvimento do motor conversacional com capacidade de geração automática de resposta – modelação.
4. Treino do modelo, através da realimentação dos dados conversacionais.
5. Disponibilização/exposição do modelo.
6. Desenvolvimento da aplicação-cliente.

4.2 MÉTODOS E COMPONENTES PARA DESENVOLVIMENTO DO SISTEMA

Na presente secção, são apresentados os diferentes métodos e componentes, que constituem o desenvolvimento do sistema YEC de acordo com os passos anteriormente mencionados. As diferentes tarefas a realizar, que englobam e abrangem esses métodos e componentes são individualizadas e descritas nas seguintes sub-secções.

4.2.1 EXTRACÇÃO INÍCIAL DE DADOS E ARMAZENAMENTO DE INFORMAÇÃO

Como já referido, uma das principais características do sistema é a sua capacidade de efetuar um tratamento diferenciado por utilizador.

Por tratamento diferencial entende-se, a capacidade do sistema em se enquadrar e se contextualizar perante diferentes utilizadores, conseguindo "lembrar-se" não só de diversos atributos característicos deste como por exemplo o seu nome, a sua idade, entre outros, mas também de tópicos de conversa anteriores, conseguindo retomar ou simplesmente abordar novamente esse tópico, num espaço temporal diferente daquele no qual foi originalmente debatido.

Todos os utilizadores, na primeira interacção com o sistema YEC, estão sujeitos ao mesmo treino do modelo, isto é, sob o mesmo histórico conversacional, devendo este ser inicialmente formulado de maneira a permitir ao modelo implementado, efetuar uma aprendizagem suficientemente capaz de viabilizar ao sistema, a realização de uma conversa estruturada e coerente com o seu utilizador, devendo este ter a capacidade de responder a questões genéricas como *"Como te chamas"*, *"Qual é a tua idade"*, *"Qual é a tua função"*, entre outras.

A formulação e tratamento deste histórico conversacional inicial, diz respeito à tarefa de **extracção inícial de dados**, realizada de tal maneira a permitir a alimentação destes no modelo YEC, de modo a realizar a sua primeira aprendizagem.

Depois de realizada a primeira conversa com o utilizador, esta é acoplada ao histórico conversacional original, sendo efetuada uma nova iteração de aprendizagem no modelo, **específica** para o utilizador em questão.

Toda esta informação de utilizador, nomeadamente o seu estado atual de aprendizagem em termos de modelo bem como o seu histórico conversacional, encontra-se armazenado numa directoria do sistema, específica para este, como se pode observar na figura 4.2.

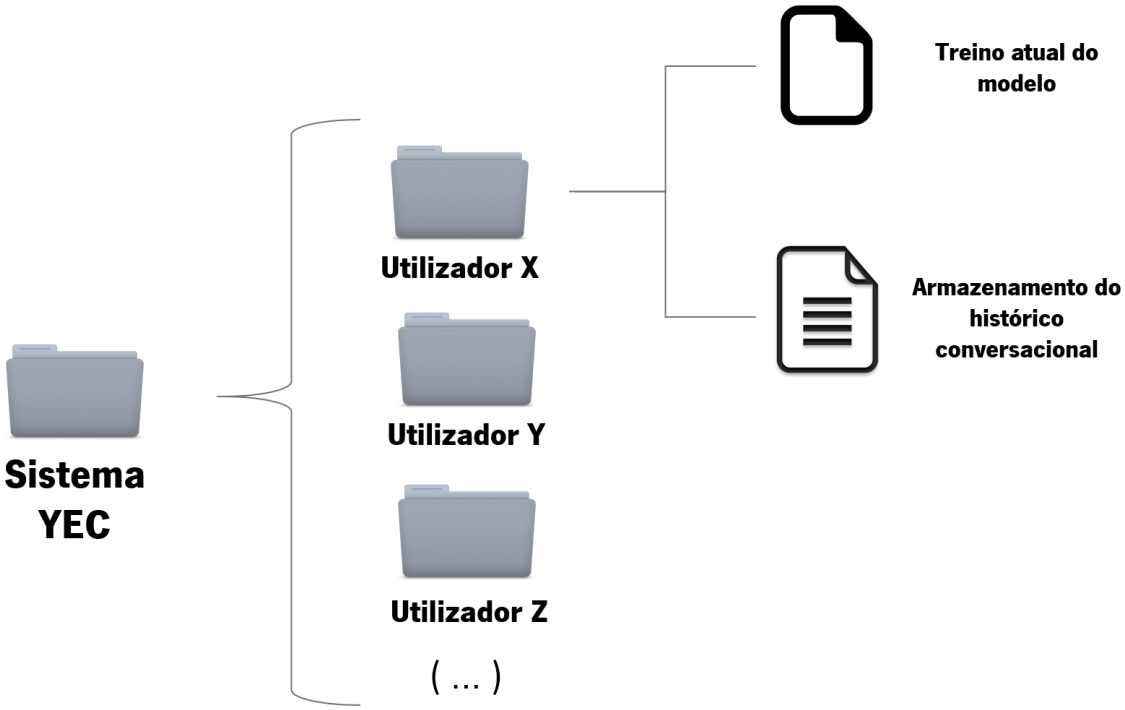


Figura 4.2 – Estrutura proposta para o armazenamento de conteúdo no sistema YEC, diferenciado por utilizador.

Devido à diversidade de utilizadores a interagirem com o sistema, torna-se necessário estabelecer uma maneira de efetuar o mapeamento entre o utilizador e o seu conjunto de dados. A projecção inicial passa pelo desenvolvimento de uma base de dados capaz de armazenar informação acerca de todos os utilizadores presentes no sistema, mapeando cada um com a sua directoria correspondente.

Esse rastreio do conjunto de utilizadores é possível devido ao sistema proposto, como iremos verificar nas sub-secções seguintes, ser projetado com um mecanismo de autenticação, obrigando cada utilizador a efetuar o registo no sistema para efetuar qualquer interacção com este.

A construção da base de dados representaria desta maneira, a formulação de uma entidade capaz de mapear cada utilizador com a localização de toda a sua informação.

Numa fase inicial, nesta diretoria apenas se encontra o modelo atualmente treinado para a interacção conversacional e o conjunto de conversas já efetuadas, mas numa perspectiva evolutiva, possuíndo cada utilizador um determinado espaço físico no sistema, torna-se possível incluir outro tipo de dados que possam vir a serem utilizados em YEC, como imagens ou videos, representativos de tipos de dados usualmente utilizados numa conversa virtual.

A lógica apresentada, relativa ao mapeamento utilizadores-directoria, poderia ser formulada da maneira ilustrada na figura 4.3.

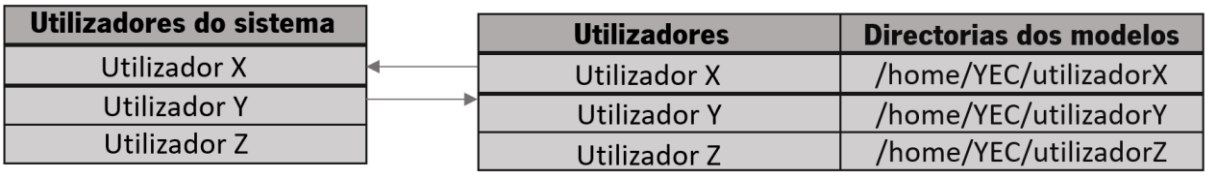


Figura 4.3 – Lógica projetada para efetuar o mapeamento diferenciado entre cada utilizador e a sua directoria do sistema.

4.2.2 DESENVOLVIMENTO DO MOTOR CONVERSACIONAL

Nesta sub-secção são apresentadas as diferentes fases e elementos que compõe o modelo de geração de resposta, proposto para o sistema YEC.

O modelo desenvolvido e adotado na construção de um Chatbot é interpretado como o núcleo do sistema. É ele o motor conversacional, encarregue de formular a resposta devolvida pelo sistema para o utilizador, sendo portanto, a sua projecção e concepção, o passo mais importante na construção do sistema YEC.

O tipo de modelo a adotar para a construção do sistema proposto, será um modelo do tipo **generativo**, baseado na combinação de técnicas de *Deep Learning* e NLP*,* apresentadas anteriormente. A geração de resposta por parte de YEC é dividida em duas fases: A fase de **treino** e a fase de **geração de resposta**, mais conhecida por fase de **previsão** na maior parte das tarefas de *Deep Learning*.

A figura 4.4 ilustra os passos envolvidos em cada uma das fases acima mencionadas, que permitem a formulação da resposta final a ser devolvida para o utilizador:

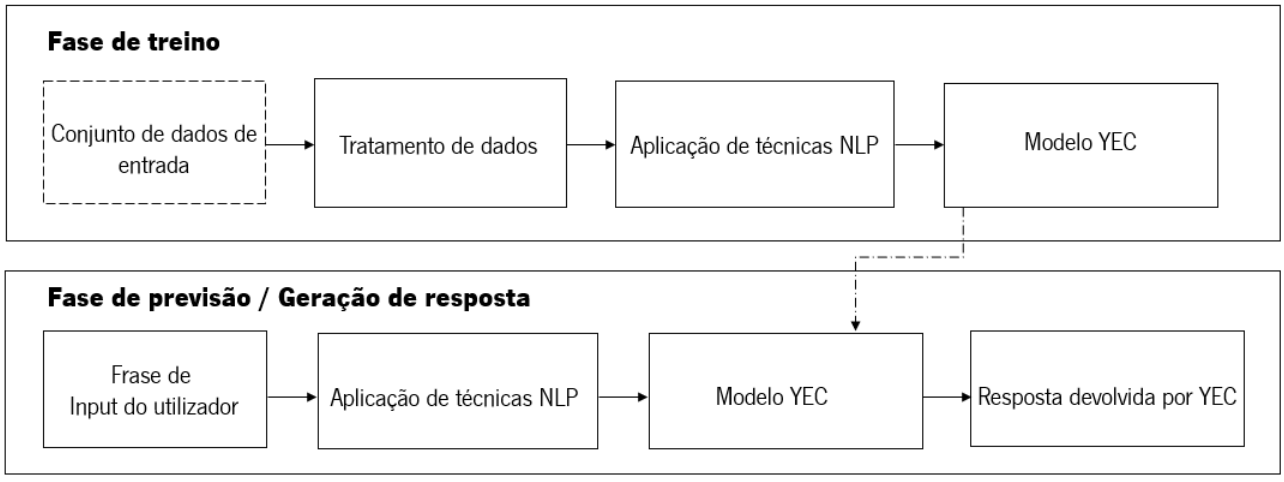


Figura 4.4 – Fases presentes na geração de resposta por parte do sistema YEC.

Fase de treino

Esta fase possui como produto final, o modelo de *Deep Learning* treinado, de maneira a efetuar por parte do sistema, a geração de resposta na fase seguinte.

Os modelos do tipo *encoder-decoder* têm possuído um enorme sucesso na tarefa de processamento textual e geração conversacional, pelo que modelo proposto para implementação no sistema YEC tem como base o modelo *sequence-to-sequence* apresentado na secção anterior.

Contudo, antes de ser efetuado o passo relativo ao treino do modelo, como se pode verificar no diagrama anterior, existem três tarefas importantes que devem ser realizadas, de forma a que o treino deste possua um desempenho minimamente aceitável para a geração de respostas coerentes e estruturadas.

O primeiro passo diz respeito ao tratamento dos dados de entrada, de maneira a realizar determinadas transformações importantes sobre estes, nomeadamente:

- → Passagem de letras maiúsculas para letras minísculas.
- → Remoção de qualquer tipo de acentuação.
- → Substituição de caracteres especiais de uma certa língua por caracteres genéricos. *Exemplo*: *ç* por c.

Apesar da ultima transformação apresentada resultar numa má formulação frásica, devido ao facto de conter erros sintáticos, tal é necessário devido ao facto dos modelos genéricos de processamento textual e das técnicas de NLP revelarem alguma dificuldade no processamento e interpretação desses caracteres, devendo por essa razão, numa primeira fase de experimentação, ser efetuada essa substituição.

Depois de efetuado o tratamento de dados, prontos a alimentarem o nosso modelo, o passo seguinte consiste na aplicação de técnicas de NLP, que permitam ao modelo interpretar e contextualizar os diferentes dados, resultando por consequência numa melhor aprendizagem.

Uma das técnicas essenciais e requeridas pelo modelo proposto, é a tokenização da frase de *input* e o consequente armazenamento sob a forma de um vetor de tamanho fixo. Qualquer outra técnica pode ser aplicada de forma a melhorar o desempenho da aprendizagem do modelo, não sendo possível prever quais as técnicas que resultam numa melhor aprendizagem, até ser realizado na prática o treino do modelo implementado.

O último passo consiste então no treino do modelo desenvolvido, até este atingir uma *performance* na qual é capaz de gerar a resposta correta, dadas as diferentes entradas do utilizador.

Fase de Geração de Resposta

Depois de treinado o modelo, o sistema deverá ser capaz de, perante uma certa frase de entrada, devolver uma resposta lógica, contextualizada e bem estruturada. Se tal não acontecer, é necessário rever os parâmetros de treino do modelo, efetuado na primeira fase, e modificá-los de maneira a que o seu desempenho melhore substancialmente.

No fundo, estas duas fases andam sempre interligadas e em constante alteração: Primeiramente, devido ao facto de no sistema projetado, o conjunto de dados de entrada crescer à medida que mais conversas são realizadas, entre o sistema e o utilizador , resultando progressivamente numa melhor aprendizagem do modelo: a interecção n entre utilizador e o sistema YEC, irá expéctavelmente, possuir resultados de treino do modelo superiores quando comparado com a interacção $n-1$.

Em segundo lugar, a quantidade diversificada de técnicas de processamento de linguagem natural, como as apresentadas anteriormente, permitem-nos uma abordagem de constante *tentativa-melhoria*, procurando conjugar essas técnicas com o conjunto de dados de entrada, de maneira a que a aprendizagem do modelo seja a melhor possível.

A geração do *output* por parte do sistema, não é nada menos que a previsão por parte do modelo, da resposta mais adequada em cada contexto conversacional, de acordo com o treino efetuado.

4.2.3 Disponibilização do motor conversacional

Depois de desenvolvido o modelo e ter sido atingido neste, um treino cuja performance lhe permita o estabelecimento de uma conversa estruturada e duradoura com o utilizador, é altura de realizar o último passo, de forma ao desenvolvimento do sistema YEC ficar concluído.

A última peça a desenvolver, consiste na exposição do modelo de geração conversacional implementado, através de **API**, garantindo os mecanismos de autenticação relevantes para o tratamento diferenciado de utilizador, sendo essa autenticação necessária para o armazenamento de todas as interacções utilizador-sistema, bem como para efetuar a diferenciação e proteção de dados.

Do ponto de vista do utilizador final, este não tem qualquer interesse pelo detalhe técnico e funcional do sistema desenvolvido, tendo como única expectativa, encontrar uma *interface* limpa e objetiva do sistema, de maneira a, aquando do fornecimento de uma pergunta por parte deste, seja devolvida uma resposta lógica, estruturada e enquadrada no seu contexto, não interessando a maneira como esta é formulada.

Uma API fornece uma maneira simples de tornar a lógica de geração conversacional do sistema, implementada no passo anterior, abstrata para o utilizador final.

No fundo, podemos pensar numa API como o mensageiro do sistema YEC, responsável por comunicar com o modelo de geração conversacional, fornecendo-lhe as diferentes entradas do utilizador para serem processadas por este. Depois de processadas, é responsabilidade da API, efetuar o reencaminhamento da resposta gerada pelo modelo desenvolvido, para o utilizador final.

A figura 4.5 é representativa do papel desempenhado pela API, na interacção entre o sistema e o utilizador. Resumindo, esta permite a não comunicação deste de forma directa com a componenete responsável pela geração de resposta, neste caso o nosso motor conversacional, estando essa função acoplada à API desenvolvida.

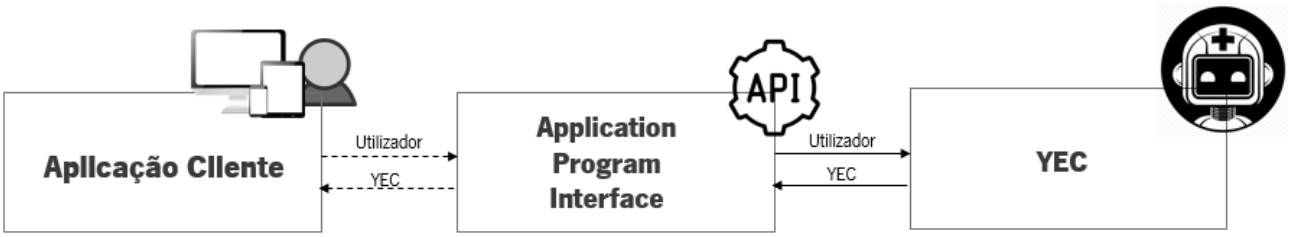


Figura 4.5 – Estrutura representativa da comunicação efetuada pela API entre o utilizador e o modelo conversacional.

Devido ao facto do sistema YEC efetuar a manipulação diferenciada de dados por utilizador, como apresentado anteriormente, revela-se necessária a presença de um mecanismo de **autenticação** na API desenvolvida, para permitir que a lógica do repositório de modelos e histórico conversacional seja possível de implementar.

O mecanismo de autenticação exige que qualquer utilizador que deseje interagir com o sistema YEC, se registe para tal efeito, tornando a manipulação de informação acerca dos diversos utilizadores mais fácil do ponto de vista do desenvolvedor.

Quando implementado, juntamente com o serviço de autenticação, devem estar presentes três partes envolvidas [103]:

→ O utilizador, que possui dados que são acedidos por meio da API e que deseja permissão para que o aplicativo os aceda.

→ O aplicativo, que deve aceder aos dados por meio da API, em nome do utilizador.

→ A API, que controla e permite acesso aos dados do utilizador.

Na autenticação, o utilizador fornece para a aplicação YEC, o seu nome de utilizador e a *password* escolhida aquando do seu registo, esperando obter permissão de interacção com o sistema e permitindo desta forma ao sistema identificar o utilizador e aceder aos seus dados prévios.

Depois disso, a aplicação entra em contacto direto com o mecanismo de autenticação , construído numa camada acima da API, de forma a solicitar um *token* de acesso, sendo este devolvido novamente para a API no instante seguinte com o objetivo de validar o acesso.

Depois de validado o acesso entre a aplicação e a API, tal é comunicado para o utilizador por parte da aplicação, sendo possível a partir desse momento a interacção do utilizador com o sistema [104]. O mecanismo acima descrito vai de encontro à lógica implementada atualmente na maior parte das APIs desenvolvidas, tendo como base o protocolo *OAuth*.

O motor conversacional, responsável por gerar a resposta retornada ao utilizador por intermédio da API, em nada se relaciona com o serviço de autenticação ou num primeiro passo, de forma directa com o utilizador que tenta aceder à aplicação. Todo esse mecanismo encontra-se acoplado à API desenvolvida.

O fluxo de comunicação total, na interacção utilizador e a aplicação YEC encontra-se ilustrada na figura 4.6.

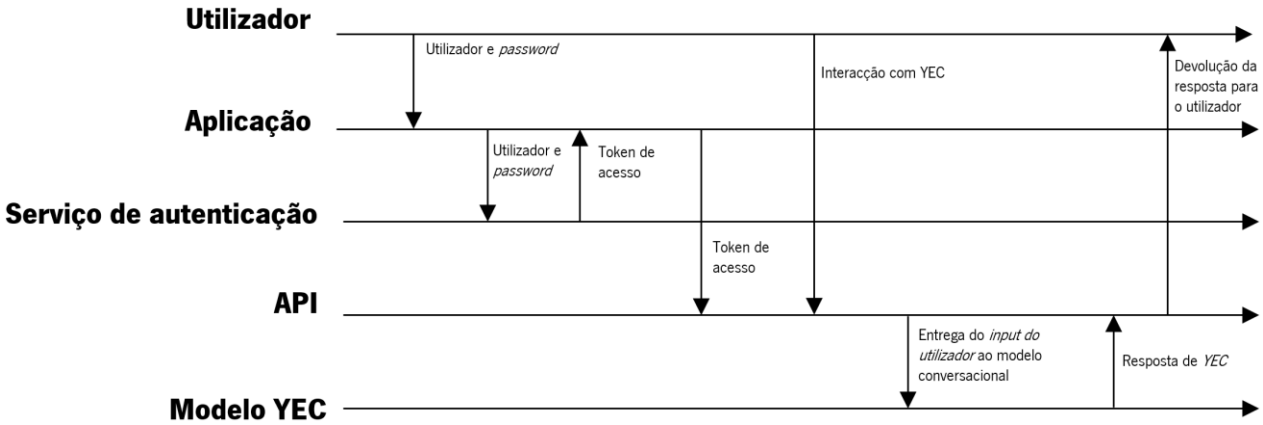


Figura 4.6 - Fluxo existente na comunicação entre o sistema YEC e o seu utilizador.

A autenticação é como já referido, um mecanismo crucial para a implementação da lógica apresentada até ao momento do sistema YEC. Graças a este mecanismo é possível diferenciar os utilizadores que interagem com YEC, fornecedo um certo tipo de memória virtual, sob a forma de realimentação constante das conversas anteriormente realizadas.

A figura 4.7 representa o fluxo de carregamento de dados para o modelo de geração de resposta, desde o momento que um utilizador se autentica no sistema YEC.

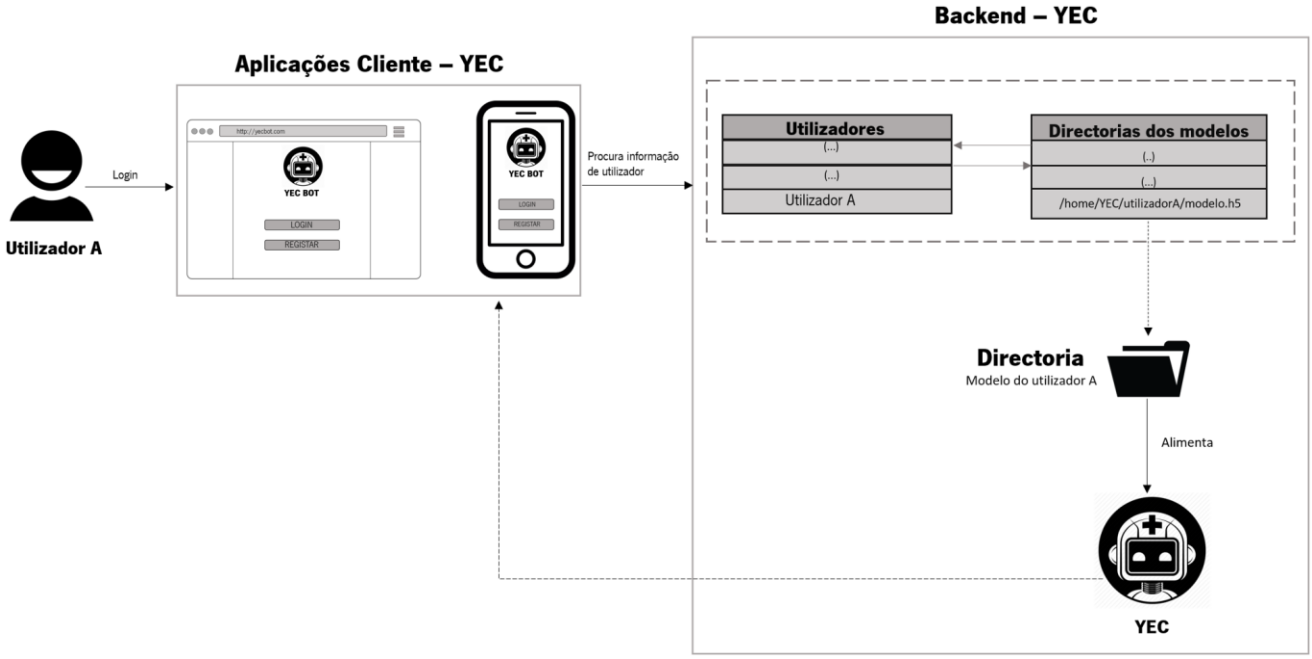


Figura 4.7 - Lógica de carregamento do modelo do utilizador, aquando da sua autenticação no sistema YEC.

4.3 ARQUITETURA GLOBAL DO SISTEMA YEC

Na secção anterior foram apresentados os componentes que formam o sistema YEC proposto nesta dissertação. Nesta sub-secção é efetuado um resumo da funcionalidade do sistema bem como da lógica presente nos diferentes componententes, terminando com a apresentação de um diagrama ilustrativo da arquitetura de desenvolvimento do sistema proposto.

Num momento inicial revela-se necessária a extracção de informação conversacional para suportar o treino e desenvolvimento de conhecimento comum a todos os utilizadores. Tal domínio conversacional é igual a todos os utilizadores no sistema, ficando toda esta informação registada sob a forma de directorias diferenciadas, em que cada uma contém o modelo do utilizador em questão devidamente treinado, bem como o seu histórico conversacional, podendo tal ser visto como um par diferenciado por (utilizador,directoria).

À medida que os diferentes utilizadores interagem com o sistema, estes vão realimentado o seu treino, evoluíndo desta forma o seu modelo, e permitindo assim uma auto-evolução do modelo conversacional. De facto parece má prática que YEC seja mais evoluído perante o utilizador X quando comparado com o utilizador Y, no entanto, desde que na primeira interacção, seja garantido para todos os utilizadores, de igual forma, a capacidade de formulação de resposta, a evolução de YEC é o rumo natural da inteligência artificial – quantos mais dados para aprendizagem do modelo, mais evoluído ele será.

Esta é a componente que diferencía YEC, da maior parte dos *chatbots* desenvolvidos – a sua capacidade de tratar cada utilizador como um individuo diferente, não olhando só para o **contexto frásico** do *input* recebido mas também para **quem** se encontra inserido nesse contexto.

Toda a resposta devolvida pelo sistema YEC é formulada no motor conversacional, considerando o histórico e evolução diferenciada por utilizador. Este modelo de geração conversacional tem como base técnicas de *Deep Learning* e NLP, possuindo o sistema, inteligência e dinâmica de treino devido à constante evolução da base de conhecimento que o suporta.

Numa última fase do projeto, seria garantida a exposição do modelo de conversação implementado, através de API, garantindo mecanismos de autenticação relevantes na diferenciação e proteção de dados necessários para suportar a lógica do sistema.

A utilização e desenvolvimento de uma API permitiria ao universo de potenciais aplicações-cliente, incorporadoras do modelo de conversação desenvolvido, abstrair-se de toda a lógica técnica e funcional implementada, preocupando-se só com a utilidade final do sistema – **conversar**.

A figura 4.8 ilustra a arquitetura global para desenvolvimento do sistema YEC , bem como as várias dependências entre os seus componentes.

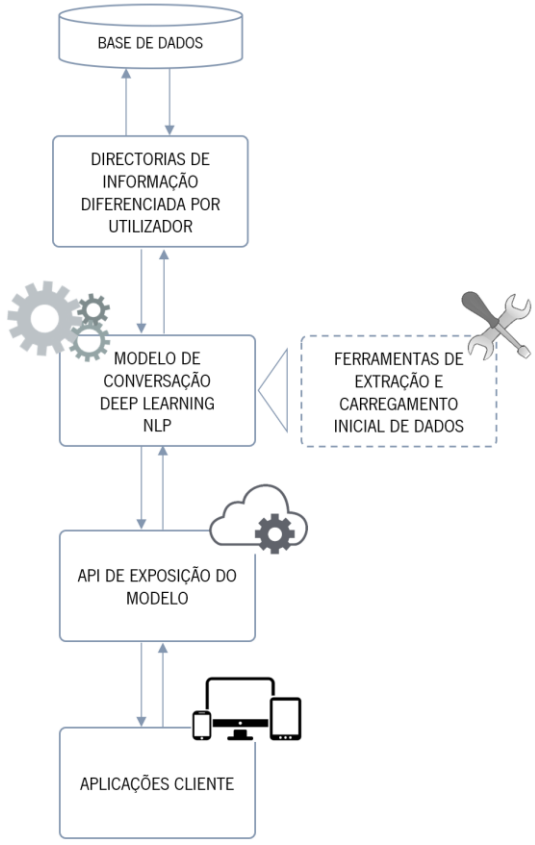


Figura 4.8 - Arquitetura global para implementação do sistema YEC.

CAPÍTULO 5
IMPLEMENTAÇÃO DO SISTEMA YEC

No presente capítulo é apresentada a abordagem prática que permitiu a realização parcial da arquitetura proposta anteriormente, desde a componente relativa à extracção inicial de dados, que permitiu a construção do *dataset* utilizado, até à implementação prática do motor conversacional, expondo todos os passos desde o pré-processamento sobre o conjunto de dados formulado até ao desenvolvimento e treino do modelo de *Deep Learning*.

Nesta dissertação não foram desenvolvidos os componentes relativos à disponibilização do modelo, não sendo criada nenhuma aplicação-cliente final. Por consequência, revelou-se impossível efetuar o desenvolvimento e projecção da base de dados, responsável por interligar os diferentes utilizadores às suas respectivas directorias de informação.

Por último, o capítulo termina com a comparação e discussão dos resultados, obtidos a partir da tarefa de previsão sob o modelo treinado, permitindo assim a apresentação de uma perspectiva global acerca do seu funcionamento e desempenho.

5.1 REQUISITOS

Os desenvolvimentos práticos do sistema YEC foram realizados sob a linguagem de programação Python, cuja escolha se deveu maioritariamente a três razões:

Primeiramente devido ao facto de Python ser a linguagem mais utilizada para processamento de dados textuais, fornecendo bibliotecas como a *Natural Language Toolkit (NLTK)*, que permitem a aplicação de técnicas de NLP ao nosso conjunto de dados, de forma simples e abstracta, possibilitando-nos preocupar apenas o propósito do seu uso e não lógica e metodologias que as compõe.

A segunda razão prende-se pelo facto da implementação e avaliação do modelo de *Deep Learning* a construir, ser efetuada em **Keras** – uma biblioteca escrita em Python, capaz de abstrair a lógica presente nas bibliotecas de *Deep Learning* de nível mais baixo como o *Tensorflow* ou o *Theano*.

Por último, Python revela-se como uma linguagem de *scripting* muito potente, permitindo o desenvolvimento de *scripts* para processamento e manuseamento de dados de forma rápida e eficaz, como se verificou no caso da construção do *dataset* utilizado neste projecto.

Depois de efetuado o desenvolvimento do motor conversacional, as tarefas de treino e previsão foram realizadas num sistema com as seguintes características:

→ **Sistema Operativo** – Ubuntu 14.04.5 LTS (64-bit)

→ **CPU** – Intel Xeon E5-1650 (6 cores – 12 processadores)

→ **RAM** – 64 gb

→ **Memória secundária** – 3 discos: dois de 2 TB e um de 512 gb para efetuar boot do sistema.

→ **GPU** – NVIDIA Quadro P6000

- Cuda Parallel-Processing Cores – 3840
- Memória da GPU – 24 GB GDDR5X
- FP32 Performance – 12 TFLOPS

5.2 CONSTRUÇÃO DO DATASET

Para efetuar a alimentação inicial e consequente treino primário do modelo de *Deep Learning* implementado, revelou-se necessário efetuar a construção de um *dataset* específico, que tivesse a capacidade de representar comportamentos conversacionais estruturados e consistentes.

A utilização directa de dados de conversas reais, sem qualquer tipo de filtro ou tratamento, revela-se normalmente, um obstáculo dificil de ultrapassar aquando do seu processamento no treino do modelo, resultando na maior parte dos casos numa má aprendizagem devido ao facto destes conterem bastante ruído, sob a forma de erros ortográficos, informação errada, acentuação, entre outros, originando um mau desempenho na formulação de resposta por parte do sistema.

Adicionalmente a este problema, a falta de dados estruturados e que representassem conversas reais, também se revelou uma tarefa árdua de superar, não havendo nenhum *dataset* referente a conversas na língua portuguesa, disponível para utilização, pelo que a solução adoptada se baseou na extração destes dados a partir das nossas próprias conversas, localizadas em diferentes redes sociais.

Tanto o *Facebook* como o *Twitter* fornecem a possibilidade ao utilizador, deste descarregar todos os dados a si referentes, desde imagens partilhadas até às conversas efetuadas no *chat*.

Como se pode verificar na imagem 5.1, os dados podem ser criados em dois tipos de formatos diferentes: HTML e JSON.

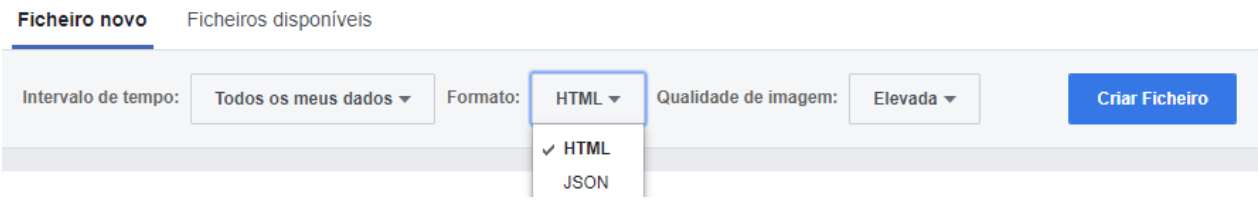


Figura 5.1 - Ferramenta presente na rede social facebook para extração de dados relativos ao utilizador.

Devido ao facto do conjunto de dados conversacionais a utilizar no modelo desenvolvido, necessitar de se encontrar sob o formato CSV, revelou-se imprescindível, o desenvolvimento de uma *script* escrita em Python, que permitisse a geração de um ficheiro nesse formato, a partir do ficheiro JSON extraído.

A figura 5.2 ilustra a sequência de acções implementadas, para a geração do ficheiro CSV.

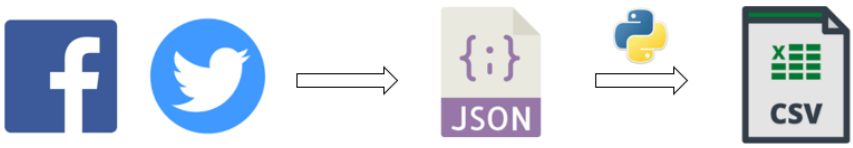


Figura 5.2 - Lógica implementada para a construção do dataset inicial a ser utilizado pelo sistema.

O desenvolvimento desta *script* foi efetuado com base nas bibliotecas *"json"* e *"csv"*, disponibilizadas pela linguagem Python, capazes de fornecer os métodos necessários para o correto manuseamento desses mesmos tipos de dados, permitindo no final, a transformação de um ficheiro JSON para um ficheiro do tipo CSV, estruturado sob a forma n linhas $\times$ 2 colunas – Uma coluna relativa às frases enviadas por um utilizador e outra relativa às respostas do outro agente conversacional.

Um exemplo desta conversão encontra-se ilustrado na figura 5.3:

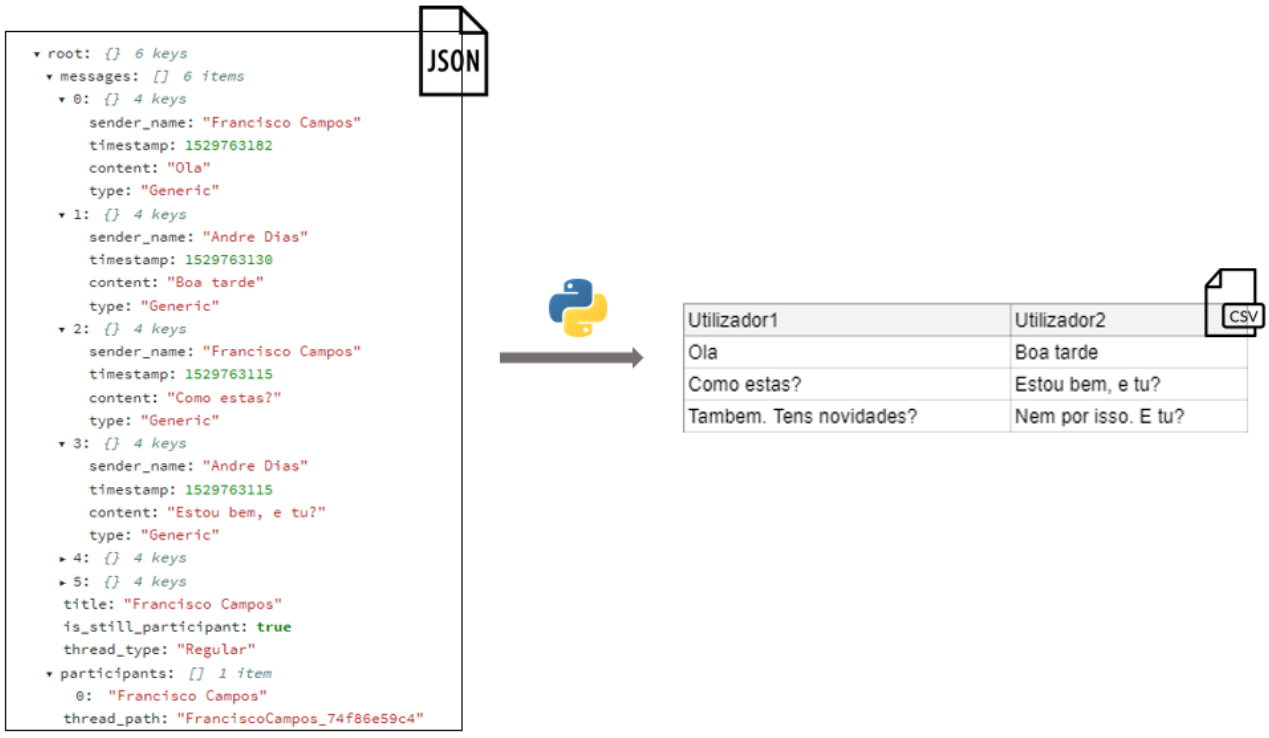


Figura 5.3 - Exemplo da transformação de um excerto conversacional extraído em formato JSON para formato CSV.

Todo o *dataset* utilizado no treino do modelo desenvolvido, foi formulado através da *script* acima referida. Devido à falta de interacções genéricas e que permitissem efetuar a aprendizagem do modelo de forma a que este se assumisse como uma entidade personificada, as respostas a perguntas como "Qual a tua idade" , "Como te chamas?" ou "Qual o teu trabalho?" foram modificadas, e na falta destas, formuladas manualmente, sendo posteriormente adicionadas ao *dataset.* A sua alteração foi efetuada com o recurso a expressões regulares, permitindo a modificação de respostas do género, "Chamo-me João", para "Chamo-me YEC".

Através da utilização de um conjunto de dados construído de raíz e no qual se procedeu à eliminação de ruído originalmente presente nos dados, permitiu-nos obter um maior controlo sobre o formato da conversa gerada pelo sistema, levando a uma avaliação mais objetiva e clara acerca dos pontos fortes e limitações deste.

5.3 DESENVOLVIMENTO DO MOTOR CONVERSACIONAL

A geração de resposta a devolver para o utilizador é uma tarefa encarregue daquilo a que chamamos de motor conversacional, constituído pelos módulos de pré-processamento, treino e previsão, como ilustrado na figura 5.4.

Apesar da interacção com o utilizador final, se realizar apenas no terceiro módulo, relativo à previsão, o núcleo do motor conversacional encontra-se no módulo relativo ao treino do modelo, onde se implementa o modelo de *Deep Learning* com o auxílio da biblioteca Keras, bem como se efetua a sua aprendizagem, influenciando diretamente o módulo que o sucede.

1. **Módulo relativo ao pré-processamento** realizado sobre o conjunto de dados a alimentar no modelo. Para além de qualquer tratamento sobre a amostra de dados, é neste módulo que é declarada a percentagem de dados de **treino** (que têm o propósito de treinar o modelo, isto é, fazê-lo aprender) e a percentagem de dados de **teste**, aos quais os dados de treino são aplicados, com o objetivo de inferir sobre a credibilidade e grau de confiança do modelo.
2. **Módulo relativo ao treino do modelo**, onde se encontra presente a nossa implementação do modelo encoder-decoder *(seq2seq)* e a aplicação de técnicas de processamento de linguagem natural. É neste módulo que se realiza o treino do modelo implementado, alimentado com o conjunto de dados previamente processados. De maneira a avaliar a sua performance, o treino é realizado com diferentes hiperparâmetros em cada iteração, declarados no presente módulo, que em conjugação com as

métricas estabelecidas para a sua avaliação, permitem-nos encontrar os seus valores ótimos, resultantes no treino mais consistente para o modelo.

3. **Módulo de previsão**, que permite ao utilizador testar o modelo treinado, fornecendo novas entradas, na esperança de que a resposta devolvida por este seja coerente e devidamente estruturada. No fundo é este o objeto final do desenvolvimento, capaz de estabelecer conversas com os seus utilizadores.

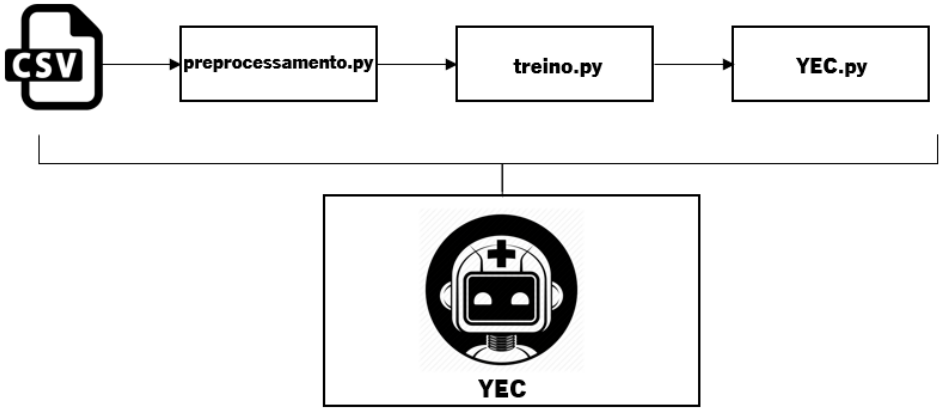


Figura 5.4 - Fluxo presente entre os diferentes módulos desenvolvidos e que compõe o motor conversacional presente em YEC.

5.3.1 PRÉ-PROCESSAMENTO DO CONJUNTO DE DADOS

O primeiro passo na construção do motor conversacional, consistiu no desenvolvimento do módulo relativo ao pré-processamento do *dataset*, anteriormente formulado. Este conjunto de dados, será responsável por alimentar o módulo seguinte, com o objetivo de efetuar o treino do modelo, nele implementado, sendo que, quanto mais limpos e claros estes dados forem, melhor será a aprendizagem realizada por parte deste.

Após observação do *dataset* criado, foram identificados alguns pontos de melhoria:

→ O texto presente nos dados extraídos, contêm uma mistura de letras maiúsculas e letras miniscúlas.

→ Existem caracteres específicos da língua portuguesa no conjunto de dados.

→ Existem frases com estrutura semelhante mas que no entanto, possuem significados diferentes.

Existem possivelmente, mais alguns factores que devem ser melhorados no *dataset*, no entanto os três pontos acima identificados são os mais críticos e os que saltam mais à vista numa primeira análise. A verdade é que temos sempre a hipótese de numa fase posterior procurar mais alterações sob este conjunto de dados, tendo em vista a elevação da qualidade de treino do modelo, dado que este é um processo longo e iterativo, nunca sendo atingido um desempenho considerado máximo.

De modo a aplicar as correcções acima mencionadas, o primeiro passo consistiu na criação de uma função que permitisse o carregamento do dataset criado, numa maneira que fosse possível preservar os caracteres especiais da língua portuguesa, como por exemplo o "ç" ou termos que envolvam qualquer tipo de acentuação:

```
def carregar_dataset_limpo(filename):
    return load(open(filename, 'rb', encoding='iso-8859-1'))
```

A função responsável por efetuar a limpeza dos dados foi construída com o auxílio das bibliotecas *"re"*, fornecedora de operações de mapeamento de dados através de expressões regulares e da biblioteca *"unicodedata"*, capaz de conceder acesso à base de dados de caracteres *unicode*, onde se encontram definidas todas as propriedades para estes caracteres, permitindo assim uniformizar e eliminar os caracteres especiais, específicos da língua portuguesa.

```
def limpar_dataset(lines):
    cleaned = list()
    re_print = re.compile('[^%s]' % re.escape(string.printable))
    # preparar tabela de traduçao
    table = str.maketrans('', '', string.punctuation)
    for pair in lines:
        clean_pair = list()
        for line in pair:
            # normalizar os caracteres sob padrao unicode
            line = normalize('NFD', line).encode('ascii', 'ignore')
            line = line.decode('UTF-8')
            line = line.split()
            # converter para letra miniscula
            line = [word.lower() for word in line]
            # remover caracters especiais
            line = [re_print.sub('', w) for w in line]
            # remove caracteres com numeros associados
            line = [word for word in line if word.isalpha()]
            # armazenar como string
            clean_pair.append(' '.join(line))
        cleaned.append(clean_pair)
    return array(cleaned)
```

Depois de efetuado o tratamento sob todo o *dataset*, obtivemos um bom número de exemplos para o treino do modelo conversacional. A geração de um ficheiro para armazenar estes dados, já transformados, foi efetuada com o auxílio da biblioteca *"pickle"*, sendo esta não só utilizada para armazenamento, mas também para carregamento de todos os ficheiros de dados desenvolvidos.

A complexidade do nosso sistema conversacional aumenta com o número de exemplos, com o seu comprimento frásico e com o tamanho do vocabulário de entrada.

O *dataset* a utilizar sobre o modelo não possui frases extensas, tendo sido estabelecido um comprimento máximo igual a oito termos por frase, sendo o vocabulário inicial pouco maior do que três centenas de termos.

Numa fase inicial, acredita-se que esta seja a melhor abordagem para realizar um treino mais eficaz do modelo e inferir acerca da qualidade da solução, apesar desta limitar a aprendizagem do sistema e por consequência, o seu espectro de devolução de resposta.

De maneira a testar o desempenho do modelo no módulo seguinte, dividiu-se o *dataset* em dados de treino e de teste, da seguinte forma:

- *X_train* – conjunto de dados de treino, representativo de 90% da amostra inicial.
- *X_test* – conjunto de dados de teste, representativo de 10% da amostra inicial.

O módulo de pré-processamento termina com um *shuffle* de todo o conjunto de dados. O *shuffle*, é um método presente na biblioteca Python *"random"*, capaz de tornar aleatória a sequência de perguntas e respostas presentes no dataset, impedindo desta forma a introdução de alguma tendência no sistema.

5.3.2 TREINO DO MODELO

Como já referido, o propósito do sistema YEC é efetuar a previsão e retornar para o utilizador, a resposta correta num dado contexto. Dado esse propósito, é então possível resumir o objetivo do treino do modelo, como sendo a maximização da probabilidade de acerto na resposta devolvida, isto é:

$$output_{sistema} = \max(p(resposta|contexto)).$$

Tanto o contexto como a resposta são vistas por parte do sistema, como sequências de palavras, transformando assim, o objetivo anteriormente apresentado, para a maximização da probabilidade de gerar corretamente a sequência de *tokens* a devolver para o utilizador, dada uma sequência de *tokens* de entrada, tida como contexto, tal que:

$$output_{sistema} = \max(p(r_1, \dots, r_n|c_1, \dots, c_m)).$$

Dada a formulação anterior do objetivo do sistema, o modelo *encoder-decoder* adequa-se de maneira perfeita a esse mesmo propósito, que conjugado com a sua facilidade de implementação e modificação, fizeram com fosse a base para o modelo desenvolvido neste segundo módulo.

O primeiro passo efetuado na componente de treino, consistiu no carregamento de todos os conjuntos de dados formulados no módulo anterior, nomeadamente, do conjunto de dados de teste, treino e o conjunto de dados total (treino + teste). A declaração de um ficheiro relativo ao conjunto total de dados, permitirá definir mais facilmente atributos relevantes a serem utilizados pelo modelo como o comprimento frásico máximo e o conjunto de vocabulário induzido e inferido pelo sistema.

Uma alternativa, seria definir estas propriedades no conjunto de dados de treino, limitando nos dados de teste o conjunto de frases que fossem maiores que o declarado, ou que possuíssem vocabulário não

presente no conjunto de dados de treino, no entanto esta abordagem permitiu uma automatização deste processo.

Depois de efetuado o carregamento dos dados, foi desenvolvida uma função que nos permitisse aplicar uma das técnicas de NLP de análise morfológica, mais específicamente, a técnica de tokenização, capaz de fragmentar sob a forma de tokens, as frases presentes nos diferentes conjuntos de dados.

A biblioteca Keras fornece o seu próprio *tokenizer*, designado para não só de dividir sob termos, as diferentes frases, mas também de mapear esses termos a números inteiros, criando um género de índice.

```
1.  from keras.preprocessing.text import Tokenizer
2.  # tokenizacao das frases
3.  def tokenizacao(lines):
4.      tokenizer = Tokenizer()
5.      tokenizer.fit_on_texts(lines)
6.      return tokenizer
```

Depois de aplicado ao conjunto de dados, este *tokenizer* fornece bastantes operações úteis para o manuseamento de informação, como por exemplo a disponibilização da contagem de cada termo nos dados fornecidos ou o cálculo do número de frases distintas em que um dado termo é referido. Estas acções revelam-se bastante pertinentes aquando da análise da aprendizagem do sistema, permitindo inferir se por exemplo, existe um desbalanceamento na frequência de termos.

Depois de preparado o *tokenizer* a ser utilizado sobre o nosso modelo, definido o vocabulário e estabelecido o comprimento frásico máximo para o nosso conjunto de dados, é então altura de preparar o conjunto de dados para o treino:

Todas as frases de entrada e saída devem ser mapeadas para números inteiros e preenchidos até ao valor máximo de comprimento frásico previamente estabelecido (técnica em inglês denominada *padding)*, para a sua correta utilização no modelo encoder-decoder. Esta acção prende-se pelo facto de na implementação do nosso modelo *sequence-to-sequence*, acoplarmos à primeira rede *(encoder)*, uma camada adicional para aplicação da técnica de *word embedding*, apresentada anteriormente, sob o nosso conjunto de dados de entrada, permitindo-nos alcançar melhores resultados na tarefa de geração conversacional.

A necessidade de aplicar a técnica de *one-hot encoding* à frase devolvida como saída da rede reside no facto do modelo implementado ter como funcionamento, a previsão da probabilidade de cada palavra presente no vocabulário, devolvendo os termos mais prováveis como resposta. Depois de aplicadas as duas funções acima apresentadas ao nosso conjunto de dados de treino e teste, é altura de definir o modelo.

A estrutura genérica do modelo de *Deep Learning* implementado nesta dissertação encontra-se ilustrada na figura 5.5:

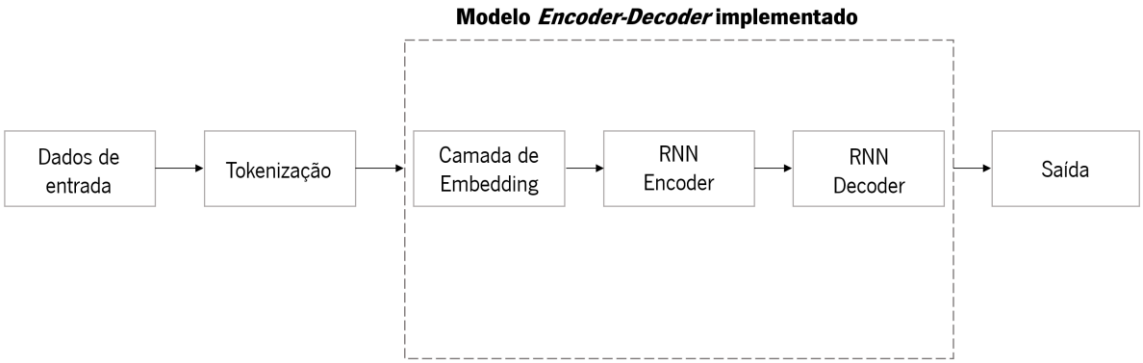


Figura 5.5 - Sequência de processos ocorridos no módulo de treino, com vista a aprendizagem do modelo implementado.

O modelo implementado nesta dissertação é um modelo *encoder-decoder*, também chamado de sequence-to-sequence, cujo funcionamento se encontra explicado no capítulo três. Adicionalmente à sua estrutura original, constituída por duas redes RNN, á primeira rede foi adicionada, como já referido, uma camada de *embedding*, com vista um aumento do desempenho da aprendizagem do modelo, quando comparado à estrutura original.

Para o estebelecimento de medidas comparativas, as redes *encoder* e *decoder* foram elaboradas de duas maneiras distintas: Numa primeira implementação o tipo de células RNN utilizadas foram redes do tipo LSTM enquanto que numa segunda abordagem foram redes do tipo Gru-RNN.

Desta forma, tornou-se possível estabelecer as diferenças, se existentes, em termos de desempenho, quando utilizado cada um dos dois tipos de rede , permitindo aferir sobre a melhor implementação para o modelo desenvolvido.

Ambas as redes são disponibilizadas para utilização pelo módulo *Keras*, que para além destas, fornece também bibliotecas para implementação do tipo de modelo, neste caso, um modelo do tipo Sequencial, dado se tratar de uma tarefa de geração conversacional, onde a ordem temporal é relevante e deve ser cumprida de maneira a obter resultados coerentes.

A camada *Embedding* é também fornecida pela biblioteca *Keras*, requisitando apenas que os dados de entrada estejam codificados, de forma única, sob números inteiros, tarefa já realizada aquando da aplicação do *tokenizer*, sendo a sua aprendizagem é realizada simultaneamente com o resto do modelo.

```
1. from keras.layers import LSTM
2. from keras.layers import GRU
3. from keras.layers import Dense
4. from keras.layers import Embedding
5. from keras.layers import RepeatVector
6. from keras.layers import TimeDistributed
7. # Modelo Encoder-Decoder em YEC
8. def YEC_model(vocab, timesteps, n_units):
9.     model = Sequential()
```

```
    model.add(Embedding(vocab, n_units, input_length=timesteps, mask_zero=True))
    #model.add(LSTM(n_units)
    model.add(GRU(n_units))
    model.add(RepeatVector(timesteps))
    model.add(GRU(n_units, return_sequences=True))
    #model.add(LSTM(n_units, return_sequences=True))
    model.add(TimeDistributed(Dense(vocab, activation='softmax')))
    return model
```

A função *YEC_model()* apresentada acima, demonstra o aspecto final do modelo implementado, possuíndo um certo número de argumentos utilizados para a sua configuração, nomeadamente o tamanho de vocabulário, o comprimento frásico máximo extraído anteriormente e o número de células utilizadas em cada uma das RNN de *encoder* e *decoder*.

O modelo é treinado utilizando o optimizador *"Adam"*, cuja escolha residiu no facto deste ser extremamente eficiente e alcançar bons resultados num curto espaço de tempo. Utilizando modelos e conjuntos de dados extensos, foi demonstrado, como se verifica na figura 5.6, que o algoritmo *Adam* consegue resolver os diversos problemas de *Deep Learning* de uma maneira mais eficiente quando comparado com outros algoritmos, fazendo dele uma boa escolha na generalidade dos casos [105].

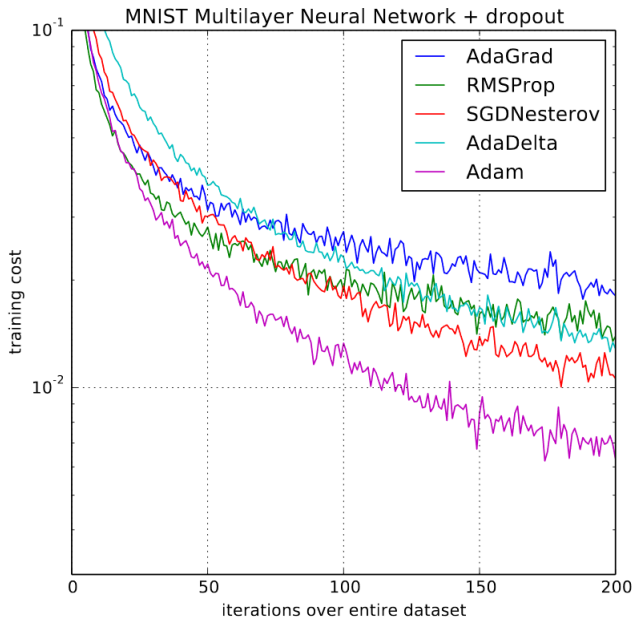


Figura 5.6 - Desempenho do algoritmo Adam quando comparado com outros algoritmos de optimização, aquando do treino de uma rede ANN (retirado de [105]).

Devido ao facto do problema de previsão, exposto num sistema conversacional, poder ser descrito como uma classificação multi-classe, i.e.prever os termos mais corretos a devolver em cada fracção temporal, a função de perda (em inglês, *loss function*) escolhida para ser minimizada durante o treino, foi a função *categorical crossentropy*, traduzindo-se esta na seguinte equação:

$$Perda(y_i, \acute{y}_i) = -y_i \log(\acute{y}_i)$$

Onde $\acute{y}_i$ representa o vector representativo das probabilidades previstas pelo modelo sobre todas as palavras no vocabulário formulado, num instante de tempo igual a i, e em que y_i diz respeito ao vector de saída *(one-hot)* devolvido pelo sistema [106].

A função de perda, aplicada num sistema conversacional, é baseada no cálculo do quão distintos são a resposta prevista pelo sistema, da atual resposta tomada como válida nos dados de treino. Várias investigações e pesquisas de diversos autores como *Vinyals* e *Le* [107] , *Serban et al* [108] e *Li et al* [109], revelam, contudo, que devido à diversidade de respostas possíveis para uma determinada entrada de utilizador, a função de perda revela-se como uma má métrica na avaliação de chatbots.

Isto porque, por exemplo, dada a pergunta "Como foi o teu dia?", o sistema tem ao seu dispor uma infinidade de respostas possíveis, sendo a simples devolução da resposta "foi bom", do ponto de vista do utilizador, uma resposta adequada e bem formulada, mas que no entanto, se iria traduzir, como métrica de qualidade da previsão, numa má resposta se nos dados de treino esta não se revelar uma resposta válida.

Devido ao facto da geração conversacional ser uma tarefa tão ambígua, a função de perda tradicional, não é mais indicada para efetuar a análise e treino de bons Chatbots, devido ao facto de aquando da existência de várias respostas para a mesma pergunta, no conjunto de dados de treino, a função de perda fazer com que o modelo ganhe **tendência** a devolver respostas genéricas e aquelas que considera mais "seguras".

No entanto, na presente dissertação, apesar de a função de perda não ser utilizada para classificar o sistema desenvolvido, os gráficos de perda são apresentados com o intuíto de perceber quais os hiperparâmetros a definir no modelo aquando do treino seu, que resultam num melhor desempenho do modelo.

```
1. # definir e compilar o modelo
2. model = define_model(all_vocab_size, all_length, 256)
3. model.compile(optimizer='adam', loss='categorical_crossentropy')
```

A seleção dos parâmetros de treino do algoritmo é um processo que não segue uma ordem concreta, ou seja, não existe um padrão para a sua realização. È preciso ter em mente que pequenas oscilações nestes parâmetros podem levar a grandes diferenças, tanto no tempo de treino como no modelo obtido.

A combinação destes parâmetros com o uso de poucas células nas camadas intermédias, tanto na rede *encoder* como na rede *decoder*, pode levar a que estas sejam incapazes de modelar dados mais complexos, resultando num fenómeno chamado *underfitting*. Esta situação torna-se percetível quando o erro é demasiado grande, tanto nos casos de treino como nos casos de teste.

Pelo contrário, caso o número de células seja elevado relativamente ao conjunto de dados presentes, poderá contribuir para o excesso de treino da rede, ocorrendo assim o fenómeno de *overfitting* e, por conseguinte, resultando na perda da capacidade de previsão por parte do modelo.

A figura 5.7 remete para um gráfico um gráfico ilustrativo da ocorrência dos fenómenos de *overfitting* e *underfitting*, explicados anteriormente.

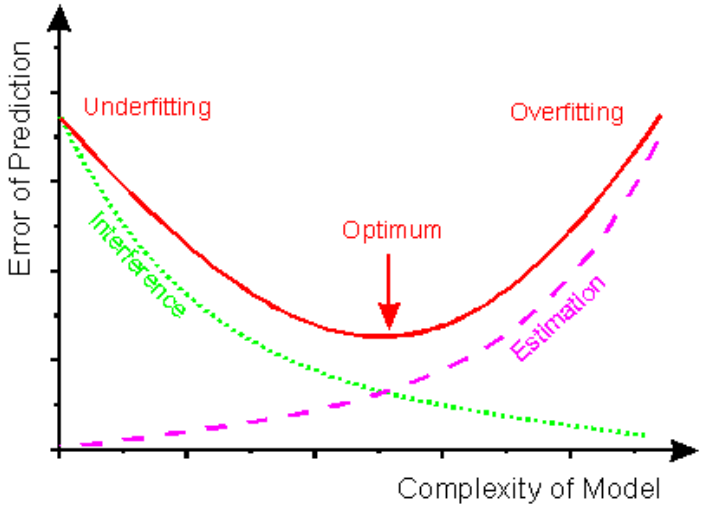


Figura 5.7 - Gráfico relativo aos fenómenos de underfitting e overfitting, retirado de [105].

Depois de compilados e definidos os parâmetros de optimização e de perda, procedeu-se ao treino do modelo. Este foi realizado a partir do módulo *fit*, fornecido pela biblioteca Keras e consistiu em duas fases, uma para cada tipo de rede implementado:

Em cada fase o modelo foi treinado em três ocasiões distintas, referentes a um diferente número de *epochs* em cada iteração: 60, 120 e 200. O valor do *batch size*, manteve-se constante em todas as iterações, sendo constituído por 64 exemplos, valor extraído das experimentações realizadas por *Neishi, Sakuma, Tohda, Yoshinaga* et. Al [110]. Foi ainda utilizado um mecanismo de *checkpoint* para garantir que cada vez que o desempenho do modelo melhore durante a fase de teste, o modelo seja gravado para um ficheiro distinto, de modo a ser utilizado na fase de previsão.

Os gráficos de acerto (em inglês, *accuracy*), não são expostos devido ao facto de não possuírem uma importância significativa na análise a efectuar, pois nos sistemas de geração conversacional, a leitura de gráficos de *accuracy* revela-se uma tarefa árdua de executar pois estes tendem a conter um elevado número de falsos negativos – número de exemplos positivos, classificados como negativos.

De seguida apresentam-se os resultados, relativos aos gráficos de perda, para análise da aprendizagem do modelo desenvolvido.

→ **Redes Gru-RNN**

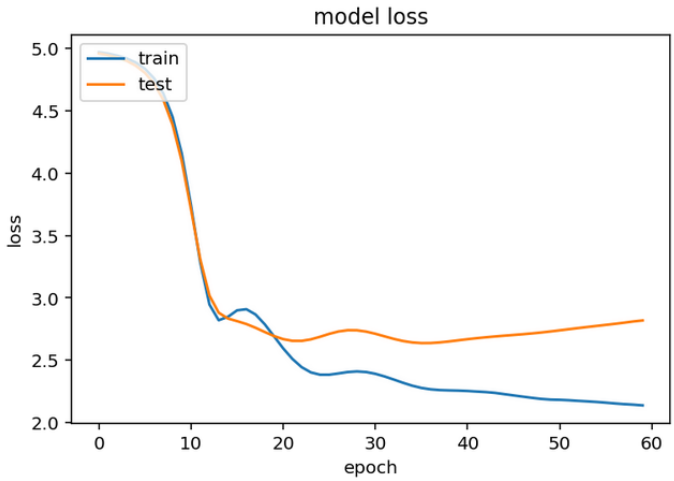


Figura 5.8 - Gráfico de perda aquando do treino do modelo com redes Gru para 60 epochs.

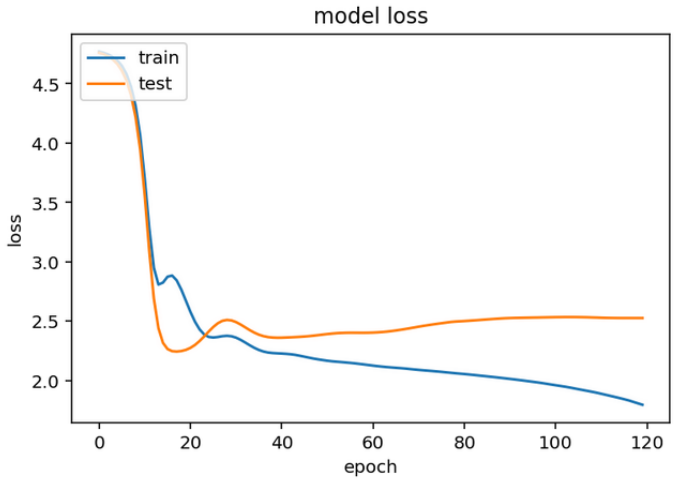


Figura 5.9 - Gráfico de perda aquando do treino do modelo com redes Gru para 120 epochs.

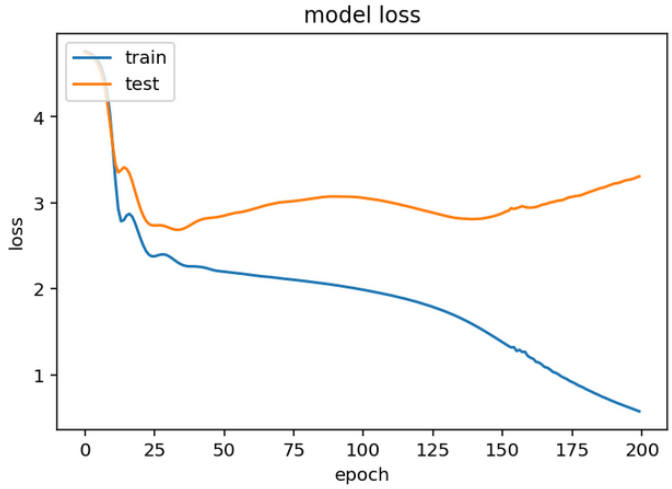


Figura 5.10 - Gráfico de perda aquando do treino do modelo com redes Gru para 200 epochs.

Podemos verificar a partir da observação dos gráficos relativos a 120 e 200 *epochs,* que treino do modelo, entrou em *overfitting* que tal como mencionado anteriormente, ocorre quando a rede é extremamente treinada, resultando na perda da capacidade de previsão. No ponto em que o modelo atinge os 40-50 epochs, sensivelmente, significa que o modelo chegou ao máximo da sua aprendizagem e não necessita de mais *epochs* para treinar.

Até este ponto, verifica-se que a perda dos dados de treino acompanha a perda dos dados de teste, havendo inclusive, ocorrências pontuais em que a perda dos dados de teste supera as de treino, o que é bastante bom, uma vez que significa que, apesar dos dados de teste corresponderem a apenas 10% do total de dados, a sua aprendizagem foi semelhante à do conjunto de dados de treino, que correspondia a 90% dos dados. O máximo de aprendizagem foi obtido para um valor de 35 epochs e um batch size de 64.

→ **Redes LSTM**

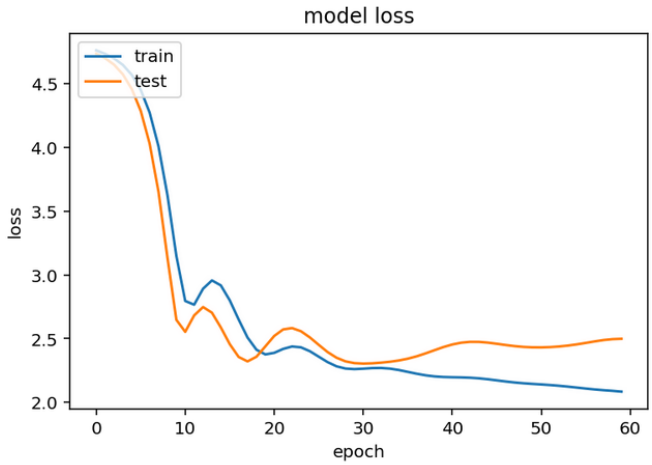


Figura 5.11 - Gráfico de perda aquando do treino do modelo com redes LSTM para 60 epochs.

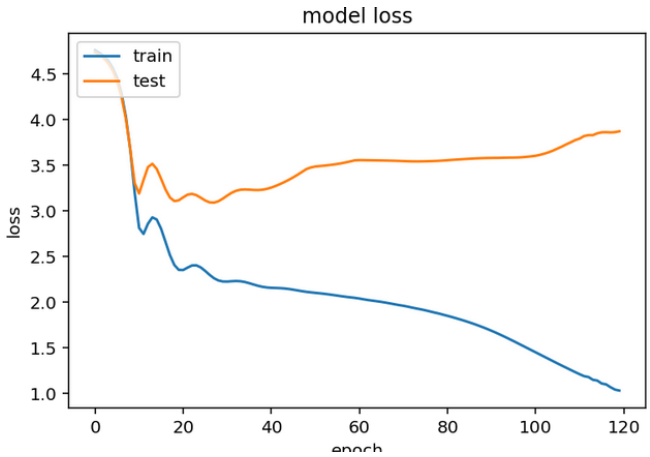


Figura 5.12 - Gráfico de perda aquando do treino do modelo com redes LSTM para 120 epochs.

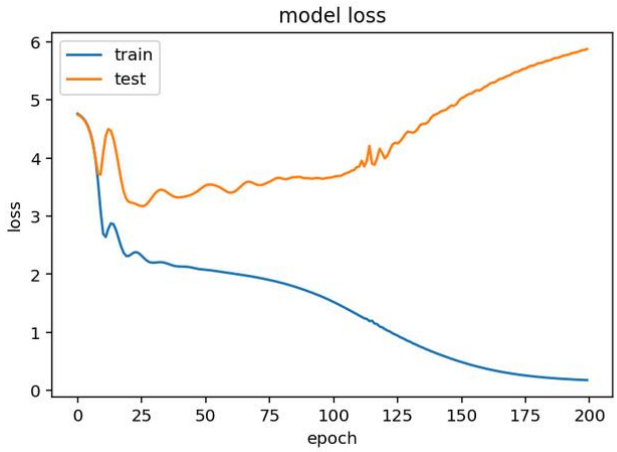


Figura 5.13 - Gráfico de perda aquando do treino do modelo com redes LSTM para 200 epochs.

Semelhantemente ao ocorrido nas redes do tipo Gru, o treino do modelo com recurso a redes LSTM demonstra novamente a ocorrência do fenómeno de *overfitting* por volta das 40-50 *epochs*, o que nos leva a concluir que para ambas as implementações, o valor ideal de *epochs* reside nesse intervalo.

Pela experiência realizada, podemos afirmar que para este caso, as redes Gru efetuam o treino de forma mais rápida, superando ligeiramente as redes do tipo LSTM em termos de desempenho de treino e minimização da função de perda, como se pode verificar por observação da escala gerada nos gráficos, acima apresentados. A obtenção de melhores resultados de treino através da utilização de redes do tipo Gru, quando comparado com as redes do tipo LSTM, para um conjunto de dados considerado "reduzido" para o problema em questão, pode ser explicado pelo número de parâmetros de aprendizagem de cada uma das redes:

As redes LSTM, derivado do facto de possuírem mais uma porta lógica (três na totalidade, como verificado no capítulo 3) na sua implementação, faz com que o seu número total de parâmetros de aprendizagem seja dado por $4(n^2 + nm + n)$, superior ao número de parâmetros de aprendizagem na rede Gru, dado por, $3(n^2 + nm + n)$ [111].

Na verdade, esta era uma das conclusões que se pretendia retirar desta experiência, provando a veracidade do exposto no capítulo 3, onde se afirmou que este tipo de redes permitiria obter resultados ligeiramente melhores.

Finalizado o treino do modelo, é altura de passar para a fase de previsão de resposta por parte do modelo com melhor desempenho, que se revelou ser o modelo implementado com redes do tipo Gru-RNN, para um número de *epochs* igual a 40 e com um *batch size* igual a 64 unidades, evitando assim a ocorrência do fenómeno de *overfitting*

5.3.3 PREVISÃO DE RESPOSTA

O último módulo desenvolvido para o sistema YEC ficar totalmente concluído, diz respeito à tarefa de previsão de resposta por parte do modelo, tendo como base no conjunto de dados que serviu para efetuar o seu treino [112].

O melhor método de avaliação de um chatbot, é através da interacção de diferentes utilizadores com este, fornecendo cada um, *feedback* da qualidade das respostas por ele devolvidas [113].

Na verdade, a avaliação humana tem sido uma das métricas mais utilizadas para avaliação de modelos conversacionais, sendo esta baseada usualmente, em dois métodos distintos: Ou na requisição perante os diferentes utilizadores que testam o modelo, para atribuírem um valor, numa escala pré-definida, à qualidade das respostas indíviduais devolvidas por este, ou para escolherem a melhor resposta de um conjunto de respostas produzidas por diferentes modelos para a mesmo entrada [107],[112],[113].

De facto existe uma grande quantidade de elementos presentes nos modelos conversacionais, aos quais podemos atribuir um certo grau de avaliação como por exemplo a gramática, a naturalidade de resposta, a coerência, o grau de qualidade de diálogo, entre outros, tornando assim a avaliação do Chatbot mais fidigna, quando comparado com métricas numéricas, que muitas das vezes estão longe de retratar corretamente a qualidade destes sistemas [106],[114].

Para podermos avaliar o nosso modelo, revelou-se necessário efetuar o desenvolvimento do módulo final, aquele que interage com os diferentes utilizadores e que utiliza todo o treino realizado anteriormente.

Os passos de carregamento do conjunto de dados e aplicação do *tokenizer* sobre estes, apresentados no módulo anterior, repetem-se no módulo de previsão, necessários para que o modelo consiga efetuar o mapeamento das palavras presentes no seu vocabulário, com a frase fornecida pelo utilizador. Depois de efetuados os passos anteriores, o modelo com melhor desempenho durante a fase de treino é carregado, recorrendo ao modulo *load_model*, fornecido pelo módulo Keras, da maneira seguinte:

```
1.  from keras.models import load_model
2.
3.  # carregar modelo com melhor desempenho
4.  model = load_model('model1.h5')
```

A implementação do método de previsão, é fornecido pela API *Keras*, através do módulo *predict*, capaz de gerar as diversas saídas de acordo com as entradas a ele fornecidas, tendo por base a aprendizagem realizada no módulo anterior. Este processo encontra-se demonstrado na função *predict_sequence*() apresentada de seguida:

```
# gerar resposta dada uma frase de entrada
def predict_sequence(model, tokenizer, source):
    prediction = model.predict(source, verbose=0)[0]
    integers = [argmax(vector) for vector in prediction]
    target = list()
    for i in integers:
        word = word_for_id(i, tokenizer)
        if word is None:
            break
        target.append(word)
    return ' '.join(target)
```

O passo final do desenvolvimento do motor conversacional consistiu então em interligar todas as peças apresentadas anteriormente, colocando desta forma o Chatbot pronto a funcionar. Todas as interacções realizadas com este são registadas num ficheiro de texto, sofrendo este todo o pré-processamento necessário para a sua realimentação, no treino do modelo seguinte. A geração deste ficheiro ocorre com o auxílio da biblioteca *logging* implementada em Python.

Podemos então finalmente, testar o motor conversacional implementado, numa primeira fase, para o nosso Chatbot YEC:

Nos diferentes testes ao sistema, devido à limitação de vocabulário, foi definido um limite de sete interacções por cada janela conversacional, procurando explorar as qualidades deste, mas também identificar os seus pontos de melhoria. Não é expectável que perante um vocabulário tão curto, composto por aproximadamente trezentos termos, que o sistema consiga acompanhar e responder corretamente a todas as interacções fornecidas pelo utilizador. Devido ao espectro conversacional de YEC, nesta primeira fase, ser muito reduzido quando comparado com a infinidade de interacções que lhe podemos fornecer, não se irá abordar tópicos específicos conversacionais, baseando-se o seu teste, na qualidade do estabelecimento de conversa numa fase inicial, fornecendo-lhe perguntas genéricas, sendo esperado que este consiga devolver respostas estruturas e coerentes, diferentes daquelas que se encontram declaradas no conjunto de dados que serviu para o seu treino.

O fornecimento de perguntas idênticas mas sob o mesmo propósito é um dos focos de teste, sendo expectável que o sistema consiga entender a sua semelhança e capaz de devolver respostas coerentes, mas estruturalmente diferentes em ambos os casos.

A devolução da mesma resposta para ambas as perguntas, seria um alerta para a possibilidade do sistema convergir para uma determinada resposta, sendo um indicador que o treino realizado anteriormente obteve um mau desempenho.

Figura 5.14 - Exemplo de interacção genérica entre o utilizador e YEC.

Numa primeira análise e a partir do conhecimento do vocabulário implementado no nosso sistema, podemos afirmar, por observação da figura 5.14, que o treino e desenvolvimento deste foi, numa primeira fase, concluído com sucesso. É notória a concordância frásica e contextual da resposta devolvida pelo sistema, sendo este capaz de interligar diferentes termos presentes em diferentes frases contidas no *dataset*, mas que se encontram sob o mesmo contexto e intuíto.

No entanto, podemos verificar que a resposta do sistema para a última interacção fornecida pelo utilizador, constituída por apenas um termo, "nada", não se enquadra no contexto. Num sistema ideal, perante o fornecimento de um termo básico, sem significado, era expectável que o sistema procurasse por mais informação do utilizador, com o objectivo de entender o seu estado emocional.

Porém acredita-se que a devolução da resposta genérica "olá" se prenda pelo facto do treino do modelo ter sido efectuado com dados insuficientes para o sistema se comportar dessa maneira desejada, devolvendo este, a resposta mais genérica para o caso, tendo em vista o estabelecimento de nova conversa.

O teste seguinte consistiu no desenvolvimento de duas interacções distinas com YEC, mas semelhantes em termos contextuais, isto é, fornecemos ao sistema diferentes maneiras de dizer a mesma coisa, esperando que este consiga entender e devolver respostas apropriadas, ainda que formuladas de diferentes maneiras. Os testes encontram-se nas figuras 5.15 e 5.16, apresentadas de seguida.

Figura 5.15 - Primeira interacção YEC-Utilizador para efeitos comparativos. ´

Figura 5.16 - Segunda interacção YEC-Utilizador para efeitos comparativos.

Por observação das figuras cima, podemos afirmar que os resultados são bastante satisfatórios para um número de dados tão reduzido. Devido ao facto do *dataset* ter sido construído de raiz, permite-nos inferir mais facilmente se as frases devolvidas não são aquelas que se encontram declaradas nele, o que levantaria o problema de *overfitting*, levando a aproximar este Chatbot a um sistema de extracção de informação.

No entanto, verificou-se que tal não ocorre, tendo o sistema capacidade para devolver diferentes respostas, coerentes e bem estruturadas, para frases contextualmente semelhantes, diferentes daquelas que serviram para efetuar a sua aprendizagem.

CAPÍTULO 6
DISCUSSÃO E CONCLUSÕES

Este capítulo encerra a dissertação, sumarizando o conteúdo de todas as secções anteriores. Fornece uma visão global sobre o sistema desenvolvido, a sua aplicabilidade e importância, terminando com uma série de perspectivas futuras de desenvolvimento, com o objetivo de melhorar o sistema proposto.

6.1 Discussão

Após serem expostos os resultados resultantes do treino do modelo desenvolvido e os resultados referentes ás diferentes interacções realizadas ao sistema YEC, é possível afirmar que os algoritmos de *Deep Learning* são extremamente poderosos na tarefa de geração e modelação conversacional, permitindo adicionar uma verdadeira componente de inteligência ao sistema, quando comparado com os métodos mais comuns, baseados na extracção de informação de conhecimento pré-declarado.

Quanto aos resultados de treino do modelo, apesar deste ser referente a um número mais reduzido de dados quando comparado com o expéctável num sistema conversacional, foram satisfatórios para uma primeira fase de aprendizagem. Os resultados obtidos com o tipo de redes Gru foram ligeiramente superiores aos resultados obtidos com o tipo de redes LSTM, ainda que o número de dados que serviu de base para o treino não ser suficientemente grande para efetuar conclusões definitivas acerca das vantagens de utilização deste tipo redes em detrimento das redes LSTM.

Olhando agora para os resultados das interacções realizadas entre utilizador e o modelo desenvolvido, podemos dizer que os resultados são, para uma fase inicial, bastante bons, permitindo o estabelecimento de uma conversa inicial entre os utilizadores e o sistema, de maneira estruturada e coerente, desde que não sejam fornecidas interacções fora do seu espectro conversacional, não tendo este a capacidade de se comportar corretamente se tal for o caso.

É importante referir que não era expectável um funcionamento total do sistema nesta dissertação, devido ao facto de ser um exercício contínuo e de constante melhoria, servindo esta primeira abordagem como guia e comprovativo da solução proposta. O sistema desenvolvido teve sempre em vista alterações futuras, tendo sido formulado de forma a que essas alterações sejam fáceis de implementar e de interligar, entre os diferentes componentes que o constituem.

Tal como acontece nos seres humanos, também estes sistemas conversacionais são projetados para uma contínua evolução cognitiva.

6.2 CONCLUSÕES

Para concluir acerca do trabalho realizado nesta dissertação, demonstra-se relevante apresentar as respostas às questões de investigação, propostas no primeiro capítulo, na sub secção 1.3. Apesar de algumas já terem sido respondidas ao longo do documento, estas foram resumidas tendo em vista a rápida compreensão por parte do leitor.

- **Que lacunas são encontradas na construção de um Chatbot baseado em *Deep Learning*?**

Os Chatbots baseados em *Deep Learning* são um fenómeno bastante recente, ainda sujeitos a bastante investigação e ainda sem resultados perfeitos nos dias de hoje. Existem diversas adversidades a enfrentar aquando da projecção de um destes sistemas, sendo os principais problemas encontrados ao longo da realização desta dissertação:

→ A falta de dados estruturados e coerentes em grandes quantidades que possam servir para alimentar os modelos desenvolvidos.

→ A dificuldade do sistema em extraír o significado inerente a diversas frases, fenómeno recorrente na lingua portuguesa onde assistimos a muita ambiguidade gramatical – dois termos exactamente iguais têm significados totalmente distintos, o que leva o Chatbot a retornar respostas inadequadas e mal formadas.

No fundo o desenvolvimento deste tipo de Chatbot irá andar sempre de mão dada com os avanços da áreas de processamento de linguagem natural e de *Deep Learning*, onde se encontram e se esperam os desenvolvimentos que permitam uma melhor compreensão conversacional por parte de um sistema computacional.

- **Qual o propósito de desenvolver um chatbot para a área da saúde ?**

Esta questão já foi respondida ao longo da dissertação, sendo que, são inúmeras as aplicações de um Chabot na área da saúde. Apesar desta dissertação se focar no desenvolvimento de um sistema que possui comoo propósito, fornecer companhia e ajudar utilizadores que se encontrem sob distúrbios psicológicos, este poderia ser projetado de mil maneiras distintas como por exemplo para efetuar a gestão de medicamentos, a gestão de marcações hospitalares, a verificação de sintomas fornecidos pelo utilizador, um sistema clínico para questões genéricas, entre outros.

- **Será possível na prática efetuar o desenvolvimento de um Chatbot baseado em métodos de *Deep Learning*, com o objetivo deste se assumir como um companheiro virtual ?**

Sim, o Chatbot desenvolvido e apresentado nesta dissertação comprova a possibilidade de desenvolver um sistema com esse mesmo propósito. O sistema apresentado foi, numa primeira fase, capaz de interagir de forma coerente com o utilizador, acreditando-se que, através do fornecimento de mais dados conversacionais genéricos, o comportamento do sistema melhorará substancialmente. No fundo, quanto mais amplo for o vocabulário deste, melhor será a sua capacidade de geração conversacional pois maior é o número de combinações entre termos que este poderá realizar e mapear, a partir da interacção fornecida pelo utilizador.

- **Será o sistema proposto capaz de ser aceite e adaptado pelos seus utilizadores-alvo identificados?**

Esta será sempre uma pergunta ambígua e dependente do utilizador em questão. Acredita-se que para os utilizadores-alvo aqui identificados, que o sistema proposto nesta dissertação é capaz, com o passar do tempo e com a sua constante alimentação de dados conversacionais, de estabelecer um grau de confiança aceitável com os seus diferentes utilizador. O tratamento diferenciado por utilizador é visto como uma característica única deste sistema, não sendo implementado nos demais Chatbots investigados e analisados para a realização desta dissertação.

6.3 CONTRIBUIÇÕES E PERSPETIVAS FUTURAS DE INVESTIGAÇÃO

Existem inúmeras recomendações para investigações e contribuições futuras, com vista a melhoria do sistema proposto e implementado nesta dissertação, dado que esta é uma área em constante evolução e onde existe muito por descobrir.

A primeira proposta para desenvolvimento futuro, tem como base a quantidade de informação utilizada para realizar a aprendizagem do nosso modelo. O Chatbot demonstrado nesta dissertação foi limitado a um domínio relativamente pequeno, sendo necessário efetuar o teste noutro domínio de maior dimensão, de forma a verificar a escalabildidade dos resultados. A alimentação deste com mais dados textuais vai permitir a inferência da sua capacidade em se moldar a um universo bastante maior, em termos contextuais. Já foi verificado que o sistema se revela capaz de extraír corretamente o significado das frases para ele fornecidas nesta fase inicial, mas como será o seu comportamento quando o número de hipóteses e de associações for bastante mais elevado?

Esta é uma das questões fulcrais a ser respondida, reveladora da verdadeira capacidade do sistema, pois naturalmente, o que se pretende num sistema conversacional é um amplo universo de discurso, e não apenas a capacidade de resposta perante perguntas genéricas básicas.

Em segundo lugar, outra modificação a ser efetuada sobre o modelo implementado, reside na maneira de como este formula a sua resposta. A abordagem atual segue uma comportamento *greedy,* seleccionando o *token* mais provável em cada iteração da rede. Seria interessante substituir esta por um algoritmo mais complexo, como é o caso do algoritmo de *beam search*, cujo funcionamento se baseia na análise dos n *tokens* mais prováveis. Desta forma seria possível efetuar uma comparação entre os dois comportamentos, permitindo a inferência da melhor implementação para o nosso modelo conversacional.

A última proposta de contribuição futura sobre o motor conversacional implementado, reside nas técnicas de NLP aplicadas sobre o conjunto de dados, antes destes serem processados pelo modelo de *Deep Learning*. Tirando a técnica de *Word Embedding* que se encontra acoplado ao modelo implementado, na presente dissertação só foi aplicada a técnica de tokenização sobre o conjunto de dados. É sugerido combinar esta técnica com outras que possam elevar a aprendizagem do nosso modelo para outro patamar, como o caso da técnica de *PoS Tagging* ou os mecanismos de análise de questões, apresentados no terceiro capítulo desta dissertação.

Como o objetivo da dissertação era avaliar o poder da utilização de redes neuronais para criar um Chatbot de apoio clínico, a implementação prática resumiu-se ao modelo da sua capacidade de geração de resposta. No entanto, o sistema YEC proposto ao longo desta, tem outra componente igualmente importante para o seu total funcionamento: o tratamento diferenciado por utilizador. Desta forma torna-se importante o desenvolvimento prático desta componente para comprovar que o sistema é capaz de tratar cada utilizador de maneira diferente, permitindo o atingimento de diferentes graus de confiança e aprendizagem, perante estes.

REFERÊNCIAS

[1] A. M. Turing, "Machinery, Computing: 'Computing machinery and intelligence,'" *Mind*, vol. 59, no. 236, p. 433, 1950.

[2] T. M. Maina, "Instant Messaging an Effective Way of Communication in Workplace," *Murang'a Univ. Coll.*, no. 2011, 2014.

[3] J. M. Layng, "The Virtual Communication Aspect: A Critical Review of Virtual Studies Over the Last 15 Years," *J. Lit. Technol.*, vol. 17, 2016.

[4] S. Hosseini, "Chat bots," no. April, pp. 1–31, 2017.

[5] B. Bazara and M. T. Fatma, "Instant Messaging: Standards, Protocols, Applications, and Reserarch Directions," vol. 7, 2009.

[6] M. B. Hoy, "Alexa, Siri, Cortana, and More: An Introduction to Voice Assistants," *Med. Ref. Serv. Q.*, vol. 37, no. 1, pp. 81–88, Jan. 2018.

[7] R. B. Jennings, E. M. Nahum, D. P. Olshefski, D. Saha, Zon-Yin Shae, and C. Waters, "A study of internet instant messaging and chat protocols," *IEEE Netw.*, vol. 20, no. 4, pp. 16–21, Jul. 2006.

[8] "Chatbot As An Enterprise Virtual Assistant – Chatbots Magazine." [Online]. Available: https://chatbotsmagazine.com/chatbot-as-an-enterprise-virtual-assistant-1b2b1c19180f. [Accessed: 10-Oct-2018].

[9] S. E. August, "Personal Healthcare Assistant / Companion in Virtual World," *Commun. ACM*, pp. 41–42, 2006.

[10] Raconteur Independent Publication, "Future of Healthcare," vol. 44, 2016.

[11] "What Do Patients Ask Chatbots in The Healthcare Industry?" [Online]. Available: https://hackernoon.com/what-do-patients-ask-chatbots-in-the-healthcare-industry-6c12dda877ad. [Accessed: 11-Feb-2018].

[12] "Chatbots Will Serve As Health Assistants - The Medical Futurist." [Online]. Available: https://medicalfuturist.com/chatbots-health-assistants. [Accessed: 20-Nov-2017].

[13] "Incapacidade de resposta do SNS leva sector privado da saúde a crescer | Saúde | PÚBLICO." [Online]. Available: https://www.publico.pt/2017/07/10/sociedade/noticia/incapacidade-de-resposta-do-sns-leva-sector-privado-da-saude-a-crescer-1778602. [Accessed: 04-Dec-2017].

[14] "Meet The Potential of Chatbots in Healthcare." [Online]. Available: https://impacting.digital/the-potential-of-chatbots-healthcare/. [Accessed: 17-Dec-2017].

[15] B. W. Hesse *et al.*, "Trust and Sources of Health Information," *Arch. Intern. Med.*, vol. 165, no. 22, p. 2618, Dec. 2005.

[16] "research2guidance - Bots in healthcare: interview with Thomas Schulz, Organiser of Botscamp." [Online]. Available: https://research2guidance.com/bots-in-healthcare-interview-with-thomas-schulz/. [Accessed: 20-Jan-2018].

[17] A. Celikyilmaz, L. Deng, and D. Hakkani-Tür, "Deep Learning in Spoken and Text-Based Dialog Systems," in *Deep Learning in Natural Language Processing*, Singapore: Springer Singapore, 2018, pp. 49–78.

[18] J. Kříž, "Chatbot for Laundry and Dry Cleaning Service," 2017.

[19] J.-P. Sansonnet, D. Leray, and J.-C. Martin, "Architecture of a Framework for Generic Assisting Conversational Agents," Springer, Berlin, Heidelberg, 2006, pp. 145–156.

[20] M. McTear, Z. Callejas, and D. Griol, "Creating a Conversational Interface Using Chatbot Technology," in *The Conversational Interface*, Cham: Springer International Publishing, 2016, pp. 125–159.

[21] "Chatbot Report 2018: Global Trends and Analysis – Chatbots Magazine." [Online]. Available: https://chatbotsmagazine.com/chatbot-report-2018-global-trends-and-analysis-4d8bbe4d924b.

[Accessed: 10-Oct-2018].
[22] "The Rise Of Chatbots [INFOGRAPHIC]." [Online]. Available: https://www.valuewalk.com/2018/04/the-rise-of-chatbots-infographic/. [Accessed: 22-Feb-2018].
[23] Ubisend, "Mobile Messaging Report 2016," 2016.
[24] K. Nimavat and T. Champaneria, "Chatbots: An Overview Types, Architecture, Tools and Future Possibilities," *Int. J. Sci. Res. Dev.*, vol. 5, no. 7, pp. 1019–1024, 2017.
[25] "Deep Learning for Chatbots, Part 1 – Introduction – WildML." [Online]. Available: http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/. [Accessed: 05-Mar-2018].
[26] "Ultimate Guide to Leveraging NLP & Machine Learning for your Chatbot." [Online]. Available: https://chatbotslife.com/ultimate-guide-to-leveraging-nlp-machine-learning-for-you-chatbot-531ff2dd870c. [Accessed: 05-Mar-2018].
[27] "Deep Learning chatbot - analysis and implementation - Sigmoidal." [Online]. Available: https://sigmoidal.io/chatbots-for-b2c-and-deep-learning/. [Accessed: 07-Mar-2018].
[28] "Rule based bots vs AI bots – #WeCoCreate – Medium." [Online]. Available: https://medium.com/botsupply/rule-based-bots-vs-ai-bots-b60cdb786ffa. [Accessed: 08-Oct-2018].
[29] F. Guo, A. Metallinou, C. Khatri, A. Raju, A. Venkatesh, and A. Ram, "Topic-based Evaluation for Conversational Bots," Jan. 2018.
[30] "Building a Chatbot: analysis & limitations of modern platforms | Tryolabs Blog." [Online]. Available: https://tryolabs.com/blog/2017/01/25/building-a-chatbot-analysis–limitations-of-modern-platforms/. [Accessed: 12-Mar-2018].
[31] R. Yan, "'Chitty-Chitty-Chat Bot': Deep Learning for Conversational AI," pp. 5520–5526, 2018.
[32] "Building chatbots: Why message length matters." [Online]. Available: https://www.invisionapp.com/inside-design/chatbots-message-length/. [Accessed: 19-May-2018].
[33] "Chatbot Building Best Practices—Why Message Length Matters." [Online]. Available: https://uxdesign.cc/chatbot-building-best-practices-why-message-length-matters-e951bed1b550. [Accessed: 15-Mar-2018].
[34] "A Comparative Analysis of ChatBots APIs – ActiveWizards: machine learning company – Medium." [Online]. Available: https://medium.com/activewizards-machine-learning-company/a-comparative-analysis-of-chatbots-apis-f9d240263e1d. [Accessed: 17-Mar-2018].
[35] A. F. Van Woudenberg, "A Chatbot Dialogue Manager Chatbots and Dialogue Systems : A Hybrid Approach," 2014.
[36] S. J. Russell, P. Norvig, J. F. Canny, J. M. Malik, and D. D. Edwards, *Artificial Intelligence A Modern Approach*. 1995.
[37] J. Weizenbaum and Joseph, "ELIZA—a computer program for the study of natural language communication between man and machine," *Commun. ACM*, vol. 9, no. 1, pp. 36–45, Jan. 1966.
[38] H. Pruijt, "Social interaction with computers: An interpretation of Weizenbaum's ELIZA and her heritage," *Soc. Sci. Comput. Rev.*, vol. 24, no. 4, pp. 516–523, 2006.
[39] K. M. Colby, "Modeling a paranoid mind," *Behav. Brain Sci.*, vol. 4, no. 4, p. 515, Dec. 1981.
[40] "Chatbot Parry, Kenneth Mark Colby." [Online]. Available: https://www.chatbots.org/chatbot/parry/. [Accessed: 20-Jul-2018].
[41] H.-Y. Shum, X. He, and D. Li, "From Eliza to XiaoIce: Challenges and Opportunities with Social Chatbots," 2018.
[42] M. L. Mauldin, "CHATTERBOTS, TINYMUDS, and the Turing Test Entering the Loebner Prize Competition," 1994.
[43] B. AbuShawar and E. Atwell, "ALICE Chatbot: Trials and Outputs," *Comput. y Sist.*, vol. 19, no. 4,

Dec. 2015.
[44] R. Wallace, “The elements of AIML style,” 2003.
[45] R. Carpenter and J. Freeman, “Computing machinery and the individual: the personal Turing test.”
[46] S. L. Lim and O. S. Goh, “Intelligent Conversational Bot for Massive Online Open Courses (MOOCs),” *Arxiv.Org*, pp. 1689–1699, 2016.
[47] “Chatbots for Healthcare - Comparing 5 Current Applications.” [Online]. Available: https://www.techemergence.com/chatbots-for-healthcare-comparison/. [Accessed: 21-Mar-2018].
[48] “Your.MD - Health Guide and Symptom Checker.” [Online]. Available: https://www.your.md/. [Accessed: 21-Mar-2018].
[49] “The Top 12 Health Chatbots - The Medical Futurist.” [Online]. Available: https://medicalfuturist.com/top-12-health-chatbots. [Accessed: 02-Apr-2018].
[50] “Florence - Your health assistant.” [Online]. Available: https://florence.chat/. [Accessed: 02-Apr-2018].
[51] R. Kibble, “Introduction to natural language processing,” 2013.
[52] A. Branco and F. Costa, “Gramática e Processamento da Linguagem Natural: Fundamentos,” 2008.
[53] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, “Natural language processing: an introduction.,” *J. Am. Med. Inform. Assoc.*, vol. 18, no. 5, pp. 544–51, 2011.
[54] S. Do and L. Pereira, “Processamento de Linguagem Natural.”
[55] M. J. S. Pereira, “Processamento de Linguagem Natural para Produtos de Seguros,” p. 88, 2014.
[56] N. Hardeniya, M. Iti, J. Nisheeth, D. Chopra, and J. Perkins, *Natural language processing : Python and NLTK - Learn to build expert NLP and machine learning projects using NLTK and other Python libraries*. .
[57] S. WETSTEIN, “Designing a Dutch financial chatbot,” 2017.
[58] C. Michael dos Santos and D. Carlos Manuel Jorge da Silva Pereira, “Classificação de Documentos com Processamento de Linguagem Natural,” 2004.
[59] “Stemming and lemmatization.” [Online]. Available: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html. [Accessed: 07-Jul-2018].
[60] S. Kodimala, “Study of stemming algorithms,” 2010.
[61] “Information Retrieval: CHAPTER 8: STEMMING ALGORITHMS.” [Online]. Available: http://orion.lcg.ufrj.br/Dr.Dobbs/books/book5/chap08.htm. [Accessed: 10-Jul-2018].
[62] M. A. Hafer and S. F. Weiss, “Word segmentation by letter successor varieties,” *Inf. Storage Retr.*, vol. 10, no. 11–12, pp. 371–385, Nov. 1974.
[63] M. T. Dang and S. Choudri, “Simple Unsupervised Morphology Analysis Algorithm (SUMAA).”
[64] W. B. Croft, “Clustering large files of documents using the single-link method,” *J. Am. Soc. Inf. Sci.*, vol. 28, no. 6, pp. 341–344, Nov. 1977.
[65] D. Sharma, “Stemming Algorithms: A Comparative Study and their Analysis,” 2012.
[66] J. Cahn, “CHATBOT: Architecture, Design, and Development,” 2017.
[67] J. Cahn, “CHATBOT: Architecture, Design, and Development,” pp. 1–46, 2017.
[68] L. Sarmento, “Processamento de Linguagem Natural Uma necessariamente breve, muito incompleta, assumidamente parcial mas eventualmente motivadora panorâmica da área.”
[69] N. R. F. G. Cardoso, “Mineração de Texto em Literatura Biomédica,” 2008.
[70] “A Gentle Introduction to the Bag-of-Words Model.” [Online]. Available: https://machinelearningmastery.com/gentle-introduction-bag-words-model/. [Accessed: 03-Sep-2018].
[71] J. Alberto Sousa Torres, “Descoberta de Informação Através da Mineração de Texto - Fundamentos e Aplicações.,” no. February, 2012.

[72] "The TF*IDF Algorithm Explained | Elephate." [Online]. Available: https://www.elephate.com/blog/what-is-tf-idf/. [Accessed: 10-Sep-2018].
[73] A. Berger, R. Caruana, D. Cohn, D. Freitag, and V. Mittal, "Bridging the lexical chasm: statistical approaches to answer-finding," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR '00*, 2000, pp. 192–199.
[74] S. C. S. Pinto, "Processamento de Linguagem Natural e Extração de Conhecimento," *Process. Ling. Nat. e Extração Conhecimento*, p. 157, 2015.
[75] D. Jurafsky and J. H. Martin, "Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition Third Edition draft."
[76] R. Collobert, J. Weston, J. Com, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural Language Processing (Almost) from Scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011.
[77] E. Agirre and P. Edmonds, *Word Sense Disambiguation Algorithms and Applications*. 2007.
[78] D. Laham, P. W.Foltz, and T. K Landauer, "Introduction to Latent Semantic Analysis," 1998.
[79] S. He, Z. Li, H. Zhao, H. Bai, and G. Liu, "Syntax for Semantic Role Labeling, To Be, Or Not To Be," Association for Computational Linguistics.
[80] R. P. Talhadas dos Santos, "Automatic Semantic Role Labeling for European Portuguese," 2014.
[81] A. Lally *et al.*, "Question analysis: How Watson reads a clue," *IBM J. Res. Dev.*, vol. 56, no. 3.4, p. 2:1-2:14, 2012.
[82] P. Cortez and J. Neves, "Redes Neuronais Artificiais," 2000.
[83] V. T. Cortez *et al.*, "INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA APLICAÇÃO DE REDES NEURONAIS ARTIFICIAIS À DETEÇÃO E ISOLAMENTO DE FALHAS EM PROCESSOS INDUSTRIAIS," 2015.
[84] "Deep Learning: Feedforward Neural Network – Towards Data Science." [Online]. Available: https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7. [Accessed: 30-Aug-2018].
[85] E. Snelleman, "Decoding neural machine translation using gradient descent," 2016.
[86] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," *Nips*, p. 9, 2014.
[87] "Difference Between Supervised, Unsupervised, & Reinforcement Learning | NVIDIA Blog." [Online]. Available: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/. [Accessed: 02-Sep-2018].
[88] D. Saad, B. Birmingham, and S. A. Solla, "Dynamics of On-Line Gradient Descent Learning for Multilayer Neural Networks," *Adv. Neural Inf. Process. Syst. 8*, pp. 302–308, 1996.
[89] A. T. C. Goh, "Back-propagation neural networks for modeling complex systems," *Artif. Intell. Eng.*, vol. 9, no. 3, pp. 143–151, Jan. 1995.
[90] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A Critical Review of Recurrent Neural Networks for Sequence Learning," pp. 1–38, 2015.
[91] "Understanding LSTM Networks – colah's blog." [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. [Accessed: 19-Aug-2018].
[92] "Recurrent Neural Networks and LSTM – Towards Data Science." [Online]. Available: https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5. [Accessed: 17-Aug-2018].
[93] "Essentials of Deep Learning : Introduction to Long Short Term Memory." [Online]. Available: https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/. [Accessed: 22-Aug-2018].
[94] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated Feedback Recurrent Neural Networks," 2015.

[95] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Gated Feedback Recurrent Neural Networks,” 2015.
[96] “Why One-Hot Encode Data in Machine Learning?” [Online]. Available: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/. [Accessed: 02-Oct-2018].
[97] “What is One Hot Encoding and How to Do It – Michael DelSole – Medium.” [Online]. Available: https://medium.com/@michaeldelsole/what-is-one-hot-encoding-and-how-to-do-it-f0ae272f1179. [Accessed: 02-Oct-2018].
[98] “Why One-Hot Encode Data in Machine Learning?” .
[99] “What is One Hot Encoding and How to Do It – Michael DelSole – Medium.” .
[100] “The amazing power of word vectors | the morning paper.” [Online]. Available: https://blog.acolyer.org/2016/04/21/the-amazing-power-of-word-vectors/. [Accessed: 05-Oct-2018].
[101] “Introduction to Word Vectors – Jayesh Bapu Ahire – Medium.” [Online]. Available: https://medium.com/@jayeshbahire/introduction-to-word-vectors-ea1d4e4b84bf. [Accessed: 06-Oct-2018].
[102] A. STRIGÉR, “End-to-End Trainable Chatbot for Restaurant Recommendations,” *Kth R. Inst. Technol. Sch. Comput. Sci. Commun.*, 2017.
[103] “IBM Knowledge Center - OAuth 2.0 support.” [Online]. Available: https://www.ibm.com/support/knowledgecenter/zh/SSELE6_8.0.1.3/com.ibm.isam.doc/config/concept/oauth.html. [Accessed: 10-May-2018].
[104] P. Identity, “The Essential OAuth Primer: Understanding OAuth for Securing Cloud APIs,” *Ping Identity*, pp. 1–15, 2011.
[105] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” Dec. 2014.
[106] R. K. Csaky, “Deep Learning Based Chatbot Models,” 2017.
[107] O. Vinyals and Q. Le, “A Neural Conversational Model,” Jun. 2015.
[108] I. V. Serban *et al.*, “A Hierarchical Latent Variable Encoder-Decoder Model for Generating Dialogues,” May 2016.
[109] J. Li, M. Galley, C. Brockett, G. P. Spithourakis, J. Gao, and B. Dolan, “A Persona-Based Neural Conversation Model,” Mar. 2016.
[110] M. Neishi, J. Sakuma, S. Tohda, S. Ishiwatari, N. Yoshinaga, and M. Toyoda, “A Bag of Useful Tricks for Practical Neural Machine Translation: Embedding Layer Initialization and Large Batch Size,” *Proc. 4th Work. Asian Transl.*, pp. 99–109, 2017.
[111] R. Dey and F. M. Salemt, “Gate-variants of Gated Recurrent Unit (GRU) neural networks,” *Midwest Symp. Circuits Syst.*, vol. 2017–Augus, pp. 1597–1600, 2017.
[112] L. Shao, S. Gouws, D. Britz, A. Goldie, B. Strope, and R. Kurzweil, “Generating High-Quality and Informative Conversation Responses with Sequence-to-Sequence Models,” Jan. 2017.
[113] H. Zhou, M. Huang, T. Zhang, X. Zhu, and B. Liu, “Emotional Chatting Machine: Emotional Conversation Generation with Internal and External Memory,” Apr. 2017.
[114] T. Zhao, A. Lu, K. Lee, and M. Eskenazi, “Generative Encoder-Decoder Models for Task-Oriented Spoken Dialog Systems with Chatting Capability,” Jun. 2017.