Mestrado em Informática e Sistemas

FAQBot.PT: Sistema de conversação e resposta a perguntas em português

Trabalho de Projeto apresentado para obtenção de grau de Mestre em Informática e Sistemas
Especialização em Desenvolvimento de Software

Autor
Ricardo Ferreira Filipe

Orientadores
Professora Ana Oliveira Alves
Professora do Departamento de Informática e Sistemas
Instituto Superior de Engenharia de Coimbra

Professor Hugo Gonçalo Oliveira
Professor do Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

Coimbra, maio de 2019

Abstract

In last years, there have been an increasing number of conversational systems called chatbots, these systems have the purpose of imitating a human conversation. This increasing is due to two factors: firstly because of the advances in technology of Natural Language Processing (NLP) and secondly to the use of these systems by large companies like Facebook, Google e Microsoft. More and more small businesses see the benefits of these systems to help them with small tasks such as explaining doubts of their customers while their employees focus on issues more important to the company. This study intends to analyze the most used tools in the market for the creation of chatbots in order to be able to create our own chatbot capable to deal with issues written in portuguese and in this way capable of help business in tasks related to the doubts of its clients. This system intends to receive as a base for extracting data a document of Frequently Asked Questions (FAQ). It is intended that this system be as flexible as possible in order to be reused by any company being only necessary to change the document of frequently asked questions used.

Keywords: Conversational system, chatbot, frequently asked questions, natural language processing, information retrieval, platforms and tools of chatbots.

Resumo

Nos últimos anos tem se verificado um crescimento no número de sistemas conversacionais, normalmente chamados de chatbots, estes sistemas têm o propósito de imitar uma conversação humana. Este crescimento deve-se a dois fatores: primeiramente aos avanços nas tecnologias de Processamento da Linguagem Natural (PLN) e ao uso destes sistemas por grandes empresas como Facebook, Google e Microsoft. Cada vez mais pequenas empresas vêem os benefícios destes sistemas para os ajudar com pequenas tarefas como por exemplo as dúvidas dos seus clientes enquanto os seus funcionários focam-se em questões mais importantes para a empresa. Com este estudo pretende-se analisar as ferramentas mais utilizadas no mercado para a criação de sistemas de conversação a fim de conseguir criar o nosso próprio sistema de respostas automáticas a perguntas frequentes capaz de lidar com questões escritas na língua portuguesa e desta forma conseguir ajudar as empresas em tarefas relacionadas com as dúvidas dos seus clientes. Este sistema pretende receber um documento de perguntas e respostas frequentemente feitas pelos seus clientes, normalmente chamada de FAQs (*Frequently Asked Questions*). Pretende-se que este sistema seja o mais flexível possível de forma a conseguir ser reutilizado por qualquer empresa sendo apenas necessário alterar o documento de perguntas e respostas utilizado.

Palavras-chave: Sistema conversacional, perguntas e respostas frequentes, processamento de linguagem natural, recuperação de informação, resposta automática a perguntas, sistemas de perguntas e respostas, plataformas e ferramentas de sistemas conversacionais.

Agradecimentos

Gostaria de expressar o meu mais profundo agradecimento aos orientadores, Ana Oliveira Alves e Hugo Gonçalo Oliveira, pelo apoio prestado, pela sua disponibilidade durante todo o projeto e bom humor em todas as reuniões.

Aos restantes professores do ISEC que partilharam o seu conhecimento e me inspiraram a aprofundar o meu.

Aos meus pais e irmão pelo apoio incondicional dado ao longo de todo o meu percurso académico.

Conteúdo

Lista de Figuras

Lista de Tabelas

Acrónimos

CLN	Compreensão da Linguagem Natural
FAQ	*Frequently Asked Questions*
IDE	*Integrated Development Environment*
IA	Inteligência Artificial
IDF	Inverso da Frequência nos Documentos
ML	*Machine Learning*
NLI	*Natural Language Interface*
ODQA	*Open Domain Question Answering*
PDF	*Portable Document Format*
PI	*Paraphrase identification*
PLN	Processamento de Linguagem Natural
QA	*Question Answering*
RI	Recuperação de Informação
RAP	Resposta Automática a Perguntas
RDQA	*Restricted Domain Question Answering*
SDK	*Software Development Kit*
SPR	Sistemas de Pergunta-Resposta
STC	*Short Text Conversation*
TF	Frequência do Termo

Capítulo 1

Introdução

Conversas entre um humano e um computador são um dos problemas mais difíceis no âmbito da Inteligência Artificial (IA), uma vez que envolvem a compreensão da linguagem natural e o uso do conhecimento de senso comum. Apesar das constantes evoluções tecnológicas, o avanço neste campo não é tão grande como o desejado, devido às grandes dificuldades linguísticas, como ambiguidade de certas palavras, ou à variabilidade das mesmas. Pode-se assumir que esta variabilidade vem de fatores regionais, de dialetos sociais e varia também de indivíduo para indivíduo de acordo com a situação e ambiente em que o mesmo a usa (Risso e Sant'Ana, 1973). Para além destas dificuldades, os autores Ji, Lu e Li (2014) acrescentam ainda que este atraso deve-se à falta de uma base de conhecimento com dados de qualidade capaz de ser utilizada nestes sistemas de conversação. Segundo os mesmos esta base de conhecimento deve ter os dados todos bem estruturados e organizados e acima de tudo bem etiquetados.

Para um ser humano, a forma mais natural de interagir com outros humanos é utilizando a linguagem natural, seja esta interação feita de forma escrita ou por voz. Segundo esta ideia, grandes empresas tecnológicas como Google[1], Facebook[2], Apple[3], Microsoft[4] e Amazon[5] começaram a desenvolver os seus próprios sistemas conversacionais.

Nos últimos anos verificou-se um grande crescimento no número de sistemas conversacionais, estes são normalmente chamados de *Chatbots*. Segundo Braun et al. (2017), existem muitas razões para este crescimento, nomeadamente o aumento dos serviços de *chat*, como o Facebook Menssenger[6], Slack[7], Telegram[8], e também pelo grande número de plataformas que disponibilizam o seu software para Compreensão de Linguagem Natural - CLN (em inglês, *Natural Language Understanding* – NLU), muitas vezes de forma gratuita.

O Processamento de Linguagem Natural (PLN) é uma abordagem para analisar textos utilizando técnicas computacionais com o objetivo de conseguir entender o significado e a intenção de cada um (Liddy, 2001). Para um sistema de Recuperação de Informação (RI) baseado em PLN, este tem o objetivo de fornecer a informação

[1] `https://www.google.com/about/` (Dezembro 2018)
[2] `https://www.facebook.com/` (Dezembro 2018)
[3] `https://www.apple.com/` (Dezembro 2018)
[4] `https://www.microsoft.com/` (Dezembro 2018)
[5] `https://www.amazon.com/` (Dezembro 2018)
[6] `https://www.messenger.com/` (Dezembro 2018)
[7] `https://slack.com/` (Dezembro 2018)
[8] `https://telegram.org/` (Dezembro 2018)

mais precisa e completa ao utilizador, conforme as necessidades do mesmo e analisando a informação que este fornece ao sistema.

Existem muitas dificuldades em PLN, destacando-se a ambiguidade e a variabilidade linguística, como já foi anteriormente falado. Uma forma de contornar estas dificuldades passa por associar uma representação formal para a linguagem natural, que capture o significado e ao mesmo tempo permita a interpretação pelo computador. Assumindo que para compreender o contexto de uma frase basta conseguir perceber algumas das palavras-chaves, estas dificuldades serão bastante minimizadas ao criar um módulo de Compreensão de Linguagem Natural (CLN).

O objetivo deste trabalho passa pela criação de um sistema de conversação que suporte interações na língua portuguesa. Este sistema terá de conseguir responder a perguntas num domínio específico, mas, ao mesmo tempo, deverá ter uma arquitetura o mais genérica possível, de forma a que caso se queira alterar o domínio, o sistema consiga continuar a responder a perguntas de forma ininterrupta.

Pretende-se que a base de conhecimento deste sistema seja representada através de um documento com perguntas e respetivas respostas. Documentos deste género podem ser obtidos, por exemplo, a partir de listas de perguntas frequentes (em inglês, *Frequently Asked Questions* – FAQs).

Este trabalho não estaria completo sem em primeiro, se analisar as plataformas atuais que ajudam a criar agentes conversacionais, como é o caso do DialogFlow[9], Wit.ai[10], Luis[11], IBM Watson[12] e Amazon Lex[13] explicadas em detalhe no capítulo 3. Devido à sua aparente facilidade de utilização, estas plataformas apresentam-se como um bom ponto de partida para a criação de um sistema conversacional. Foi por isso analisada cada plataforma quanto à sua flexibilidade, quanto ao objetivo pretendido e a sua capacidade para lidar com o a língua portuguesa.

Este estudo prévio permitiu perceber até que ponto faria sentido desenvolver o sistema a partir de uma das plataformas disponíveis ou implementá-lo de raiz considerando para isso a experiência dos orientadores no campo do PLN e reutilizando outras ferramentas e recursos desenvolvidos previamente, nomeadamente para o português.

Sistematizando, os objetivos gerais deste trabalho são os seguintes:

- Estudo das melhores formas de implementar um sistema conversacional, incluindo para isso um estudo das plataformas e de outras bibliotecas existentes.
- Desenvolvimento de um sistema de respostas automáticas a perguntas frequentes que suporte interações com o português, com base numa FAQ.
- Criação do sistema com uma arquitetura flexível, em que seja possível alterar o domínio do sistema de respostas automáticas e mesmo assim continue o seu normal funcionamento, idealmente com base em ferramentas já existentes.

[9] `http://dialogflow.io/` (Dezembro 2018)
[10] `https://wit.ai/` (Dezembro 2018)
[11] `https://www.luis.ai/` (Dezembro 2018)
[12] `https://www.ibm.com/watson/` (Dezembro 2018)
[13] `https://aws.amazon.com/pt/lex/` (Dezembro 2018)

A restante estrutura deste documento organiza-se da seguinte forma:

No capítulo 2 é apresentado o estudo do estado da arte para este trabalho. Divide-se em duas partes: a primeira dedicada aos estudos na área dos sistemas conversacionais e a segunda parte apresentar os estudos relacionados com o tipo de sistema que se pretende desenvolver, sistemas relacionados com a resposta automática a perguntas (RAP) num ambiente em que se conhece o domínio em que o sistema irá trabalhar e com abordagens para lidar com conversas curtas (*Short Text Conversation* - STC).

No capítulo 3 são apresentadas ferramentas de suporte à criação de agentes conversacionais. Divide-se em três subcapítulos: o primeiro é dedicado às plataformas mais conhecidas para a criação de sistemas conversacionais, o segundo apresenta uma análise de *standards* para representar a base de conhecimento e fazer a gestão dos diálogos nos mesmos sistemas, e o terceiro analisa o Lucene uma ferramenta de recuperação de informação de textos, que acabou por ficar com a responsabilidade da criação do nosso sistema.

No capítulo 4 é apresentado o estudo do sistema que se pretende desenvolver. Este divide-se em cinco subcapítulos, sendo o primeiro a análise de requisitos, o segundo sobre a ferramenta que será utilizada para desenvolver o sistema conversacional, o terceiro subcapítulo uma apresentação da arquitetura que o sistema deverá seguir, seguido da explicação da preparação do ambiente de desenvolvimento e, por último, detalham-se os recursos utilizados pelo sistema.

No capítulo 5 é apresentado o trabalho desenvolvido. Primeiro são apresentados os dados em que o sistema foi aplicado e a entidade de onde foram recolhidos. Segue-se uma apresentação de várias versões do sistema, construídas como prova de conceito em comparação com outras ferramentas, até ao projeto final. O último subcapítulo apresenta os testes realizados ao sistema.

Para finalizar, o capítulo 6 resume as conclusões deste trabalho. Inicialmente, é apresentada uma visão geral deste projeto, passando por analisar os objetivos concluídos, são apresentadas as principais contribuições do projeto. Posteriormente são apresentadas as limitações e problemas encontrados no projeto e, para finalizar, são dadas orientações para trabalhos futuros.

Capítulo 2

Estado de Arte

Neste capítulo são apresentados e analisados alguns dos conteúdos e trabalhos de investigação nas áreas de relevo para o trabalho. Na primeira secção começa-se por explicar a origem e evolução dos sistemas de conversação e por fim são apresentados estudos relacionados ao trabalho que se pretende realizar.

2.1 Contextualização

Segundo Jurafsky e Martin (2009) a linguagem é a marca da humanidade e o diálogo é a mais fundamental área da linguagem.

Muito antes do termo *Chatbot* ser usado, Turing (1950) descreveu-o como um teste para a Inteligência Artificial (IA) que envolvia um diálogo entre um ser humano e um computador. Nos últimos anos verificamos um grande crescimento no número de agentes conversacionais em todas as áreas. Segundo Braun et al. (2017), existem várias razões para este aumento, nomeadamente os avanços em *Machine Learning* (ML) e o grande número de plataformas que disponibilizam o seu software de Processamento de Linguagem Natural (PLN) de forma gratuita, o que ajuda os programadores a criar o seu próprio *chatbot*. Exemplos dessas plataformas são o DialogFlow[1], Wit.ai[2], Luis[3], IBM Watson[4], entre outros que serão analisadas no capítulo 3.

Segundo Shawar e Atwell (2007), um sistema de conversação é definido como um agente que interage com o utilizador, turno por turno, usando linguagem natural. Neste cenário, um turno consiste numa interação por meio de um dos intervenientes, podendo este ser uma afirmação, uma pergunta do utilizador ou uma resposta do sistema a uma pergunta feita anteriormente.

Desde a década de 1960 que um grande número de sistemas conversacionais foram desenvolvidos. Os exemplos mais conhecidos são ELIZA (Weizenbaum, 1966), PARRY (Colby, 1973), CONVERSE (Batacharia et al., 1999) e ALICE (Wallace, 2009). Mais recentemente, os avanços tecnológicos deram origem a novos sistemas, que começaram a ser utilizados em várias áreas, com domínios muitos diferentes, como apoio ao cliente, educação ou saúde (Dutta, 2017).

Segundo Heller et al. (2005), os chatbots são sistemas programados para imitar conversas humanas. O ELIZA (Weizenbaum, 1966) é normalmente considerado o

[1] `http://dialogflow.io/` (Dezembro 2018)
[2] `https://wit.ai/` (Dezembro 2018)
[3] `https://www.luis.ai/` (Dezembro 2018)
[4] `https://www.ibm.com/watson/` (Dezembro 2018)

primeiro chatbot. Foi criado para emular uma abordagem psicoterapêutica desenvolvida por Carl Ransom Rogers, cujo objetivo deste seria a libertação da personalidade do paciente. O seu funcionamento é baseado em padrões de decisão que dão a ilusão de entendimento por parte do sistema. Mais especificamente, o sistema é programado para reconhecer palavras-chave específicas e, a partir daí, escolher a resposta mais apropriada a dar.

Para entender melhor o sistema ELIZA, este usa correspondência por padrões para reconhecer palavras chave de uma frase como "*You are* ***X***" e responder por exemplo com "*What makes you think i am* ***X***". Este tipo de abordagem é eficaz para criar sistemas de domínio aberto, uma vez que estes não precisam de ter conhecimentos sobre um domínio especifico. São sistemas que "não sabem nada" sobre o mundo à sua volta.

Anos mais tarde o sistema de conversação ALICE (*Artificial Linguistic Internet Computer Entity*) foi apresentado. Inicialmente desenvolvido por Richard Wallace em 1995(Heller et al., 2005), ALICE tinha uma base de conhecimento constituída por conversas em inglês e guardada em ficheiros no formato *standard* AIML ou *Artificial Intelligence Mark-up Language*, que é um derivado do XML (*Extensible Mark-up Language*).

Uma das competições mais conhecidas para a análise destes sistemas de conversação é a competição Loebner[5], onde todos os sistemas são submetidos ao teste de Turing. Este teste analisa a capacidade de um programa exibir um comportamento de um humano, parecendo assim ao utilizador que não está a falar com uma máquina. Apesar de nenhum dos sistemas ter conseguido passar neste teste, todos os anos é atribuído um prémio para o sistema que conseguiu a melhor performance, no Anexo B é possível ver os vencedores deste prémio assim como as técnicas utilizadas.

Segundo Ji, Lu e Li (2014), os avanços neste campo não são tão grandes como o desejado, devido às grandes dificuldades linguísticas, como ambiguidade de certas palavras, ou à variabilidade da mesma, mas também por devido à falta de uma base de conhecimento de qualidade, com os dados bem estruturados e organizados e cima de tudo bem etiquetados. Para dar a volta a este problema, os mesmos autores utilizam apenas "conversas curtas", normalmente estas conversas são mais objetivas e as respostas focadas à pergunta, não divagando para outros temas para além do que é perguntado. Ainda segundo os mesmos autores exemplos de "conversas curtas" são fáceis de obter a partir de redes sociais, como o Twitter[6] e Weibo[7].

Apesar da grande quantidade de sistemas conversacionais atualmente criados, nem todas as organizações precisam de um sistema conversacional capaz de dar uma resposta a tudo o que lhes é perguntado. Muitas precisam de um sistema inteligente treinado num campo específico da área, ou seja, um sistema conversacional especializado. O problema destes sistemas é que muitas vezes têm de ser construídos de raiz e os dados que vão fundamentar a sua base de conhecimento muitas vezes precisam de ser reconstruídos, o que leva a novos custos para a organização (Yu et al., 2018).

[5] `https://www.aisb.org.uk/events/loebner-prize` (Dezembro 2018)
[6] `https://twitter.com/` (Dezembro 2018)
[7] `https://www.whatsonweibo.com/sinaweibo/`(Dezembro 2018)

Muitas empresas pretendem adquirir um chatbot para ajudar no apoio ao cliente, mas os dados que são utilizados para a criação da base de conhecimento têm a origem em fóruns. Para estes sistemas os dados que fundamentam a base de conhecimento são muito importantes para a construção de um chatbot específico e devem estar bem estruturados, mas o problema é que dados retirados de fóruns contêm perguntas e respostas que muitas vezes não estão no formato de diálogo, podendo ser algumas perguntas muito complexas levando a ruído na extração, ou a respostas que não são bem estruturadas ou não inteiramente focadas na pergunta. Por isso, os documentos fornecidos para a criação deste tipo de sistema devem ser analisados com cuidado. Outro grande problema é a grande quantidade de dados necessária para construir o chatbot e da qualidade dos pares pergunta-resposta. Uma solução a este problema seria a de crowdsourcing, mas esta solução teria novos custos para a empresa, uma vez que seria necessário um grande volume de pares perguntas-respostas para alimentar um sistema do género.

Hoje em dia há sistemas conversacionais mais sofisticados. Eles podem marcar vôos ou reservar um restaurante entre outras tarefas, mas a simples correspondência por padrões do sistema ELIZA teve um papel crucial no desenvolvimento destes novos sistemas.

Devido ao crescimento dos dados disponíveis na Internet, uma simples pesquisa pode levar-nos a centenas de documentos, por isso um sistema que nos consiga dar uma resposta direta e curta dado um problema é cada vez mais necessário. Exemplos destes sistemas são os agentes conversacionais orientados a tarefas (em inglês, *task-oriented dialog agents*), que são sistemas construídos com o intuito de ajudar os utilizadores com uma tarefa. Outro fator que influenciará o aumento da utilização destes sistemas é a evolução do uso dos *smartphones* para aceder à informação. Uma vez que o uso do texto não será a forma mais rápida de conversar, estes sistemas deverão ser capazes de reconhecer a fala do utilizador.

Sistemas conversacionais orientados a tarefas como foi apresentado anteriormente são sistemas implementados para permitir uma compreensão da linguagem natural (CLN) num tópico ou domínio restrito, de forma a conseguir ajudar o utilizador na realização de uma tarefa. Estes sistemas diferem dos chamados chatbots no sentido em que estes últimos cobrem uma grande variedade de tópicos ou domínios. No entanto, de acordo com Niculescu e Banchs (2015), estes últimos têm uma compreensão da linguagem natural limitada.

Para finalizar esta secção é apresentado um dos recursos utilizados nas tarefas de PLN nestes sistemas conversacionais que complementam a base de conhecimento, sendo estes as wordnets, estas são bases de dados de conhecimento linguístico amplamente utilizadas. Apesar de existirem muitas wordnets, a primeira criada para a língua portuguesa não está disponível de forma gratuita para ser utilizada pela comunidade e devido a esta lacuna, outras foram criadas para o português e disponíveis para download. Contudo, estas foram desenvolvidas por diferentes equipas, seguindo diversas abordagens o que resultou em wordnets com coberturas distintas, que nem sempre seguem os modelos standard do WordNet (Gonçalo Oliveira, 2018).

2.2 Estudos Relacionados

Uma das tarefas a incluir para a criação de um chatbot é a tarefa de Resposta Automática a Perguntas (RAP), onde o objetivo desta tarefa passa por recorrer a uma coleção de documentos ou uma base de conhecimento para encontrar, de forma automática, a resposta a ser dada ao utilizador. Para isso é necessário nestes sistemas uma forma de Recuperação de Informação (RI) que tenha como objetivo o de encontrar automaticamente a informação crucial. Normalmente é possível encontrar essa informação numa pergunta feita pelo utilizador, procurando por palavras-chaves e assim conseguindo dar uma resposta o mais acertada possível.

Se pensarmos que para responder a uma pesquisa de um utilizador, apenas uma parte específica da informação da pesquisa é necessária, como por exemplo analisando as palavras-chave, desta forma será razoavelmente mais fácil criar um sistema conversacional capaz de o ajudar nas suas pesquisas. É aqui que entra o campo e RI, este é um estudo dos métodos capaz de fornecer ao utilizador a informação necessária proveniente varias fontes, pesquisando nestas pela informação essencial para entender e responder ás necessidades propostas pelo utilizador. Estas fontes podem ter varias formas, mas para este trabalho vamos nos focar em informação derivadas de textos.

Segundo Quarteroni e Manandhar (2009), os sistemas de Resposta Automática a Perguntas (RAP, em inglês *Question Answering* – QA) podem ser vistos como sistemas de RI que têm como alvo responder a perguntas feitas em linguagem natural e devolver respostas apropriadas, em vez de indicar documentos completos.

Segundo Yu et al. (2018), os sistemas de RAP podem ser separados em dois tipos: sistemas baseados em RI (Quarteroni e Manandhar, 2009) e sistemas que se focam em gerar informação. Estes autores focam-se apenas nos sistemas baseados em RI, explicando que a componente-chave deste modelo é a classificação atribuída a cada par pergunta-resposta da base de conhecimento relativamente à pergunta feita pelo utilizador. Segundo Yu et al. (2018), estes sistemas têm sido extensivamente estudados na última década e apresentados em muitos artigos, no entanto, e apesar desses estudos, há ainda muitas dificuldades na criação de sistemas conversacionais baseados em pergunta-resposta. Para além das dificuldades da variabilidade linguística, outra apresentada pelos autores é a falta de dados anotados na base de conhecimento e o custo associado à sua anotação manual. Estes explicam que todos estes estudos sobre o tema têm como ponto de partida uma grande base de conhecimento com os dados anotados, o que na realidade não acontece, pois normalmente os dados extraídos para a criação das bases de conhecimento decorrem de fóruns ou de documentos encontrados na Internet sobre o domínio. Outro grande problema é a eficiência da procura das perguntas: muitos métodos focam-se em encontrar a melhor resposta sem se preocuparem com a eficiência do método. No entanto, numa grande base de conhecimento de perguntas e respostas, a procura poderá ser algo morosa.

Segundo Kolomiyets e Moens (2011), um sistema de RAP pode ser considerado como um sistema de RI mais sofisticado, onde apenas são retornados ao utilizador pequenas frações de informação específica consideradas relevantes para ajudar o utilizador a concluir as suas tarefas. Exemplos dessas frações de informação podem ser uma frase, um parágrafo, uma imagem, um fragmento de áudio ou uma única

palavra, em oposição aos sistemas clássicos de RI onde longos documentos eram considerados relevantes e apresentados ao utilizador. Segundo os mesmos autores, estes sistemas de RAP podem ter na sua base de dados conjuntos de dados estruturados e não estruturados. Exemplos de dados estruturados são, por exemplo, dados em bases de dados relacionais, onde os objetos têm atributos bem definidos, por outro lado, dados não estruturados são textos, discursos, imagens, áudio e vídeos.

Existem ainda duas outras categorias nas quais podemos classificar sistemas de RAP: os focados num domínio específico (*Restricted Domain Question Answering* – RDQA) e os de domínio aberto (*Open Domain Question Answering* - ODQA). Os primeiros são desenhados para responder a perguntas feitas dentro de um domínio e as respostas dadas são extraídas de uma base de conhecimento com dados relativos a esse domínio. O segundo tipo de sistema é capaz de responder a questões sobre qualquer domínio, sendo as respostas encontradas numa grande base de dados ou muitas vezes baseadas em pesquisas na Internet. Os últimos sistemas tendem a dar respostas mais vagas que os primeiros (Kolomiyets e Moens, 2011). Tendo em conta estes dois tipos sistemas, o sistema que melhor se adequa ao que se pretende desenvolver, será um sistema RDQA, onde o domínio no qual o sistema atuará será bem conhecido.

Mesmo que um chatbot tenha uma base de conhecimento abrangente para responder a todas as perguntas de um determinado domínio, normalmente os utilizadores têm a tendência de fazer perguntas fora desse domínio, o que muitas vezes deixa o chatbot sem resposta e consequentemente desaponta os utilizadores. Para responder a este problema existem duas abordagens: a primeira consiste em responder ao utilizador que não existe uma resposta a dar à sua pergunta; ou na segunda abordagem, utilizar um gerador de respostas onde será procurada a resposta noutro local, podendo este ser uma grande base de dados ou o resultado de uma pesquisa na Internet (Ameixa et al., 2014).

Uma das formas utilizadas por Pedro Fialho e Trancoso (2013) para aumentar o leque de possíveis correspondências entre as perguntas feitas pelo utilizador e os dados na base de conhecimento, foi criar uma lista independente de sinónimos, com entradas como "palácio é um sinónimo de castelo". Assim, se o sistema não conseguir reconhecer a pergunta comparando-a com os textos na base de conhecimento, será feita uma alteração na pergunta do utilizador com os sinónimos e feita uma nova pesquisa na base de conhecimento, criando assim uma maior abrangência às perguntas feitas pelo utilizador.

O sistema de respostas a perguntas frequentes que se pretende criar é um "sistema orientado a uma tarefa" (Task-oriented system) que utiliza a tarefa de RAP, com a finalidade de ter conversas curtas com o utilizador, procurando pela resposta a dar ao utilizador de uma forma automática numa base de conhecimento previamente extraída de uma lista de FAQs.

Neste contexto desenvolver um sistema de conversas curtas ou *Short Text Conversation* (STC), como o que se pretende criar, é, de uma forma geral, mais simples de se implementar, uma vez que não é necessário lidar com um grande número de turnos. Normalmente estes sistemas apenas têm que lidar com um único turno e não precisam compreender o contexto da conversa. Apesar de tudo, são essencialmente sistemas deste tipo que encontramos nos *smartphones* e assistentes por voz, como a

Siri, da Apple (Ji, Lu e Li, 2014).

Uma abordagem à criação de um sistema conversacional pode-se basear em RI e numa vasta base de dados de pequenas conversas, em que a base de dados está organizada com a pergunta seguida da resposta, em que para dado um texto do utilizador t_u, o sistema procura na sua base de dados o texto t_b mais parecido com t_u e, de seguida, apresenta o texto t_{b+1} que aparece em sequência a t_b. Ou seja, não é gerado um novo texto é apenas apresentado um texto que pode seguir-se a um texto próximo do inserido.

O termo conversas curtas ou *Short Text Conversation* (STC) é definido como conversas com apenas um turno, em que cada turno consiste numa primeira mensagem de um humano e com uma resposta por um computador. Juntando RI a uma base de dados de STC, poderemos criar um sistema de respostas denominado como sistema *retrieval-based STC*. Sendo assim, dada uma pergunta feita pelo utilizador, o sistema procurará na sua base de conhecimento pela resposta cuja pergunta seja mais semanticamente parecida e de seguida irá apresentá-la ao utilizador (Ji, Lu e Li, 2014; Zhao Yan, 2016).

Capítulo 3

Ferramentas de suporte à criação de agentes conversacionais

Sistemas conversacionais são programas capazes de imitar conversas entre humanos. Estes têm como entrada frases, possivelmente perguntas, em linguagem natural e o seu objetivo é o de conseguir dar uma resposta ao utilizador. Este processo é repetido até que este consiga ajudar o utilizador na realização do seu objetivo.

Quando estes sistemas conversacionais apareceram, para se conseguir o mínimo de realismo nas conversas apresentadas pelo sistema era necessário que o programador tivesse experiência para o seu desenvolvimento, tanto ao nível de programação como ao nível de técnicas de Compreensão da Linguagem Natural (CLN). Além da complexidade do sistema criado, muitas vezes era difícil de fazer a sua manutenção. Outro ponto chave para um sistema conversacional é a sua base de conhecimento, o local onde todas as perguntas e respostas são guardadas e mapeadas consoante o seu significado (Abdul-Kader e Woods, 2015).

Atualmente existem muitas plataformas, ferramentas e arquiteturas bem estudadas para a criação de um sistema conversacional. Algumas requerem pouco conhecimento de programação e mesmo assim permitem criar chatbots robustos capazes de ajudar o utilizador a concretizar o seu objetivo. Outras oferecem a possibilidade de utilizar uma API, podendo assim haver algum nível de controlo sobre o sistema. Algumas destas plataformas são pagas devido ao nível de funcionalidades que fornecem, e a maioria oferece a possibilidade de utilizar *Machine Learning* e Inteligência Artificial.

Neste capítulo serão apresentadas algumas das estratégias para a criação de um sistema conversacional. As estratégias apresentadas são utilizadas para a extração de significado de palavras ou de frases completas de forma a conseguir dar ao utilizador a resposta mais acertada. Começando por explicar neste capítulo as plataformas mais utilizadas para a criação destes sistemas, estas são normalmente fáceis de utilizar e de rápida integração com outras plataformas externas como Facebook Messenger[1], Slack [2], Telegram [3], mas têm a desvantagem de não permitirem um controlo absoluto do sistema, o que torna difícil criar novas funcionalidades para além das oferecidas pela plataforma. Na segunda secção deste capítulo são apresentados os standards mais utilizados para a criação de um chatbot de raiz, nomeadamente para representar a sua base de conhecimento, que como foi dito anteriormente é um dos pontos chaves para a criação de sistemas deste género. Estes standards que se

[1] `https://www.messenger.com/` (Dezembro 2018)
[2] `https://slack.com/` (Dezembro 2018)
[3] `https://telegram.org/` (Dezembro 2018)

baseiam na manipulação de padrões, garantem ao programador um controlo maior sobre o funcionamento desejado para o chatbot.

3.1 Análise das plataformas *online*

Para facilitar a criação de chatbots, algumas empresas fornecem os seus serviços de Compreensão de Linguagem Natural embutidos em plataformas, sendo estas normalmente de fácil intuição. Alguns exemplos destas plataformas são o DialogFlow[4](Google), Wit.ai [5](Facebook), LUIS [6](Microsft), Watson[7](IBM) e Amazon Lex[8](Amazon), tendo os seus serviços disponíveis nos respetivos web sites e sendo todas elas baseadas em *Cloud Computing*. *Cloud Computing* é um modelo que fornece constante acesso via Internet a uma grande variedade de recursos (exemplos: servidores, armazenamento de dados, aplicações e serviços), que podem ser rapidamente acedidos pelos utilizadores (Mell e Grance, 2011).

Estas plataformas fornecem um conjunto de métodos pré-definidos e pré-instalados, dando ao programador um conjunto de ferramentas que o ajudam a desenvolver um sistema conversacional robusto.

Graças à *Cloud Computing* estas plataformas podem ser facilmente acedidas em qualquer lugar, conseguem lidar com grandes quantidades de dados, estão sempre com a versão mais atual e, acima de tudo, não dependem de nenhum requisito de hardware.

Nesta secção queremos analisar algumas das plataformas mais utilizadas baseadas em *cloud computing* que utilizam ferramentas de CLN. Todas elas suportam várias linguagens de programação e diferentes linguagens naturais, mas cada uma tem funcionalidades diferentes e formas distintas de fazer a correspondência entre o que o utilizador insere e a resposta a dar.

Devido à ampla utilização de alguns conceitos importantes por estas plataformas, estes serão explicados previamente, nomeadamente os conceitos de *intents* e *entities*.

Um *intent* representa um mapa entre o que o utilizador diz e a ação que o chatbot vai desempenhar, isto é, uma parte de uma conversa. Uma *entity* são mecanismos para extrair parâmetros de uma entrada do utilizador em linguagem natural. Qualquer valor que se deseje extrair de uma conversa terá uma *entity* correspondente.

De forma a entender melhor estes conceitos é possível analisar a seguinte frase de exemplo, que poderia ser inserida por um utilizador: "Qual o tempo em Coimbra?". Esta frase faz parte de um *intent* que tem como objetivo analisar perguntas relacionadas com a meteorologia e de seguida a palavra "Coimbra" poderá corresponder a uma *entity* ou variável que guarda o nome da cidade para que o programa

[4] `https://dialogflow.com/` (Dezembro 2018)
[5] `https://wit.ai/` (Dezembro 2018)
[6] `https://www.luis.ai/home` (Dezembro 2018)
[7] `https://www.ibm.com/watson/` (Dezembro 2018)
[8] `https://aws.amazon.com/pt/lex/` (Dezembro 2018)

entenda qual o local onde se deseja saber o tempo.

Todas as plataformas aqui descritas permitem a definição de novos *intents* e de novas *entities*, assim como utilizar as pré-definidas pela plataforma. Algumas plataformas têm um *intent* especial chamado de *"Default Fallback"*, que representa a resposta que será dada ao utilizador se o sistema não conseguir identificar o *intent*.

3.1.1 DialogFlow

DialogFlow é uma plataforma para desenvolver chatbots baseado em conversas em linguagem natural. A visão desta plataforma é de ter uma curva de aprendizagem muito baixa e mesmo assim permitir criar chatbots robustos. Anteriormente chamada de Api.ai, foi adquirida pela Google e ganhou o nome de DialogFlow.

Esta plataforma usa um conceito chave chamado *"Contexts"* para modelar o comportamento do chatbot. O conceito de *context* é usado para distinguir as várias entradas do utilizador, podendo assim chamar diferentes *intents* dependendo das pesquisas anteriores do utilizador (Dutta, 2017).

Para entender o comportamento do DialogFlow, quando o utilizador faz o seu pedido, por texto ou voz, é primeiro verificado se essa entrada faz correspondência com alguns dos *intents* previamente colocados na plataforma. De seguida, se foi encontrada alguma correspondência com algum *intent* será retornada a resposta ligada a esse. Caso não encontre nenhuma correspondência entre o que foi dito pelo utilizador e os *intents* na plataforma, o *intent Default Fallback* é ativado e pode retornar, por exemplo, o valor "Não foi encontrada nenhuma resposta" (Gregori, 2017).

Um dos pontos chave na utilização desta plataforma é a sua funcionalidade de "*Slot-Filling*", que permite indicar para um dado *intent* quais as variáveis que devem ser guardadas e quais as que ainda falta guardar, permitindo assim armazenar dados ao longo da conversa e mais tarde utilizá-los.

Utilizando como exemplo um web site que faz entregas de pizzas para entender o conceito de "*Slot-Filling*", ao longo da conversa com o utilizador, o sistema de conversação desse site pode ir fazendo perguntas até que este preencha as variáveis ou *slots* necessárias para prosseguir com a encomenda da pizza. Perguntas como "Quais os ingredientes?", "Qual o local da entrega?" ou "A que horas quer que seja feita a entrega?", podem estar diretamente ligadas a *intents* específicos, que irão guardar variáveis para prosseguir com a encomenda.

Esta plataforma fornece integração com outras plataformas externas como Slack[9], Google Assistant[10], Facebook Messenger[11], Twitter[12], Kik[13], Telegram[14] e outras mais (Gregori, 2017). Das plataformas que aqui serão apresentadas é a que tem mais

[9] `https://slack.com/` (Dezembro 2018)
[10] `https://assistant.google.com/` (Dezembro 2018)
[11] `https://www.messenger.com/` (Dezembro 2018)
[12] `https://twitter.com/` (Dezembro 2018)
[13] `https://www.kik.com/` (Dezembro 2018)
[14] `https://telegram.org/` (Dezembro 2018)

integração com plataformas externas.

Segundo Dutta (2017), nesta plataforma é possível criar um chatbot robusto capaz de responder às perguntas do utilizador alinhando os seus *intents* e *contexts* a uma grande base de dados.

3.1.2 Wit.ai

Wit.ai é uma plataforma gerida pela empresa Facebook que oferece serviços de CLN. É grátis e suporta 50 linguagens naturais, mas apenas três linguagens de programação, nomeadamente Python, Node.js e Ruby, para além destes três SDK fornecidos no site da plataforma é possível utilizar a sua HTTP API e criar um sistema em qualquer outra linguagem. Esta plataforma não tem presente o *intent "Default Fallback"*, mas este poderá ser desenvolvido pelos programadores.

A Wit.ai foca-se em extrair significado de uma frase. Segundo Canonico e Russis (2018), é mais como um analisador de CLN, por isso qualquer integração com outras plataformas de chatbot, web sites e outros mecanismos que precisam de manter o contexto devem ser criados pelos programadores.

Para além do modelo de CLN que suporta *intents* e extrai *entities*, esta plataforma incorpora o modelo de histórias (*stories*) (Gregori, 2017). O modelo de histórias representa assim o comportamento do chatbot. Cada história retrata um exemplo de uma conversa humano-humano.

Para entender o modelo de histórias inicialmente tem que se treinar o chatbot a utilizar estas histórias, que como foi dito são exemplos de possíveis conversas humano-humano. Após se introduzir as histórias no sistema, de cada vez que for introduzida uma entrada pelo utilizador, o sistema vai tentar fazer correspondência entre as conversas da história e a entrada do utilizador e caso consiga encontrar numa das conversas uma correspondência irá devolver a resposta dada nessa conversa. Ao longo da conversa é possível ir extraindo variáveis de forma a guardar dados das conversas, as chamadas entidades. A engenharia de PLN nesta plataforma é treinada com exemplos (Dutta, 2017).

3.1.3 LUIS

LUIS (*Language Understanding Intelligent Service*) é uma plataforma da Microsoft que faz parte do serviço Azure Cloud Services e por fazer parte deste serviço a sua utilização tem um custo associado. Suporta muitas linguagens de programação, mas os SDKs disponíveis são apenas em C#, Python, Node.js e Android. Suporta também 10 linguagens naturais (Gregori, 2017).

Segundo Canonico e Russis (2018), para desenvolver um sistema de conversação robusto nesta plataforma deve ser em primeiro escolhido um domínio específico em que o sistema vai atuar. Isto porque esta plataforma tem um melhor desempenho se o contexto das conversas for conhecido previamente.

Nesta plataforma após se escolher o domínio em que o chatbot irá atuar, existe a funcionalidade de criar uma listas de palavras com os seus sinónimos de forma a

ter um sistema capaz de responder de forma mais eficiente aos pedidos do utilizador.

Sendo esta uma plataforma desenvolvida pela Microsoft, é possível utilizar modelos da Cortana[15] e do Bing[16], e uma vez que estes são modelos de pesquisa é possível com eles retornando pesquisas da Internet no formato JSON (*JavaScript Object Notation*). A plataforma LUIS oferece um conjunto de REST APIs que podem ser utilizados para automatizar o processo de desenvolvimento de sistemas deste género (Canonico e Russis, 2018).

A plataforma LUIS oferece-nos a possibilidade de construir o nosso próprio modelo do chatbot. Os *intents* desta plataforma são obtidos através de exemplos de conversas entre humanos e é possível nesses *intents* criar etiquetas específicas para conseguir extrair dados, as chamadas *entities* (Dutta, 2017). Por exemplo, na frase, "Marque um vôo para Roma", uma possível *entity* a ser extraída pelo sistema seria a cidade de destino, criando assim uma etiqueta na palavra Roma.

3.1.4 Watson Conversation

A plataforma Watson Conversation foi desenvolvida pela IBM e faz parte do seu serviço na *cloud*. Por essa razão, a sua utilização tem um custo associado. Suporta muitas linguagens de programação e 12 linguagens naturais. Entende *intents* e interpreta *entities*. Segundo Canonico e Russis (2018), esta plataforma tenta conseguir o que é pedido pelo utilizador tendo acesso a uma grande base de dados da IBM.

A plataforma divide o pedido do utilizador assim como as potenciais respostas no seu corpus (base de dados), que funciona como uma base de conhecimento do sistema, e examina o conteúdo de várias maneiras, até conseguir encontrar as respostas que lhe parecem mais acertadas. De seguida, analisa todos os resultados e concede a cada uma das respostas um grau de confiança, ajudando assim a escolher qual a resposta ou comportamento a dar ao chatbot.

3.1.5 Amazon Lex

Amazon Lex foi desenvolvida pela Amazon e faz parte dos serviços AWS (Amazon Web Services), por isso existem custos pela sua utilização. Apesar de suportar muitas linguagens de programação, apenas suporta uma linguagem natural, sendo ela o Inglês.

Uma vez que apenas pode ser acedida pelos serviços da Amazon, esta plataforma partilha o mesmo nível de conhecimento da Amazon Alexa que é o chatbot oficial da Amazon. Esta plataforma utiliza CLN para entender os *intents* dos textos introduzidos pelo utilizador, ajudando assim os programadores a conseguir uma sistema muito parecido com uma conversa entre humanos (Canonico e Russis, 2018).

[15] `https://www.microsoft.com/en-us/cortana` (Dezembro 2018)
[16] `https://www.bing.com/` (Dezembro 2018)

Segundo Gregori (2017), a Amazon Lex reduz o esforço para desenvolver um sistema conversacional capaz de ser embebido em plataformas externas, como o Facebook Messenger [17], Slack [18]ou Twilio[19], sendo fornecido pela plataforma um manual para a integração com estas plataformas externas, é possível interagir com esta plataforma utilizando texto ou voz. Suporta também SDKs para Java, JavaScript, Python, CLI, .Net, Ruby on Rails, PHP, GO e C++.

3.1.6 Comparação das Plataformas

De forma a resumir a análise feita e facilitar a sua comparação, esta secção coloca lado-a-lado as várias plataformas, nomeadamente na tabela 3.1. Cada plataforma é descrita através dos seguintes campos:

- Número de linguagens naturais que cada plataforma suporta (coluna LNs);
- Se incorpora ou não a língua portuguesa (coluna PT): importante de destacar uma vez que o trabalho pretendido será realizado nesta linguagem;
- Número de SDK's e em que linguagens de programação são suportadas (coluna LPs);
- Número de *entities* pré-construídas pela plataforma (coluna Entities), mostrando assim os dados que são possíveis de extrair sem ser necessário criar de raiz;
- Número de *intents* pré-construídos (coluna Intents), mostrando assim os domínios em que o sistema está mais à vontade. Para todas estas plataformas é possível criar novos *intents*, criando assim novos domínios onde o sistema consegue trabalhar;
- Se incorpora o *intent* de "Fall Back" (coluna FB): importante uma vez que este é um *intent* que será muito utilizado quando são feitas perguntas fora do domínio utilizado para criar o sistema;
- Se tem capacidade para ser integrado de forma rápida com plataformas externas (coluna Integr), tais como Facebook Messenger, Slack, etc.;
- Preço da sua utilização (coluna Preço): normalmente as plataformas deste tipo oferecem os seus serviços de forma gratuita e a partir de uma certa quantidade de utilizações é necessário pagar um certo preço.

Após a análise das plataformas mais utilizadas para a criação de sistemas conversacionais do mercado, é possível verificar que todas elas à exceção do Amazon Lex, tentam abranger com o uso das suas ferramentas de CLN um grande número de linguagens naturais. Deste modo, podem ser criados sistema de conversação capazes de abranger um grande número de utilizadores. Quanto à língua portuguesa, o serviço LUIS e a plataforma Watson Conversation não integram a sua variante europeia, no entanto os dois integram o português do Brasil.

Outra análise que se consegue fazer ao observar a tabela 3.1 é que todas estas plataformas tentam facilitar o trabalho ao utilizador na criação dos seus sistemas, incluindo nos seus serviços um conjunto de *intents* e *entities* geradas automaticamente

[17] `https://www.messenger.com/` (Dezembro 2018)
[18] `https://slack.com/` (Dezembro 2018)
[19] `https://www.twilio.com/` (Dezembro 2018)

	LNs	PT	LPs	Entities	Intents	FB	Integr	Preço
DialogFlow	15	Sim	11	60, desde localidades a cores	34, desde conversas curtas a dialogos	Sim	Sim	Grátis (Standard Edition)
Wit.ai	50	Sim	3: Node.js, Python, Ruby	22, desde localidades a emails	0	Não	Não	Grátis, Contrato para utilizações prolongadas
LUIS	10	Não	4: Python, Node.js, C, Android	13, desde números a geografia	20, desde agendamento de voos a meteorologia	Sim	Não	Grátis até 10 mil utilizações mês
Watson Conversation	12	Não	6	7, desde tempo até pessoas	0	Sim	Não	Grátis até 10 mil utilizações/mês
Amazon Lex	1	Não	9	5, desde Emails a números de telefone	16, desde tempo a pedidos de ajuda	Sim	Sim	Gratis até dez mil utilizações por texto e cinco mil por voz

TABELA 3.1: Tabela de comparação de plataformas

que facilitam a construção de sistemas em certos domínios à exceção das plataformas Wit.ai e Watson Conversation que não fornecem no seu sistema um conjunto de *intents* pré-definidos. Todas elas podem ser inicialmente utilizadas gratuitamente, sendo cobrada uma uma taxa à *posteriori* que é necessário pagar à medida que o número de utilizações vai crescendo. A plataforma DialogFlow é a única que permanece gratuita.

3.2 Standards de arquitetura

Um dos pontos importantes na criação de um sistema de conversação é a sua base de conhecimento e a forma como ela é mapeada. Este mapeamento deve conseguir permitir pesquisas rápidas de forma a devolver a resposta correta ao utilizador no mínimo tempo possível.

Tendo em conta estes objetivos, foram desenvolvidos vários standards para a criação e mapeamento das bases de conhecimento, de forma a conseguir com a utilização de padrões específicos de cada um destes standards, ser capaz de criar modelos que interpretam o que é pedido pelo utilizador e devolvem a resposta mais acertada.

Com estes padrões é possível criar modelos que conseguem interpretar a pergunta do utilizador, mesmo que a mesma seja feita de formas diferentes tendo estes modelos apenas que representar as palavras chave de cada pedido do utilizador.

Alguns dos standards mais utilizados para a criação destas bases de conhecimento são o AIML e o Chatscript. Nesta secção, serão analisados estes dois standards utilizados para a construção de uma base de conhecimento de um sistema conversacional e por fim será apresentada uma tabela comparativa entre os dois standards.

3.2.1 AIML

AIML (*Artificial Intelligence Markup Language*) é para muitos o ponto de partida para a criação de um chatbot. Foi criada em 1995 por Dr. Richard Wallace e serve como base para muitos chatbots, tais como o A.L.I.C.E e o ELIZA (Krantz e Lindblom, 2017; Giakoumakou, 2018). Segundo o autor (Giakoumakou, 2018), a grande virtude do AIML é a sua simplicidade e facilidade de aprender e implementar.

Este standard é próprio para a criação de uma base de conhecimento para um sistema conversacional. Para se entender o que é uma base de conhecimento, este é o local onde todos os dados se encontram e será nestes dados que o chatbot irá procurar a resposta a dar ao utilizador. Esta base de conhecimento poderá seguir as normas do AIML, normas estas que serão explicadas mais à frente.

O standard AIML é baseado em XML, não sendo necessário nenhum IDE específico pois é possível escrever os ficheiros AIML num simples editor de texto.

O conceito por detrás deste standard é a correspondência por padrões. Essencialmente, esta correspondência pode ter a forma de uma árvore de decisão. Inicialmente, é dado um input pelo utilizador e depois é feita uma pesquisa por entre os nós da árvore até encontrar uma correspondência. De seguida a ação encontrada é executada. Essa ação pode ser, por exemplo, uma resposta textual ou uma página web, dependendo das tags atribuídas ao nó encontrado.

O conceito de correspondência por padrões pode tornar-se muito pouco genérico se a base de conhecimento não for relativamente grande. Nesse caso, poderá haver muitas correspondências iguais para perguntas diferentes, o que pode resultar num output com uma resposta errada, devendo assim para o uso deste standard ser necessário um grande número de pares pergunta-resposta. Para entender a importância de uma grande base de conhecimento, analisemos o seguinte exemplo: existem muitas maneiras de perguntar "A que horas abre a loja?", como por exemplo "Quando abre a loja?", "Está aberta a loja hoje?" ou "A loja abre hoje?". Para se criar um chatbot robusto utilizando este standard é necessário ter uma grande variação para cada par pergunta-resposta.

Devido a ser necessário um grande número de pares para se conseguir formar o output com a melhor resposta, cria-se um novo problema, o da complexa manutenção do ficheiro AIML. Isto porque pode ser difícil para uma pessoa que não esteja familiarizada com a estrutura do ficheiro entendê-lo. Para atenuar este problema deve-se no início do projeto definir regras para a criação do ficheiro AIML, estabelecendo normas para a criação de novos pares perguntas-respostas e regras de formatação do ficheiro, facilitando no futuro a rápida interpretação do ficheiro.

Uma vez que os ficheiros AIML são baseados na linguagem XML, eles incluem tags ou etiquetas. É por isso necessário entender alguns conceitos por detrás de cada uma delas. Existem três tags chave para a criação de uma base de conhecimento em AIML: *category*, *pattern* e *template*. A tag *category* é a base para cada *intent*. É onde se encontram as perguntas que se tentará corresponder com o que o utilizador escrever e onde estarão também guardadas as respostas a essa pergunta. Cada pergunta será incluída numa tag *pattern* e a resposta a essa pergunta será incluída numa tag

template.

Como se pode ver na figura 3.1, é apresentado um exemplo simples de um ficheiro AIML, onde se consegue entender melhor os conceitos *category*, *pattern* e *template*.

```
<aiml>
    <category>
        <pattern>A QUE HORAS ABRE A LOJA</pattern>
        <template>A loja abre às 9 horas.</template>
    </category>
    <category>
        <pattern>QUANDO ABRE A LOJA</pattern>
        <template>A loja abre às 9 horas.</template>
    </category>
    <category>
        <pattern>HOJE ABRE A LOJA</pattern>
        <template>A loja abre às 9 horas.</template>
    </category>
    <category>
        <pattern>A LOJA ABRE HOJE</pattern>
        <template>A loja abre às 9 horas.</template>
    </category>

</aiml>
```

FIGURA 3.1: Demonstração dos conceitos em AIML

Existem muitos outros conceitos que se podem utilizar para criar um chatbot mais robusto, capaz de guardar variáveis, as chamadas *entities*. Ao se analisar a figura 3.2, existem dois novos conceitos apresentados, o conceito de *star* e *srai*. A tag *star* tem duas funcionalidades, sendo a primeira guardar valores e poder utilizá-los em qualquer altura da conversa. Por exemplo, assim é possível guardar o nome do utilizador ou outros dados importantes. A segunda funcionalidade é a permitir criar um chatbot que consiga responder a um maior número de perguntas. Por exemplo, se tivermos um *intent* com uma pergunta "OLÁ *", este *intent* fará correspondência com perguntas "Olá tudo bem?" ou "Olá como estás?", não sendo assim necessário escrever dois *intents*, um para cada pergunta, agilizando a criação da base de conhecimento.

O segundo conceito é a tag *srai*, utilizada para direcionar para outro *intent*. Desta forma é possível ter mais do que um *intent* com a mesma resposta, sendo apenas necessário um deles ter a resposta e os outros apenas a referência para essa resposta.

No exemplo dado na figura 3.2, verifica-se a utilização destas duas tags de forma a criar um chatbot capaz de responder a uma maior variação de perguntas. Seguindo o exemplo da figura, se o utilizador escrever "Sabes a capital de Portugal?", o *intent* correspondente será "SABES A CAPITAL *", que direciona para o *intent* "QUAL A CAPITAL DE PORTUGAL", retornando a resposta "Lisboa". Este *intent* foi escolhido pois a palavra Portugal foi guardada na tag *star* e utilizada para encontrar o

intent para que foi direcionado.

Existem outras tags para a criação de um ficheiro AIML, como: *random*, que escolhe um *intent* aleatoriamente; *that*, que escolhe o *intent* baseado no contexto atual; ou *think*, que é utilizada para guardar várias variáveis ou alterar o contexto da conversa; entre muitas outras.

```
<aiml>
    <category>
        <pattern>QUAL A CAPITAL DE PORTUGAL</pattern>
        <template>Lisboa.</template>
    </category>
    <category>
        <pattern>QUAL A CAPITAL DE ESPANHA</pattern>
        <template>Madrid.</template>
    </category>
    <category>
        <pattern>SABES A CAPITAL DE *</pattern>
        <template>
            <srai>QUAL A CAPITAL DE <star/></srai>
        </template>
    </category>

</aiml>
```

FIGURA 3.2: Exemplo AIML

Por fim, para se conseguir utilizar este standard é necessário a utilização de um interpretador específico, ou seja, um programa que consiga ler o input do utilizador e que tenha a capacidade para executar todas as regras do AIML. Existem muitos, em várias linguagens como Java, Ruby, Phyton. C++. C#, Pascal, entre outras (Geethanjali e Mary J, 2017).

3.2.2 Chatscript

ChatScript é uma linguagem de script desenvolvida por Bruce Wilcox e lançada em 2010. Apesar de ser mais recente que o AIML, tem tido muito sucesso. Ganhou em 2012 um concurso de Chatbots para "Best new Bot" e no mesmo ano um chatbot que utilizou este standard ganhou o Loebner Prize, é possivel ver os vencedores deste prémio assim como as tecnologias utilizadas por estes no Anexo B.

ChatScript é como no AIML organizado por padrões, que estão dentro de tópicos. Ao contrário do AIML que procura apenas pela melhor correspondência, o ChatScript começa por procurar pela melhor correspondência nos tópicos e depois executa as regras nesse tópico. Segundo o autor Giakoumakou (2018), o Chatscript proporciona uma manutenção mais facilitada nos ficheiros que representam a base de conhecimento que o AIML. No entanto, apesar do Chatscript fornecer as mesmas funcionalidades que o AIML, tem uma maior curva de aprendizagem.

Para ilustrar esta comparação entre o AIML e o Chatscript, o autor dá como exemplo um ficheiro AIML com 70 categorias, que resulta em 600 linhas de código mas, quando traduzido para ChatScript, necessita de apenas 70 linhas. Isto torna o

ChatScript muito mais fácil de entender por pessoas que não criaram o código, facilitando assim também a sua manutenção.

Apesar de se conseguir construir ficheiros em Chatscript com apenas algumas linhas de código, este tem a desvantagem de utilizar novas terminologias que programadores comuns não estão familiarizados, ao contrário do standard AIML que deriva do XML, tendo assim uma curva de aprendizagem maior.

Para entender melhor os conceitos para a criação de um ficheiro ChatScript são apresentados dois exemplos. Na figura 3.3, é apresentado na primeira linha o conceito de *topic* este chamado de introdução como se pode ver na linha 1 da figura.

```
1   topic: ~introdução repeat[olá hi]
2
3   t: [Olá][Hi], [fala comigo][diz alguma coisa].
4
5   u: (O que és tu) Eu sou uma máquina.
6
7   u: (* está tudo bem) Sim e contigo?
8
9   u: ([Quem Como]) Boa pergunta.
10
11  u: (O meu nome é _*) O teu nome é '_0 ?
12      a: (~sim) $nome = '_0
13      a: (~não) Está bem, então qual é o teu nome?
14
15  u: (Qual o meu nome) O teu nome é $nome .
```

FIGURA 3.3: Exemplo ChatScript 1

Qualquer ficheiro Chatscript precisa de um tópico para se conseguir escolher o contexto da conversa, sendo possível existir vários tópicos no mesmo ficheiro, estes serão selecionados consoante as correspondências feitas entre a entrada do utilizador e as palavras chave dentro dos parênteses retos de cada tópico, como é possível verificar na figura 3.3.

Na mesma figura na linha 1 é apresentado um método do Chatscript, o método "repeat". Este método, diz respeito à possibilidade do chatbot se conseguir repetir, podendo assim voltar a utilizar a mesma regra mesmo que esta tenha sido utilizada recentemente. Normalmente, com este standard, se existir a possibilidade de escolher entre várias perguntas, o chatbot irá escolher sempre um par novo, nunca repetindo o que já foi dito. Com o uso deste método, poderá voltar a escolher uma resposta já dada.

Seguindo um exemplo para explicar o uso do método "repeat", se o sistema puder escolher entre dizer "Tudo bem", "Como estás?" ou "Como estás hoje?", ele vai selecionar uma das possibilidades aleatoriamente e, na interação seguinte, poderá voltará a escolher a mesma. O uso deste método é visível na linha 3 da figura 3.3, onde é apresentado um novo conceito "t:". Este é o ponto de partida para o chatbot, sendo a frase que inicialmente será dita quando o sistema é iniciado e, como se consegue ver na figura, haverá várias possibilidades de escolha, sendo elas: "Olá, diz alguma coisa", "Hi, fala comigo", "Olá, fala comigo" ou "Hi, diz alguma coisa".

Na linha 5 da figura 3.3 é apresentado um novo conceito “u:”, que diz respeito ao que se espera que seja perguntado e ao que será respondido pelo sistema. O que está entre parênteses refere-se ao que poderá ser dito pelo utilizador, e o que está fora, refere ao que será respondido pelo sistema. Assim como no AIML, é possível agilizar o processo da criação dos *intents* utilizando o carácter *. Na linha 7 é possível ver que para qualquer pergunta do utilizador que inclua “está tudo bem”, irá fazer correspondência com esta regra apresentando a resposta ao utilizador "Sim e contigo?".

Outro conceito apresentado na figura 3.3 é o “a:”, onde é possível criar uma árvore de decisão consoante o que foi dito anteriormente. Como se vê na linha 11 da figura, após a pergunta do sistema, espera-se uma resposta de sim ou não e, consoante a resposta, o comportamento do chatbot será diferente.

Por último, nesta figura é mostrado que também é possível guardar variáveis ou *entities* em Chatscript, utilizando o conjunto de caracteres “_*” e, de seguida, guardando-o numa variável criada pelo programador, como se pode ver na linha 12, onde o nome do utilizador é guardado na variável “nome”.

Um exemplo de uma conversa baseada no Chatscript da figura 3.3 seria:

Chatbot

Chatbot: Olá, fala comigo
Utilizador: O que és tu?
Chatbot: Eu sou uma máquina.
Utilizador: Como ficou o jogo ontem?
Chatbot: Boa pergunta.
Utilizador: O meu nome é Ricardo
Chatbot: O teu nome é Ricardo?
Utilizador: Sim
Utilizador: Qual o meu nome?
Chatbot: O teu nome é Ricardo.

Outro exemplo para entender melhor as possibilidades de utilizar este standard é dado na figura 3.4, onde é demonstrado que é possível fazer perguntas ao utilizador consoante as variáveis guardadas. Nesta figura é apresentado um novo tópico chamado viagem e, consoante o que o utilizador pergunta, vão sendo guardadas variáveis, alterando assim o comportamento do chatbot.

```
1  topic: ~viagem repeat[]
2
3  t:[Olá][Hi], vou-te ajudar a escolher a viagem.
4
5  u:(* estou em _*) $partida = '_0
6
7  u:(gostava de ir para _*) $chegada = '_0
8
9  u:(!$partida) Onde estás?
10
11 u:(!$chegada) Onde gostavas de ir?
12
13 u:($partida $chegada) Tu queres ir desde $partida até $chegada .
```

FIGURA 3.4: Exemplo ChatScript 2

Um exemplo de uma conversa com este Chatscript seria:

Chatbot

Chatbot: Olá, vou-te ajudar a escolher a viagem.
Utilizador: Gostava de ir para Lisboa.
Chatbot: Onde estás?
Utilizador: Não sei.
Chatbot: Onde estás?
Utilizador: Eu estou em Coimbra.
Chatbot: Tu queres ir desde Coimbra até Lisboa.

3.2.3 Comparação dos standards de arquiteturas

De forma a facilitar a comparação entre os dois standards foi criada a tabela 3.2 que mostra, para cada um, um conjunto de itens, nomeadamente: a sua complexidade na manutenção; a sua curva de aprendizagem; a sua base de conceito; a variedade de interpretadores disponíveis; e o impacto de cada um representando a sua presença nos artigos mais recentes.

	AIML	**Chatscript**
Manutenção	difícil	fácil
Curva de aprendizagem	fácil	difícil
Base de conceito	baseado em padrões	baseado em padrões
Interpretadores	grande variedade	pequena variedade
Prova Social	grande	pequena

TABELA 3.2: Tabela AIML vs. Chatscript

Como se observa na tabela, o AIML tem uma complexidade maior que o Chatscript ao nível da manutenção. Isto deve-se porque o Chatscript possibilita que na mesma linha de código se inclua várias possibilidades de respostas a dar ao utilizador utilizando a tag 'a', ao contrário do standard AIML, que precisa de um novo conjunto de tags para cada possibilidade de respostas criando assim novas linhas de código, o que resulta num ficheiro muito mais extenso.

Uma vez que o AIML deriva do XML, uma linguagem normalmente conhecida pelos programadores, a sua curva de aprendizagem é muito menor em comparação com o standard Chatscript que utilizada novas terminologias que o programador comum não está habituado, sendo por isso mais difícil de aprender.

Outros aspetos importantes apresentados na tabela são a quantidades de interpretadores criados em outras línguas de programação e a sua prova social. Uma vez que o AIML é o standard que aparece em um maior número de artigos relacionados com a criação de sistemas de conversação dá a este standard uma prova social maior. E uma vez que tem uma prova social maior, muitos interpretadores foram até à altura criados, por exemplo em linguagens de programação como Python, Ruby, Javascript e uma grande variedade em Java.

3.3 Ferramenta de Engenharia de Pesquisa

Como foi referido no capítulo anterior, uma simples abordagem à criação de um sistema conversacional pode basear-se num sistema de Recuperação de Informação , sendo este denominado por como um sistema baseado em RI (Quarteroni e Manandhar, 2009). Estes sistemas têm como objetivo o de encontrar automaticamente a informação que o sistema necessita para responder ao utilizador na pergunta feita por este, procurando por exemplo por palavras-chaves e de seguida com a informação extraída conseguir dar uma resposta o mais acertada possível ao utilizador, pesquisando na sua base de conhecimento ou fontes externas, como por exemplo a Internet.

Estes sistemas podem ser construídos com base em ferramentas de engenharia de pesquisa. Neste subcapítulo iremos apresentar a ferramenta de pesquisa chamada Apache Lucene que é uma biblioteca Java de código aberto para pesquisa de textos.

Segundo McCandless, Hatcher e Gospodnetic (2010), Lucene é uma biblioteca de Recuperação de Informação (RI) de alta performance. Com ela é possível adicionar capacidades de indexação e de pesquisas em textos, retornando ao utilizador uma rápida resposta. É um projeto de código aberto implementado na linguagem de programação Java, sendo membro do Apache Jakarta e licenciado pela *Apache Software*. Segundo os mesmos autores é a biblioteca em Java de RI mais popular.

Lucene pode indexar e pesquisar qualquer que sejam os dados em formato de texto, esta biblioteca não se restringe à fonte dos dados, o seu formato ou a linguagem natural em que são representados desde que seja possível converter para o formato de texto. Isto significa que é possível indexar dados de web sites, documentos Word, ficheiros HTML, ficheiros em PDF ou outros formatos que seja possíveis de converter antecipadamente para texto.

Para entender o poder de pesquisa do Lucene, iremos explicar alguns dos métodos para criar pesquisas nesta ferramenta. Inicialmente é necessário criar campos de pesquisa, estes campos são importantes e essenciais nesta ferramenta uma vez que serão os campos que o programador irá escolher para fazer as pesquisas consoante o que for pedido pelo utilizador. Por exemplo, se quisermos que o nosso sistema pesquise numa grande base de dados de livros, alguns exemplos de campos que iremos ter serão os campos: "autor", "titulo" e "texto". Todos estes campos devem ser preenchidos com os devidos dados.

Após definir os campos de pesquisa é possível começar a fazer pedidos de pesquisa nesta ferramenta. Serão, de seguida, apresentados alguns exemplos de possíveis pesquisas:

- **titulo:"Lucene in Action"**: nesta pesquisa iremos procurar pela frase "Lucene in Action" no campo "titulo" e apenas os dados que satisfazerem essa pesquisa são apresentados;
- **titulo:"Lucene in Action" AND autor: "Michael McCandless"**: nesta pesquisa iremos pesquisar em dois campos, apresentando os dados que satisfazerem unicamente as duas pesquisas, isto é a sua interseção. É possível também utilizar o operador **"OR"**, onde são apresentados os dados que satisfaçam apenas uma das pesquisas.

- **titulo:"Lucene in Action" - titulo:"Second Edition"**: com esta pesquisa é mostrado como é possível excluir palavras ou frases que não queremos que nos sejam apresentadas nos resultados, neste caso queremos todas os documentos encontrados que no título tenham a frase "Lucene in Action"mas que não tenham a frase "Second Edition".

- **texto:"lucene*"**: com esta pesquisa é apresentado o uso de *wildcards* que permitem fazer pesquisas não-exatas, neste exemplo serão apresentados ao utilizador todos os dados que tenham no campo "texto" a palavra que comece por "lucene".

- **titulo:"Lucne $\sim$ 2"**: esta pesquisa mostra outras das funcionalidades do Lucene, a funcionalidade de pesquisar por aproximações, nesta pesquisa está a ser feito um pedido de pesquisa que procure pela palavra "Lucne"com no máximo duas alterações, ou seja, é possível criar pesquisas com redundância para erros ortográficos.

Após se analisar todas estas possibilidades de pesquisas é importante referenciar alguns aspetos importantes em relação às últimas duas pesquisas. No uso dos *wildcards*, não é possível utilizar estes no início da pesquisa sendo por isso impossível criar pesquisas como por exemplo **texto:"*lucene"**. Por último é importante também dizer que o uso das pesquisas por aproximações pode ser até no máximo de 2 alterações, contando a alteração de letras maiúsculas por minúsculas como uma alteração. Segundo a própria documentação do Lucene, o uso de mais que duas correções não compensa o nível necessário de processamento com a eficácia de encontrar a informação que o utilizador necessita.

Uma vez que o Lucene é uma ferramenta de pesquisa muito utilizada, vários analisadores de textos foram criados para varias línguas naturais, como é o caso do língua portuguesa onde foi criado o "PortugueseAnalyser". Estes analisadores vêm com funcionalidades específicas de cada língua ajudando assim nas pesquisas. No caso deste analisador para a língua portuguesa, existe um método capaz de fazer pesquisas com textos com ou sem acentuação ortográfica. Isto facilita as pesquisas por parte dos utilizadores, já que se podem esquecer de colocar um acento ortográfico e mesmo assim é possível obter o resultado desejado. Outro recurso importante que estes analisadores oferecem é uma lista pré-construída de *stopwords*, específica para cada língua natural, criando assim um filtro de palavras que não serão consideradas importantes para fazer a pesquisa, exemplos de algumas dessas palavras são: "a", "o","e","é","do","da", entre muitas outras para a língua portuguesa.

Outro aspeto muito importante nesta ferramenta é a capacidade de atribuir um *score* a cada pesquisa, retornando para o utilizador uma lista dos dados encontrados que fizeram correspondência à pesquisa feita pelo utilizador ordenada pelo seu *score*. Este é determinado pela correspondência, quanto maior for a correspondência entre o que foi pedido pelo utilizador e o que foi encontrado nos seus campos de pesquisa maior será o *score*, sendo por norma a correspondência com maior *score* a que será apresentada ao utilizador.

Capítulo 4

Estudo do sistema

Neste capítulo será apresentado o estudo do sistema proposto, começando por descrever os requisitos funcionais e não funcionais, a ferramenta escolhida para o desenvolver, a arquitetura que o sistema deverá seguir, a preparação do ambiente e quais foram os recursos externos utilizados.

4.1 Análise de requisitos

Tendo em consideração os objetivos e funcionalidades apresentados nos capítulos anteriores, procedeu-se à identificação dos requisitos através de uma descrição de alto nível do sistema de respostas automáticas a perguntas frequentes a desenvolver.

Serão descritos em primeiro lugar os requisitos funcionais do sistema e por último os requisitos não funcionais importantes para atingir os objetivos do sistema.

4.1.1 Requisitos Funcionais

O sistema a desenvolver tem os seguintes requisitos funcionais:

1. Obter os dados de uma FAQ. Este requisito consiste em processar todos os dados necessários para a construção da base de conhecimento a partir de um ficheiro com todas as FAQs disponíveis de uma entidade. Contempla os seguintes parâmetros:
    - É fornecida a lista de FAQs que será utilizada para a criação da base de conhecimento do sistema.
    - Normalização do ficheiro recebido. Por exemplo: Perguntas começam com "P:" e respostas começam com "R:".
    - Utilização do ficheiro normalizado com as FAQs.
2. Capacidade de apresentar o par pergunta-resposta mais provável. Este requisito consiste em procurar na base de conhecimento o melhor par a apresentar ao utilizador, consoante a sua pergunta inicial. Contempla os seguintes parâmetros:
    - Pergunta inserida pelo utilizador em formato de texto.
    - Pesquisa na base de conhecimento do sistema pela pergunta semanticamente mais parecida.
    - Apresentação da resposta à pergunta selecionada.

3. Apresentar outros pares pergunta-resposta alternativos. Este requisito consiste na criação de uma lista de perguntas candidatas a apresentar ao utilizador. Caso o utilizador não esteja satisfeito com a resposta dada, ele deverá conseguir ver a resposta para outras perguntas semanticamente mais parecida consideradas próximas pelo sistema. Contempla os seguintes parâmetros:

    - Pergunta inserida pelo utilizador em formato de texto.
    - Apresentação da resposta para a pergunta mais próxima e o número de respostas alternativas encontradas.
    - Caso o utilizador não esteja satisfeito com a resposta apresentada, é-lhe dada a opção de ver os pares alternativos.

Os diagramas de atividades de todos estes requisitos funcionais, assim como os mockups que representam a forma de como estes devem atuar, podem ser observados no Anexo C

4.1.2 Requisitos Não Funcionais

Para além do comportamento específico é necessário determinar de que forma o sistema deve operar. Isto é especialmente importante no que toca à tomada de decisões relativamente à arquitetura a seguir. Assim, existem 4 elementos importantes a ter em consideração:

1. Reusabilidade. Uma vez que o sistema tem como objetivo ser utilizado em vários domínios, sendo cada domínio uma determinada área suportada por um ficheiro ou mais de FAQs que contém os pares perguntas-respostas, o sistema deve ser capaz de alterar o seu domínio consoante o(s) ficheiro(s) FAQ(s) que lhe é(são) atribuído.

2. Desempenho. O sistema deverá ser capaz de obter uma resposta para apresentar ao utilizador o mais rapidamente possível, uma vez que este sistema irá ser utilizado por humanos para os conseguir esclarecer nas suas dúvidas. Uma resposta rápida é crucial neste tipo de sistemas.

3. Tolerância à variação linguística. O sistema deve capaz de tolerar as seguintes especificações:

    (a) O sistema deve ser capaz de entender a mesma pergunta feita de formas diferentes usando diferentes palavras (uso de sinónimos).

    (b) O sistema deve ser capaz de fazer as pesquisas independentemente do género, número ou conjugação das palavras (uso de lemas).

    (c) O sistema deve ser capaz de pesquisar independentemente da palavra estar em letras maiúsculas ou minúsculas (insensível ao caso).

    (d) O sistema deve ser capaz de fazer a pesquisa pela palavra correta, independentemente dos erros ortográficos.

    (e) O sistema deve ser capaz de fazer as pesquisas independentemente da acentuação gráfica.

4. Confiabilidade. Uma vez que o objetivo final deste sistema é o de apresentar uma resposta ao utilizador, é importante que tenha uma taxa de acerto elevada, de forma a conseguir encontrar o maior número de vezes a resposta que ele procura.

4.2 Ferramentas Escolhidas

Após a exploração de várias ferramentas, como será demonstrado no capítulo 5, a ferramenta escolhida para a criação do sistema de conversação foi o Lucene, disponível como uma biblioteca em Java de código aberto, com classes que podem ser estendidas.

Esta escolha deveu-se ao facto de esta ferramenta ser uma biblioteca em Java, com uma grande capacidade de indexação de conteúdos textuais, e por permitir um maior controlo sobre o sistema, quando comparado com as plataformas online para o desenvolvimento de chatbots. O Lucene permite alterar a forma como é feita a escolha da resposta a apresentar, e assim personalizar o sistema de forma a obter os resultados mais adequados para o objetivo pretendido.

O Lucene oferece ainda um conjunto de possibilidades, tais como a utilização de uma lista de palavras a ignorar nas pesquisas (*stopwords*) ou um mapa de sinónimos, a considerar quando é feita a pesquisa. Estas funcionalidades permitem, por um lado, ignorar palavras menos relevantes para o significado das frases e, por outro, aumentar a abrangência das pesquisas.

O Lucene permite definir vários campos de pesquisa associados a um documento, e definir quais devem ser indexados, em princípio, e sobre os quais serão feitas as pesquisas. Por exemplo, no nosso caso é possível definir um campo "Pergunta", com o texto da pergunta, e um campo "Resposta" com o texto da resposta associada.

Por último, é importante reforçar que o Lucene se baseia numa indexação dos campos textuais que permite uma pesquisa de alto desempenho. Isto responde a um dos requisitos não funcionais descrito anteriormente.

4.3 Arquitetura Proposta

Nesta secção é apresentada a arquitetura que o sistema deverá seguir. Como representado na figura 4.1, o utilizador começa por fazer uma pergunta ao sistema, que a irá processar com a ajuda de um analisador e dos vários recursos utilizados. Inclui-se aqui uma base de sinónimos, uma lista de *stopwords* e ferramentas como o *Tokenizer*, *POS Tagger* e *Lemattizer*, estes três últimos são utilizados no processo de lematização. Todos estes recursos e ferramentas serão explicados no subcapítulo 4.5.

Após o pré-processamento, é feita uma pesquisa nos dados da nossa base de conhecimento, previamente indexados, que deverá chegar à pergunta semanticamente mais próxima do texto inserido. A resposta a essa pergunta será finalmente apresentada ao utilizador.

Este módulo de pesquisa, assim como a criação da base de conhecimento indexada, serão explicados em detalhe no capítulo 5.

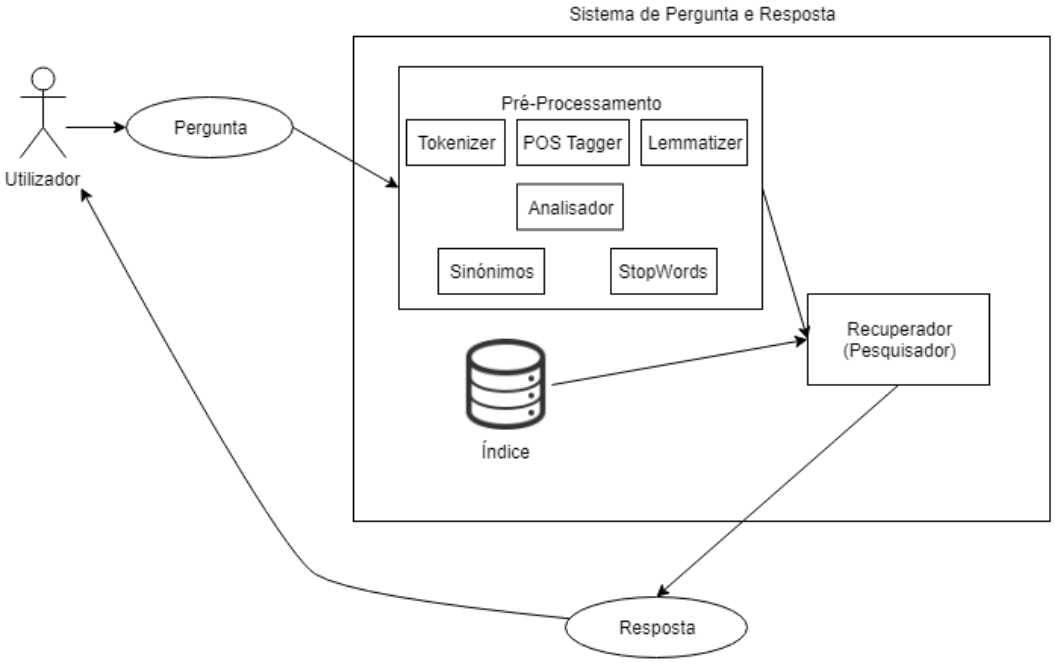


FIGURA 4.1: Visão geral da arquitetura do sistema

4.4 Preparação do Ambiente

Uma vez que a biblioteca utilizada será o Lucene, na linguagem de programação Java, foi necessário escolher um IDE que suportasse essa linguagem de programação para a construção do sistema de conversação.

O IDE escolhido para a criação do sistema foi o Eclipse, onde foi criado um projeto Maven com as dependências presentes na figura 4.2.

Nesta figura é possível ver três dependências:

- a primeira chamada de `lucene-core` necessária para utilizar a biblioteca Lucene;
- a segunda dependência chamada de `lucene-queryparser` utilizada para transformar as pesquisas do utilizador num formato que o Lucene consiga entender e prosseguir assim com a sua pesquisa;
- e por último a dependência `lucene-analyzers-common`, chamada para poder utilizar analisadores de texto pré-construídos pelo Lucene e assim conseguir normalizar pesquisas em linguagens naturais específicas, como é o caso do analisador chamado de "PortugueseAnalyzer" que tem incorporado funções específicas para a língua portuguesa.

Após se ter preparado este ambiente, com todas as dependências necessárias para a utilização do Lucene, é possível começar a criar o nosso sistema de conversação.

```
<dependency>
    <groupId>org.apache.lucene</groupId>
    <artifactId>lucene-core</artifactId>
    <version>7.4.0</version>
</dependency>

<dependency>
    <groupId>org.apache.lucene</groupId>
    <artifactId>lucene-queryparser</artifactId>
    <version>7.1.0</version>
</dependency>


<dependency>
    <groupId>org.apache.lucene</groupId>
    <artifactId>lucene-analyzers-common</artifactId>
    <version>7.4.0</version>
</dependency>
```

FIGURA 4.2: Dependências do projeto Lucene

4.5 Recursos Utilizados

Neste subcapítulo serão apresentados todos os recursos externos utilizados para a criação do sistema. Destes destacam-se quatro:

- A base de conhecimento, representada por um ficheiro que contém todas as perguntas a que o sistema deve conseguir responder e respetivas respostas;
- Uma lista de palavras e respetivos sinónimos, isto é, palavras que podem ser utilizadas com o mesmo sentido;
- A lista de *stopwords* a considerar, isto é, palavras que terão menos relevância na pesquisa;
- O lematizador, um programa que tem como objetivo obter o lema (forma não flexionada) das palavras.

Todos estes recursos serão descritos com maior detalhe nas secções seguintes.

4.5.1 Base de conhecimento

Representa todo o conhecimento sobre o qual o sistema irá pesquisar encontra-se armazenado na forma de perguntas, às quais deverá conseguir responder, e respetivas respostas, sempre em linguagem natural.

Esta base de conhecimento deverá ser criada a partir de um ficheiro ou mais com as FAQs, que contenham as perguntas mais frequentes e suas respetivas respostas (e.g., acerca de uma instituição alvo). Todas estas FAQs recolhidas, por exemplo a partir dos próprios websites da instituição alvo serão normalizados e será criado um único ficheiro que será utilizado pelo sistema.

4.5.2 Sinónimos

De forma a ter em conta a variedade linguística foi integrada no sistema informação acerca de palavras portuguesas que podem ser usadas com o mesmo significado.

Ao se considerar mais variações (sinónimos), o sistema poderá conseguir associar o texto inserido a perguntas que podem ter o mesmo significado, ainda que não usem as mesmas palavras. Isto contribuirá para um aumento da abrangência do sistema.

Para este fim, foi utilizado um recurso de pares de palavras relacionadas disponíveis para download[1], construído por Gonçalo Oliveira (2018), este utilizou dez recursos léxico-semânticos para o português, incluindo dicionários, tesauros e bases de conhecimento.

De todas as relações disponíveis no ficheiro, foram apenas utilizadas as relações de sinonímia que apareceram no mínimo em 6 recursos. A figura 4.3 mostra a estrutura do ficheiro com as relações a usar. Cada linha do ficheiro tem uma palavra; seguida do nome de uma relação, neste caso de sinonímia; seguida de outra palavra, sinónima da primeira; e, por fim, o número relativo ao número de recursos em que a relação foi encontrada. Quanto maior este número, mais comum será a relação, para além de que pode ser usado como um indicador de confiança, uma vez que a maioria dos dez recursos explorados na obtenção destas relações foram criados manualmente.

```
finalizar SINONIMO_V_DE rematar 5
consolação SINONIMO_N_DE consolo 6
quantidade SINONIMO_N_DE quantia 6
investigar SINONIMO_V_DE indagar 6
sovar SINONIMO_V_DE bater 6
estupidez SINONIMO_N_DE burrice 6
indeterminado SINONIMO_ADJ_DE indefinido 6
benefício SINONIMO_N_DE lucro 6
```

FIGURA 4.3: Exemplo do recurso utilizado para a extração de sinónimos (Gonçalo Oliveira, 2018)

4.5.3 Stopwords

Nas tarefas de classificação de texto ou Recuperação de Informação (RI) é comum não considerar todas as palavras da mesma forma, porque nem todas terão a mesma relevância para a interpretação do texto. Uma abordagem comum a esta situação passa pela remoção das chamadas *stopwords*, palavras que, na língua ou domínio em que se está a trabalhar, são muito frequentes e contribuem pouco para o significado dos documentos. Exemplos destas palavras, em português, incluem artigos, como "a" e "uns", preposições, como "de" e "em", entre outras, como "que". Por não terem grande contributo para o significado das pesquisas é comum, simplesmente, não as considerar.

Existem duas abordagens para escolher que palavras considerar como *stopwords*, nomeadamente:

- A criação de uma lista de *stopwords* específica para a língua e domínio do problema, baseada nos documentos sobre os quais as pesquisas incidirão. Este processo passa pela utilização de um método estatístico chamado de IDF (*Inverse Document Frequency*), que conta o número de documentos em que uma

[1] `http://ontopt.dei.uc.pt/index.php?sec=download_outros`

certa palavra apareceu. Tal como em (Robertson, 2004), considera-se cada frase encontrada na base de conhecimento como um documento. Assim, para uma dada palavra, quanto mais alto o seu IDF, mais específica ela é, e por isso maior é a sua importância na classificação dos documentos em que ocorre. Ou seja, o valor IDF está linearmente ligado com a relevância das palavras.

A fórmula para calcular o IDF é: $idf(t_i) = \log(\frac{N}{n_i})$, em que t_i representa o termo a ser pesquisado, N o número de documentos indexados e n_i o número de documentos nos quais esse termo foi encontrado (Robertson, 2004).

Esta abordagem obriga ao cálculo dos valores de IDF para cada palavra e tem ainda a desvantagem de ser pouco representativa caso a base de conhecimento contenha poucos documentos de pesquisa.

- A segunda abordagem é a utilização de uma lista de *stopwords* previamente extraída por outras entidades, específica para a língua, mas o mais abrangente possível, isto é, não focada num domínio concreto.

  Estas palavras poderão depois ser, simplesmente, ignoradas, quer nos documentos indexados, quer nas pesquisas. Para este trabalho, tendo em conta que lidará com a língua portuguesa, uma lista deste género é disponibilizada no website do Snowball [2], um processador de texto gratuito e amplamente utilizado para criar algoritmos de RI.

4.5.4 Lematizador

Outro dos recursos externos utilizados foi o lematizador, este tem como tarefa identificar a forma base de uma palavra, isto é, a forma em que ela aparece num dicionário, também denominada de lema (Ingason et al., 2008). Este processo consiste em transformar substantivos na sua forma masculina singular, adjetivos na sua forma masculina e singular e verbos no infinitivo. Para entender de uma forma mais clara este conceito, exemplos de palavras como "gato", "gata", "gatos" e "gatas", têm como lema a palavra "gato", ou palavras como "sou", "era" e "seria", têm como lema a palavra "ser".

Ao utilizar um lematizador num sistema de perguntas e respostas, estamos mais uma vez a garantir alguma tolerância ao nível da variação linguística e a aumentar a abrangência do sistema. Neste caso, não será preciso que o utilizador acerte no género, número ou tempo verbal em que as perguntas se encontram na base de conhecimento.

Por exemplo, se a base de conhecimento incluir a pergunta "Como posso alugar um imóvel?", o utilizador não precisa de fazer a pesquisa de forma exatamente igual e pode, por exemplo, pesquisar por "Como alugo um imóvel?", traduzida pelo lematizador para "Como alugar um imóvel?" que terá uma representação semântica mais próxima ao texto na base de conhecimento e aumentará a probabilidade de dar a resposta pretendida.

Para esta finalidade, foi utilizado o LemPort, um lematizador disponível para a língua portuguesa[3] e desenvolvido em Java, o que facilitou a sua integração.

[2] `http://snowball.tartarus.org/algorithms/portuguese/stop.txt` (Dezembro 2018)

[3] `https://github.com/rikarudo/LemPORT`

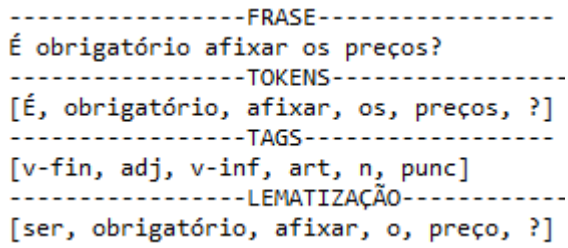


FIGURA 4.4: Exemplo do Lematizador

O módulo LemPort utilizado faz parte da biblioteca NLPPort[4], esta biblioteca fornece um conjunto de ferramentas de PLN para a linguagem portuguesa. Foi criada analisando bibliotecas de PLN como o OpenNLP e analisadores de dependências como o MaltParser, acrescentando ainda novas funcionalidades e módulos para a língua portuguesa.

A utilização do LemPort implica um pré-processamento através de outros dois módulos, o TokPort e o TagPort, todos integrados na ferramenta NLPPort (Rodrigues, Oliveira e Gomes, 2018), cuja função se explica de seguida:

- TokPort: responsável pela divisão de uma frase em unidades básicas (palavras e pontuação), normalmente chamadas de *tokens*.
- TagPort: responsável pela atribuição de uma das seguintes classes gramaticais a cada *token*: "adjetivo", "advérbio", "artigo", "nome", "numeral", "nome próprio", "preposição", "verbo" e se necessário "pontuação".
- LemPort: identificação do lema de cada *token*, considerando também a sua classe gramatical.

Como a classe gramatical depende da ordem das palavras na frase, as análises anteriores são realizadas para uma frase de cada vez.

Na figura 4.4, é possível ver um exemplo do funcionamento destas ferramentas. É inicialmente apresentada a frase "É obrigatório afixar os preços?". De seguida são criados os *tokens*, a partir da separação de cada palavra na frase. Se seguida são apresentadas as *tags*, que representa a classe gramatical de cada *token* e por fim é apresentada a frase lematizada: "Ser obrigatório afixar o preço?".

[4] `https://github.com/rikarudo/` (Dezembro 2018)

Capítulo 5

Trabalho Experimental

Neste capítulo é apresentado o sistema de respostas automáticas a perguntas frequentes desenvolvido, começando por explicar como foi obtida e criada a sua base de conhecimento, os projetos desenvolvidos até à escolha da ferramenta a utilizar, assim como o projeto realizado. Por fim são apresentados os testes feitos ao sistema.

5.1 Base de Conhecimento

Um dos recursos mais importantes para um sistema de respostas automática a perguntas frequentes é a sua base de conhecimento, é nesta onde se encontram todos os dados relativos às perguntas e respostas que serão analisadas na pesquisa de forma a conseguir ajudar o utilizador na sua tarefa.

Nesta secção será descrita a entidade escolhida que fornece o documento de FAQs para o desenvolvimento deste sistema, assim como explicada a extração dos pares pergunta-resposta.

5.1.1 Entidade Escolhida

A entidade que serviu de base a este trabalho foi o Balcão do Empreendedor (BDE)[1]. O BDE constitui um ponto único de acesso aos serviços digitais relacionados com o exercício de atividade económica em Portugal. Dirige-se aos empresários que desejem realizar serviços e obter informações inerentes às atividades económicas que praticam.

Sendo esta uma ponte de acesso a informação, existirá um grande número de dados a retirar do seu portal na Web, entre os quais listas de perguntas e respostas já estruturadas (FAQs), também disponibilizadas aos utilizadores.

Após uma análise ao portal do BDE foi possível encontrar dois ficheiros em PDF com todas as FAQs disponíveis. O primeiro tem 124 e o segundo 54 pares pergunta-resposta, o que perfaz um total de 178 pares para a construção da base de conhecimento.

5.1.2 Análise dos Dados

Após identificados os documentos com as FAQs, foi necessário analisá-los para entender a sua estrutura e quais seriam os pares pergunta-resposta candidatos para

[1] `https://bde.portaldocidadao.pt/evo/balcaodoempreendedor.aspx` (Dezembro 2018)

utilização no nosso sistema.

Apesar dos documentos serem diferentes na forma como apresentam os pares, a estrutura dos dois documentos segue um padrão onde existem vários grupos de perguntas, separados por secções. Cada secção é apresentada por um título seguido do grupo de pares pergunta-resposta correspondentes a esse título.

Uma vez que o objetivo deste trabalho não será o de alterar as perguntas encontradas nos documentos FAQs, após se analisar os documentos observou-se que existem algumas perguntas que só por si não permitem identificar o seu contexto, que é o caso das perguntas unicamente com o texto "Definição", em que a resposta depende da secção em que esta se encontra. Por esta razão, as nove perguntas deste tipo não foram incluídas na base de conhecimento do sistema de conversação, que ficou com um total de 169 pares pergunta-resposta.

5.1.3 Extração dos Dados

A última fase foi a extração e criação do documento a utilizar na construção da base de conhecimento. Aqui foram detetados dois problemas, sendo o primeiro problema o formato em que os ficheiros se encontravam (PDF) sendo necessário alterá-lo para um formato que fosse manipulável, e o segundo problema encontrado está relacionado com as diferentes estruturas dos dois documentos, a resolução destes dois problemas são detalhados nos parágrafos seguintes.

Para resolver o primeiro problema foi necessário procurar uma biblioteca que pudesse receber como entrada um ficheiro PDF e retornasse o seu texto num formato manipulável. Para isso foi utilizada a biblioteca PDFBOX[2], gratuita, desenvolvida pela Apache em Java, e capaz de extrair conteúdos de ficheiros no formato PDF.

O segundo problema foi resolvido com a criação de dois procedimentos diferentes, um para cada documento uma vez que os dois têm estruturas diferentes na apresentação das perguntas. Por exemplo, um dos ficheiros apresenta as perguntas numeradas e entre secções, o segundo ficheiro simplesmente vai apresentando as perguntas e respostas com links entre os pares.

Cada procedimento criado percorre linha-a-linha o ficheiro de texto criado com a biblioteca anteriormente referida, tendo que conseguir identificar que linhas correspondem a perguntas e quais correspondem a respostas e, por fim, guardá-las num documento com um formato manipulável.

Uma vez que nem todas as linhas correspondem a perguntas ou a respostas, como é o caso dos títulos de cada secção, número de páginas ou notas de rodapé, os algoritmos deverão ser capazes de descartar todas estas linhas, uma vez que estas não devem fazer parte da base de conhecimento. O resultado deve ser um documento textual, com todos os pares de perguntas e respostas, a usar como fonte para a base de conhecimento do sistema de conversação.

5.1.4 Criação da base de conhecimento

A figura 5.1 apresenta um diagrama que ilustra, de uma forma geral, a extração dos dados assim como a criação da base de conhecimento. O ponto de partida para a

[2] `https://pdfbox.apache.org/` (Dezembro 2018)

sua criação será um documento com todos os pares pergunta-resposta apresentado no subcapítulo anterior, que deverá ser fornecido pela organização onde o sistema vai atuar ou extraído do seu website na secção das perguntas frequentes. Após se ter o documento com os pares é necessário executar um método que consiga extrair apenas os pares desse documento.

Após a extração do documento (i.e., obtenção do texto em formato manipulável), é necessário normalizá-lo, passando este a ter uma estrutura que o sistema de perguntas e respostas automáticas consiga entender. No nosso caso, optamos por cada pergunta começar por "P:" e cada resposta começar por "R:". De seguida, o sistema irá indexar o conteúdo desse documento, com recurso a uma base de sinónimos e ainda dos módulos TokPort, TagPort, e LemPort, apresentados no subcapítulo 4.5.4, criando por fim a nossa base de conhecimento completamente indexada.

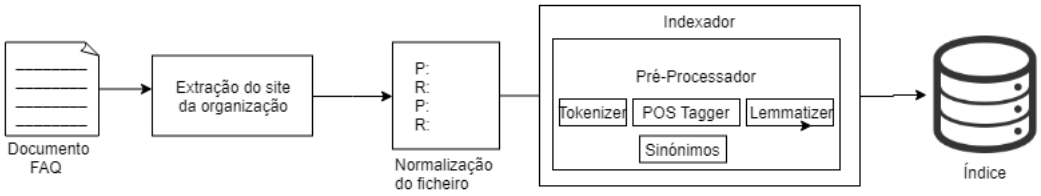


FIGURA 5.1: Visão geral da criação da base de conhecimento

5.2 Projetos Desenvolvidos

Nesta secção serão apresentados todos os projetos desenvolvidos para a criação do sistema de respostas automáticas a perguntas frequentes proposto.

Tendo em consideração a análise de todas as plataformas e ferramentas feita no capítulo 3, foi necessário entender como estas se comportavam na criação de um sistema deste género e assim chegar a uma justificação mais fundamentada da ferramenta escolhida.

Na primeira subsecção é apresentada uma primeira versão do projeto, esta baseada numa plataforma para a criação de sistemas conversacionais, onde se referem todas as vantagens e desvantagens para a sua criação. A subsecção seguinte apresenta um projeto utilizando um dos *standards* analisados e como este se comportou na criação do sistema. Por último é apresentada uma última alternativa ao desenvolvimento do sistema, que deu origem à sua terceira e última versão, baseada numa ferramenta de Recuperação de Informação. Esta última versão é apresentada em maior detalhe na secção seguinte deste capítulo.

5.2.1 Projeto DialogFlow

Tendo em conta os objetivos iniciais deste trabalho e as ferramentas identificadas na análise realizada antes do desenvolvimento propriamente dito, faria sentido basear o sistema numa plataforma que tivesse já o intuito de nos ajudar a criar um sistema

conversacional.

Como apresentado no capítulo 3, existem várias plataformas disponíveis desse tipo, onde foi analisado o seu custo, flexibilidade, linguagem, suporte da plataforma, inclusão de ferramentas de processamento de linguagem em português e ligação a plataformas externas.

A escolha recaiu sobre o DialogFlow uma plataforma desenvolvida pela Google, dando assim alguma confiança na sua utilização, à partida, devido às seguintes razões: inclui ferramentas de CLN para a língua portuguesa; ser uma plataforma que disponibiliza um SDK em Java; e permitir a sua utilização de forma gratuita.

Apesar de uma das grandes funcionalidades que deu popularidade ao DialogFlow ser a funcionalidade de "*Slot Filling*" explicada na secção 3.1.1, esta característica não traz grande utilidade ao modelo e chatbot que se pretende construir. Isto porque se pretende criar um chatbot adaptável a diferentes domínios, onde a criação de variáveis para um domínio específico poderiam deixar de fazer sentido no caso do domínio do chatbot se alterar. Por isso a utilização desta funcionalidade não foi considerada para a construção deste sistema.

Na altura em que o projeto DialogFlow se iniciou, existiam duas versões disponíveis para utilização, a versão 1 e a versão 2. A versão 2 em comparação com a versão 1, é a versão mais recente da plataforma, segundo o site oficial do Dialogflow. A alteração mais importante feita na plataforma é a de incluir novos agentes de gestão dos pedidos da API facilitando o uso aos programadores via SDK.

Apesar da segunda versão ser a que teria maiores vantagens, esta até à altura encontrava-se em fase beta, por isso a sua utilização poderia não ser o mais estável possível. Desta forma, foi decidido utilizar a versão 1 para a criação do nosso sistema.

Para se criar o projeto no DialogFlow e o poder aceder via SDK é necessário ir ao site, criar um projeto e adicionar os *Intents*, que representam a base de conhecimento do sistema conversacional. Neste caso, os nossos *intents* serão as perguntas e respostas extraídas das FAQs. Após se criar todos os *intents*, foi pedida a chave de acesso via servidor a esses *intents* e criou-se o primeiro programa em Java com o SDK fornecido pelo DialogFlow.

Após algumas utilizações com o programa criado, verificaram-se alguns problemas, sendo o principal a impossibilidade de fazer a gestão dos *Intents* via SDK. Este apenas permitia acesso à funcionalidade de *chat* onde todas as pesquisas do utilizador eram feitas. Qualquer outra funcionalidade, como a criação ou alteração de novos *Intents* tinha de ser feita manualmente no site da plataforma. Esta falta de flexibilidade foi razão suficiente para não avançar mais no primeiro projeto baseado no DialogFlow.

Nesta altura verificou-se ainda que na versão 2 do DialogFlow, ainda em fase beta, existiria a funcionalidade de gestão via SDK. Assim, criou-se o segundo projeto DialogFlow em Java utilizando o SDK da versão 2.

Após se ter o projeto a funcionar e de ter todos os *Intents* criados, sendo agora possível fazer a gestão dos mesmos via SDK devido às novas funcionalidades da

versão 2, foi altura de entender como o DialogFlow fazia a correspondência entre o que era pedido e o que era respondido, de forma a conseguir ter um maior controlo sobre o que seria apresentado ao utilizador. Por falta de documentação da plataforma não se conseguiu entender como se conseguia aceder a essa funcionalidade.

Enviou-se um email ao suporte do DialogFlow a perguntar como se poderia ter acesso à funcionalidade de correspondência e como se poderia alterar caso necessário, mas a resposta foi que essa funcionalidade não estava disponível e que não nos poderia ser dada nenhuma informação acerca de como essa correspondência é feita, nem sobre as técnicas de Compreensão da Linguagem Natural (CLN) usadas. Os emails podem ser visualizados no Anexo D.

O desenvolvimento deste projeto e da investigação associada permitiu perceber que todas as plataformas disponíveis para a criação deste tipo de sistema têm uma curva de aprendizagem pequena e possibilitam realmente a criação de um chatbot funcional em pouco tempo. Por outro lado, verificou-se que permitem apenas uma gestão de alto nível, que limita o controlo de componentes mais específicas para o processamento da língua, o que também impede a experimentação de diferentes técnicas para CLN. Ou seja, ao optar por estas plataformas, ficaríamos dependentes das ferramentas sobre as quais elas funcionam de raiz.

5.2.2 Projeto ProgramAB

Após se analisar os standards para a criação de um sistema de conversação, chegou-se à conclusão que poderia haver vantagens em adotar o standard AIML. Este standard foi usado nos mais recentes vencedores do Premio Loebner como se pode ser no Anexo B, o que lhe dá uma prova social do seu funcionamento.

Após a escolha do standard a utilizar foi necessário encontrar uma biblioteca em Java que incorporasse um interpretador para o AIML. Foi assim identificada a biblioteca ProgramAB, que implementa o standard do AIML.

O primeiro passo foi a construção do ficheiro AIML que servirá como base de conhecimento do nosso sistema. Para a sua construção foi utilizado o documento previamente extraído das FAQs e, aplicando as normas do standard AIML, foi assim criado o primeiro ficheiro AIML.

Este primeiro ficheiro foi criado de uma forma simples: com todos os pares pergunta-resposta com a mesma estrutura utilizando as etiquetas “Category”, que representam onde os pares se vão encontrar; etiquetas “Pattern”, que representam as perguntas e o local onde será feita a correspondência com o que o utilizador escrever na pesquisa. Segundo as normas do AIML, o texto da etiqueta “Pattern” têm que ter todas as letras em maiúsculas e não ter o ponto de interrogação final. Foi ainda utilizada a etiqueta “Template”, que representa as respostas, como se pode ver na figura 5.2.

O segundo passo na criação deste sistema, passou por procurar formas de aumentar a tolerância nas perguntas feitas pelo utilizador. Esta tolerância permitirá ao utilizador não ficar dependente de ter que pesquisar pela forma exata em que as perguntas se apresentam na base de conhecimento. Para conseguir criar esta redundância no sistema foi utilizado uma das funcionalidades de *wildcards* do standard

```
<category>
    <pattern>
    O REGISTO DOS ESTABELECIMENTOS DE ALOJAMENTO LOCAL ESTÁ SUJEITO A TAXAS
    </pattern>
    <template>
    Não, a mera comunicação prévia está isenta de taxas.
    </template>
</category>
<category>
    <pattern>
     SENDO O TITULAR DA EXPLORAÇÃO, TENHO DE SER EU A EFETUAR A MERA COMUNICAÇÃO
     PRÉVIA NO BALCÃO ÚNICO ELETRÓNICO OU POSSO PEDIR A UM TERCEIRO QUE A FAÇA
    </pattern>
    <template>
    A submissão da mera declaração prévia no balcão único eletrónico pode ser
    efetuada por qualquer pessoa, desde que mandatada pelo titular da exploração
    para o efeito, nomeadamente, com poderes para assinar as declarações prestadas
    e o termo de responsabilidade em representação do titular da exploração.
    </template>
</category>
```

FIGURA 5.2: Exemplo da estrutura do primeiro ficheiro AIML

AIML, descrita na secção 3.2.1.

Esta funcionalidade do standard permite ao utilizador escrever a mesma pergunta de formas diferentes e mesmo assim encontrar a resposta correta a dar. Para isso, pode usar-se no conteúdo da etiqueta "Pattern" o caracter * para representar quaisquer sequências de texto. Para entender esta funcionalidade, segue-se o seguinte exemplo; Caso o ficheiro AIML tenha na sua etiqueta "Pattern" o seguinte texto: " ONDE POSSO * CARTÃO DE CIDADÃO", o caracter * vai fazer correspondência com qualquer texto que seja escrito entre a sequência de palavras "ONDE POSSO" e "CARTÃO DE CIDADÃO", por isso frases como "Onde posso fazer o meu cartão de cidadão?" ou "Onde posso criar o meu cartão de cidadão?" irão fazer correspondência com o texto na etiqueta "Pattern" criando assim uma tolerância maior ao utilizador na forma como formula as suas frases.

No caso do nosso trabalho, este caracter foi usado para substituir palavras menos relevantes do documento, as chamadas de stopwords, apresentadas na secção 4.5.3.

Para a criação do documento AIML com as stopwords foi aplicado o método IDF, explicado também na secção 4.5.3. Este método consiste em verificar cada palavra na base de conhecimento e dar um peso consoante a sua frequência em todos os documentos. Quanto mais baixo for este peso, menor é relevância dessa palavra para encontrar a resposta a dar. Após se aplicar este método, foram identificadas 2315 palavras diferentes no documento com os seus IDF associados.

De seguida foi estabelecido um limite até onde se iria considerar que uma palavra era uma candidata a ser utilizada como stopwords. Para os primeiros testes, as palavras com um IDF menor que 1.7 seriam consideradas como stopwords. Isto corresponde a um total de 20 palavras, listadas na figura 5.3.

Tendo em consideração esta lista, foi criado o novo ficheiro AIML com a substituição dessas palavras pelo wildcard do AIML, como mostra a figura 5.4.

de	0.200340062296633
a	0.44183275227903923
o	0.6409614273893751
que	0.780158557549575
do	0.8754687373538999
ou	1.0897233412508638
é	1.1349799328389845
da	1.1823821717335685
em	1.252762968495368
não	1.252762968495368
para	1.2738163776932003
no	1.3512030413086205
alojamento	1.4226620052907655
um	1.4350845252893227
os	1.4863778196768733
com	1.526651718814813
local	1.5686159179138452
se	1.5976034547870976
à	1.6582280766035324
estabelecimentos	1.6582280766035324

FIGURA 5.3: Tabela IDF.

Nesta figura 5.4, podemos ver dois exemplos de perguntas que podem ter muitas variantes. Para entender melhor o uso dos wildcards no AIML, utilizaremos como exemplo uma possível pergunta que existe no ficheiro de FAQs: "O que é um conjunto comercial?", esta pergunta após se utilizar as regras do AIML, em que todas as perguntas devem estar em letra maiúscula e sem pontos de interrogação, foi ainda alterado para que todas as palavras que aparecem na tabela do IDF criada, apresentada na figura 5.3, fossem alteradas para o carácter "*", não podendo por regra existir dois destes caracteres seguidos, criando assim a nova pergunta "* CONJUNTO COMERCIAL", como é visível na figura 5.4.

Após se ter esta nova pergunta no formato AIML, utilizaremos duas possíveis perguntas feitas pelo utilizador como "O que é um conjunto comercial?" ou "Em que consiste um conjunto comercial?", estas frases permitem ser comparadas com a nova pergunta "* CONJUNTO COMERCIAL", pois, neste caso, o wildcard usado permite que sejam acrescentadas várias palavras no início da pesquisa, que terá, no entanto, de terminar com as palavras "conjunto comercial".

De seguida foi necessário perceber como é que biblioteca Program AB escolhia a melhor resposta e se seria possível alterar essa forma de escolha, de forma a aplicar diferentes métodos de similaridade semântica na escolha da melhor pergunta.

No entanto, devido à complexidade da biblioteca e à falta de documentação, não foi possível entender em que ponto era feita a correspondência das perguntas feitas pelo utilizador com os "Patterns" no ficheiro AIML. Isto significa que teríamos um

```
<category>
    <pattern>
    * CONSIDERADO * ÁREA * VENDA
    </pattern>
    <template>
    A área de venda é toda a área destinada a venda de produtos, onde os compradores
    tenham acesso aos produtos que se encontrem expostos ou onde estes são preparados
    para entrega imediata, nela se incluindo a zona ocupada pelas caixas de saída e
    as zonas de circulação dos consumidores internas ao estabelecimento, nomeadamente
    as escadas de ligação entre os vários pisos.
    </template>
</category>
<category>
    <pattern>
    * CONJUNTO COMERCIAL
    </pattern>
    <template>Um conjunto comercial é um empreendimento composto por um conjunto
    diversificado de estabelecimentos de comércio a retalho e/ou de prestação de
    serviços, sejam ou não propriedade ou explorados pela mesma entidade, que preencha
    cumulativamente os seguintes requisitos: a) Disponha de um conjunto de instalações
    e serviços concebidos para permitir a uma mesma clientela o acesso aos diversos
    estabelecimentos; b) Seja objeto de uma gestão comum, responsável, designadamente,
    pela disponibilização de serviços coletivos, pela instituição de práticas comuns e
    pela política de comunicação e animação do empreendimento.
    </template>
</category>
```

FIGURA 5.4: Exemplo do ficheiro AIML com wildcards

controlo limitado do sistema e que não seria possível experimentar diferentes métodos de mapeamento entre perguntas. Por esta razão, optamos por não continuar a desenvolver o sistema com esta solução baseada no standard AIML.

5.2.3 Projeto Lucene

Depois de identificar problemas nas duas soluções anteriores, decidiu-se explorar uma ferramenta de engenharia de pesquisa, capaz de alcançar os objetivos propostos.

Mais propriamente, foi identificada a biblioteca Lucene, escrita em Java, de código aberto, e muito utilizada para a indexação e pesquisa de documentos de texto. Devido ao seu poder de indexação, com o Lucene foi possível indexar os pares pergunta-resposta da base de conhecimento e conseguir uma pesquisa rápida no seu conteúdo.

Esta ferramenta serviu de base ao desenvolvimento da versão final do nosso sistema e é descrita em detalhe na secção seguinte.

5.3 Sistema Baseado em Engenharia de Pesquisa

Nesta secção será demonstrado como o sistema de perguntas e respostas proposto foi desenvolvido utilizando a ferramenta de recuperação de informação (RI) Lucene.

Para além de descrita a utilização do Lucene, será apresentado o analisador de texto escolhido, as métricas de similaridade utilizadas para escolher a melhor resposta a dar ao utilizador, os campos de pesquisa em que serão realizadas as correspondências e, por fim, demonstrado como o projeto apresenta a resposta.

5.3.1 Arquitetura do Sistema

Neste subcapítulo é apresentada uma visão geral da arquitetura do sistema de respostas automáticas a perguntas frequentes.

Como é possível observar na figura 5.5, o sistema recebe como entrada uma pergunta do utilizador. Esta pergunta passará por um analisador, capaz de interpretar o que o utilizador pediu, assim como utilizar uma base de sinónimos e uma base de stopwords. Este analisador é apresentado com maior detalhe no subcapítulo 5.3.2.

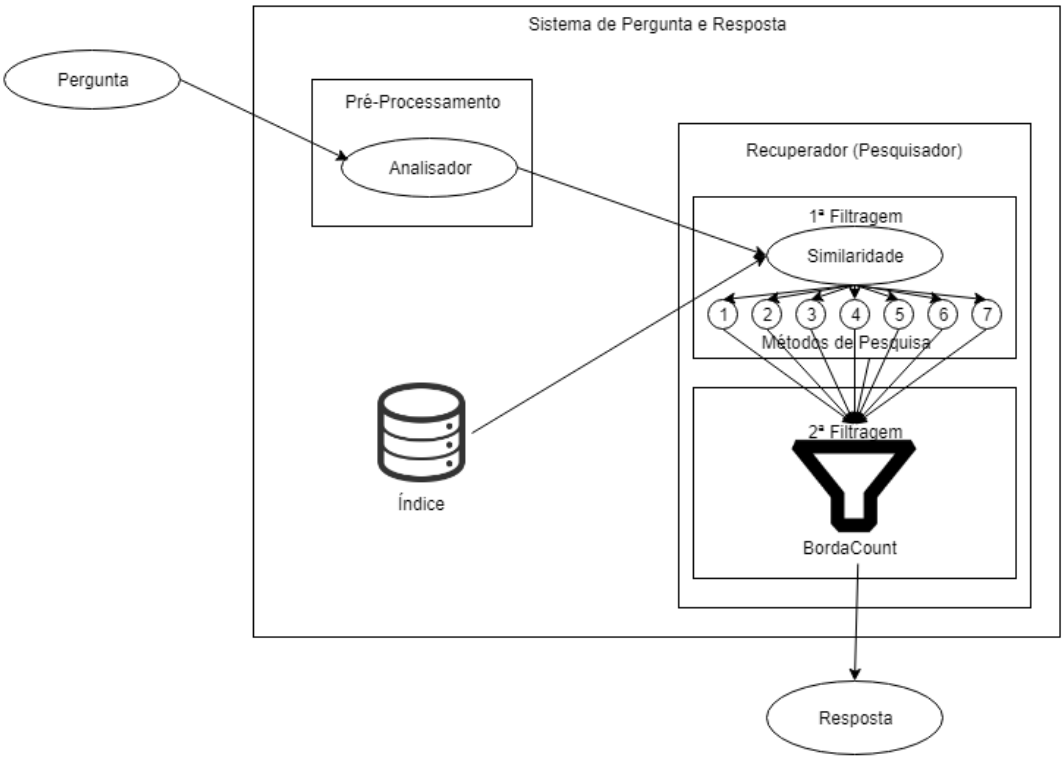


FIGURA 5.5: Arquitetura do Sistema de Respostas Automáticas a Perguntas Frequentes

Após a passagem pelo analisador e com a base de conhecimento já indexada, chega-se ao novo módulo de recuperação ou pesquisa do par a apresentar ao utilizador. Este módulo tem como principal funcionalidade receber a pergunta analisada no processo anterior e com ela pesquisar na base de conhecimento pela melhor resposta a dar ao utilizador.

Como se pode ver na figura, este módulo tem dois filtros. Neste contexto, estes representam o conjunto de pares que foram selecionados como possíveis candidatos

de todos os que formam a base de conhecimento, tendo como objetivo final a apresentação de apenas um par ao utilizador.

O primeiro filtro baseia-se no método de similaridade do Lucene descrito no subcapítulo 5.3.3 e tem como objetivo dar um valor de correspondência entre o que foi pedido pelo utilizador e o que é apresentado da base de conhecimento, resultando num maior valor caso exista uma grande similaridade entre o que foi pesquisado pelo utilizador e o que foi encontrado na base de conhecimento. Uma vez que existem várias formas de fazer essa correspondência no Lucene descritos no subcapítulo 5.3.3, foram criados sete métodos, como mostra também a figura.

Após a primeira filtragem, obtém-se sete listas de possíveis pares pergunta-resposta, respetivamente de acordo com os sete métodos utilizados. Estes métodos são descritos no subcapítulo 5.3.5. De seguida, é necessário escolher um único par para apresentar ao utilizador e, para isso, será aplicada uma segunda filtragem onde se utiliza o método BordaCount(Emerson, 2013), também apresentado no subcapítulo 5.3.5.

Após a passagem pela segunda filtragem, o sistema encontrará o par mais adequado à pergunta do utilizador, criando assim a resposta a apresentar.

5.3.2 Criação do analisador

No Lucene, um analisador pode ser empregue para processar o texto, analisando-o e indexando-o quando é feita uma pesquisa por parte do utilizador. Este analisador tem a principal função de separar o texto da pesquisa palavra por palavra, criando os chamados *tokens*, e pode ainda aplicar um conjunto de filtros, onde se pode incluir a remoção de stopwords, a consideração de sinónimos, entre outros, consoante o objetivo pretendido.

Para uma análise simples e abrangente, o Lucene inclui um analisador pré-construído, chamado de `StandardAnalyzer`. Ele faz apenas a separação das pesquisas palavra por palavra, mas o único filtro que inclui é o das stopwords, neste caso, para o inglês.

No entanto, uma vez que o Lucene é usado mundialmente e para diferentes línguas, muitos analisadores acabaram por ser construídos tendo em vista especificidades de diversas línguas. Por exemplo, há um analisador para o português, o `PortugueseAnalyser`, que inclui stopwords na língua portuguesa, e oferece a funcionalidade de entender as pesquisas mesmo que o utilizador se esqueça da acentuação gráfica nas palavras.

Apesar da existência do analisador para a língua portuguesa, decidimos que seria importante recorrer a algumas alterações com vista à sua melhoria. Foi então criado um analisador específico para este sistema, ainda que aproveitando algumas das funcionalidades do `PortugueseAnalyser`.

O novo analisador criado recebeu: um novo conjunto de stopwords e um conjunto de sinónimos, estes para que em cada pesquisa fosse possível considerar palavras diferentes, mas com o mesmo significado. Por outro lado, a separação do texto em tokens do `PortugueseAnalyser` manteve-se, bem como a capacidade de ignorar a acentuação gráfica nas palavras.

Um dos recursos utilizados foi um novo ficheiro de stopwords utilizado pelo processador de texto Snowball[3]. Este ficheiro foi usado em alternativa ao obtido anteriormente no projeto apresentado na secção 5.2.2 com recurso ao método estatístico IDF por duas razões: a primeira porque se considerou que a quantidade de textos utilizada para a criação desse ficheiro era muito pequena; e segundo, por os resultados obtidos serem específicos de um certo domínio, o que ia contra um dos objetivos deste projeto, o de criar um projeto capaz de funcionar independentemente do domínio escolhido. De referir que o processador de texto Snowball de onde retiramos o ficheiro das *stopwords* é amplamente utilizado.

Outro dos recursos foi o da base de sinónimos, apresentada no capítulo 4.5.2. Assim, com a utilização deste recurso, aumentamos a tolerância para pesquisas feitas com palavras diferentes mas com o mesmo significado, não sendo por isso necessário ao utilizador pesquisar pelas exatas palavras que estão na base de conhecimento.

Por fim, o último recurso utilizado na criação do nosso Analisador foi o filtro utilizado no `PortugueseAnalyser`, chamado de `PortugueseLightStemFilter`. Este é responsável por ignorar a acentuação gráfica nas palavras, sendo assim capaz de pesquisar pelas palavras corretas, mesmo que o utilizador se tenha esquecido da correta acentuação.

Todos estes recursos foram analisados e propostos indo de acordo com os requisitos não funcionais previstos, mais especificamente o da tolerância à variabilidade linguística, descrito na secção 4.1.2. O código fonte do analisador desenvolvido é apresentado no Anexo F.

5.3.3 Cálculo da Similaridade

Um dos componentes chaves do Lucene é a sua capacidade de pesquisar pelos documentos indexados e apresentar uma lista de candidatos à resposta a dar ao utilizador. Cada resposta candidata resulta da aplicação de um método em que atribui um resultado numérico a cada uma, sendo o par pergunta-resposta candidato que recebe o maior valor o principal a apresentar ao utilizador. Por outras palavras, este método calcula a similaridade entre a pesquisa e cada um dos documentos, atribuindo-lhes um valor.

De forma a entender como o Lucene escolhe a resposta a dar ao utilizador, foi necessário criar um método de similaridade capaz de dar uma pontuação a cada resposta candidata. Um dos métodos mais usados para este fim é o TF-IDF. O Lucene utiliza uma variante deste método chamado de *TFIDFSimilarity*.

Segundo o método *TFIDFSimilarity*, a atribuição de um valor de similaridade entre uma pesquisa (q) e um documento (pergunta) indexado (d) é feita de acordo com a seguinte equação:

$$sim(q,d) = coord(q,d).queryNorm(q).\sum_{t,q}(tf(t,d).idf(t)^2.t.getBoost().norm(t,d))$$

Explicando cada uma das partes da equação temos que:

[3] `http://snowball.tartarus.org/algorithms/portuguese/stop.txt` (Dezembro 2018)

- $coord(q,d)$ é um fator que atribui um valor consoante o número de termos encontrados na pesquisa (q) que se encontram num determinado documento (d). Assim os documentos que têm mais termos na pesquisa (q) têm maior similaridade do que os documentos que não têm nenhum termo na pesquisa (q).

- $queryNorm(q)$ é um fator de normalização de pesquisas (q), usado para permitir obter resultados de forma a se poder comparar diferentes pesquisas.

- $tf(t,d)$, indica a frequência do termo (t) em cada documento (d). Cada documento representa um par pergunta-resposta, sendo este valor maior caso o termo (t) pesquisado apareça em mais documentos (d), representado pela função:

$$TF = \frac{n^o \quad de \quad ocorrencias \quad do \quad termo \quad t \quad no \quad documento \quad d}{n^o \quad total \quad de \quad termos \quad no \quad documento \quad d}$$

- $idf(t)$, indica a relevância de um termo (t) num certo documento, quanto menor o seu idf menor será a sua relevância, representado pela função:

$$IDF = \log \frac{n^o \quad total \quad de \quad documentos}{n^o \quad de \quad documentos \quad onde \quad se \quad encontra \quad o \quad termo \quad t}$$

- $t.getBoost()$ representa um impulso que desejamos acrescentar à equação consoante o campo em que estamos a fazer a pesquisa. Estes campos onde serão feitas as pesquisas serão apresentados na subsecção seguinte. Este impulso pode ser um número entre 0 e 1. Nas experiências realizadas este parâmetro teve o valor 1 para qualquer campo de pesquisa.

- $norm(t,d)$ indica o número de termos que o documento d tem, dando um maior valor aos documentos com menos termos, mas que incluam o termo t em relação aos documentos que tem muitos termos e que incluam o termo t.

Observando esta função para o cálculo da similaridade, conclui-se que:

- Os documentos que contêm todos os termos da pesquisa terão maiores valores do que os que não têm ou têm menos termos.

- Frases onde se obtém correspondências com termos com uma frequência menor no documento terão valores mais altos, uma vez que, segundo o IDF, o valor atribuído a um termo é maior caso este ocorra menos vezes no documento. Desta forma pesquisas por palavras-chave terão maiores valores, do que outras pesquisas, uma vez que palavras-chaves tendem a ter menos frequência nos documentos.

- Documentos curtos são preferíveis a documentos longos, uma vez que em documentos curtos será mais fácil fazer a correspondência de todos os termos, aumentando o valor atribuído pela similaridade.

O código fonte da similaridade desenvolvido é apresentado no Anexo G.

5.3.4 Campos de Pesquisa

O Lucene representa os documentos através de um conjunto de campos que, para além do conteúdo do documento, podem ser usados para associar outras informações relacionadas. Dependendo da forma como se pretende fazer a pesquisa, é possível apenas considerar alguns campos.

No nosso caso foram definidos três campos de pesquisa: "Pergunta", que representa a pergunta do par pergunta-resposta; "Resposta", que representa a resposta à pergunta; e ainda "PerguntaLematizada", que representa a pergunta, mas desta vez na sua forma lematizada.

Apenas se considerou a lematização da pergunta, uma vez que a maior correspondência entre o que o utilizador irá pesquisar e a base de conhecimento estará na pergunta. Por isso foi necessário que esta estivesse na forma lematizada para se conseguir fazer correspondências nas pesquisas independentemente da forma verbal, género e número das palavras.

Após a criação dos campos é possível fazer quaisquer pesquisas sobre eles, consoante o objetivo do sistema. A tabela 5.1 mostra um exemplo de um documento que representa uma par pergunta-resposta com todos os campos preenchidos.

Documento	
Pergunta	`O que é e como aderir à chave móvel digital?`
Resposta	`A Chave Móvel Digital (CMD) é um serviço...`
PerguntaLematizada	`O que ser e como aderir a a chave móvel digital?`

TABELA 5.1: Exemplo dos campos de um documento no Lucene

5.3.5 Pesquisa

Esta secção descreve como o sistema de perguntas e respostas escolhe a resposta a dar ao utilizador, consoante a sua pesquisa.

Este processo passa por duas fases. A primeira fase passa por calcular a similaridade, como explicado em 5.3.3, e assim obter os documentos candidatos a responder ao utilizador. O número máximo de pares pergunta-resposta candidatas para cada pesquisa foi definido como cinco, e é apresentada uma lista de cinco candidatos, ordenados do mais para o menos relevante.

Contudo de forma a conseguir um maior número de pesquisas, e visto que há várias formas de fazer a pesquisa, foram criadas sete variantes para cada texto inserido pelo utilizador. Cada pesquisa retorna cinco candidatos, que podem ser comuns em pesquisas diferentes.

Estas sete pesquisas foram criadas de forma a conseguir utilizar os vários campos de pesquisa, de formas diferentes, criando assim vários tipos de pesquisa para obter a resposta mais adequada a dar ao utilizador. Estas pesquisas são:

- **PesquisaPergunta** representa os pares pergunta-resposta candidatos encontrados utilizando apenas o campo "Pergunta".

- **PesquisaPerguntaLematizada** representa os candidatos encontrados utilizando apenas o campo "PerguntaLematizada". Para esta pesquisa, o texto inserido pelo utilizador é também ele lematizado.
- **PesquisaPerguntaFuzzy** representa os candidatos encontrados utilizando o campo "Pergunta", mas tirando partido do método de pesquisa *fuzzy* do Lucene. Este método permite encontrar documentos com palavras que tenham até duas alterações quando comparadas com a pesquisa. Por exemplo, se a pesquisa tiver a palavra "emóbel", este método pode associá-la ao termo "imóvel", aumentando assim a tolerância a erros ou variações ortográficas.
- **PesquisaPerguntaLematizadaFuzzy** representa os candidatos encontrados utilizando o campo "PerguntaLematizada", mas tira partido do método de pesquisa *fuzzy*.
- **PesquisaPerguntaResposta** representa os candidatos encontrados utilizando não só o campo "Pergunta", mas também o conteúdo do campo "Resposta".
- **PesquisaPerguntaLematizadaResposta** representa os candidatos encontrados utilizando os campos "PerguntaLematizada" e "Resposta".
- **PesquisaPerguntaRespostaFuzzy**, representa os candidatos encontrados utilizando os campos "PerguntaLematizada" e "Resposta", tirando partido do método de pesquisa *fuzzy*.

Após se ter todos os candidatos de acordo com cada uma das sete pesquisas, é necessário selecionar qual dos documentos candidatos é o melhor para dar uma resposta ao utilizador. Para descobrir esse documento foi utilizado o método BordaCount.

O BordaCount é um método de votação que considera o lugar em que um candidato aparece em varias listas. Para isso, atribui um valor a cada lugar, sendo o valor do primeiro lugar na lista o candidato com o valor mais alto, ou seja, que teve maior correspondência e o último lugar o que terá o valor mais baixo, ou seja, teve menor correspondência.

Exemplificando, este valor atribuído pelo método BordaCount é o número de lugares que serão representados, no caso de haver uma lista com três candidatos ordenados por um determinado *score*, o que estiver em primeiro lugar receberá o valor de três, o segundo de dois e o último de um valor. No caso deste sistema, cada pesquisa retorna cinco candidatos, havendo assim cinco lugares.

Isto irá resultar numa atribuição do valor 5 ao primeiro lugar, 4 ao segundo, 3 ao terceiro, 2 ao quarto e 1 ao que se encontra em quinto lugar. De seguida verifica-se quantas vezes um determinado par pergunta-resposta aparece em cada um dos lugares nas várias listas de candidatos de cada pesquisa, multiplicando de seguida o número de vezes que ocorre em cada lugar pelo valor de cada lugar, isto é, se um par aparece quatro vezes em primeiro lugar e três vezes em segundo lugar nas varias listas de candidatos, este par pergunta-resposta terá 32 pontos ($4 \times 5 + 3 \times 4$).

Na tabela 5.2, é apresentado um exemplo real do nosso sistema, em que a pesquisa do utilizador foi "Pretendo arrendar um imóvel o que devo fazer?", como se vê na tabela foram encontrados cinco documentos candidatos que apareceram nas

Doc ID	Ocorrencias	Soma	1 lugar	2 lugar	3 lugar	4 lugar	5 lugar
112	7	**29**	3	2	2	0	0
113	7	**25**	2	1	3	1	0
156	7	**21**	0	3	1	3	0
105	5	**11**	0	1	0	3	1
65	4	**6**	0	0	1	0	3

TABELA 5.2: Exemplo da soma do método BordaCount

várias pesquisas realizadas. O documento com o ID 112 apareceu em todas as sete pesquisas, sendo nelas três vezes em primeiro, duas vezes em segundo e duas vezes em terceiro, obtendo uma pontuação de 29 pontos ($3 \times 5 + 2 \times 4 + 2 \times 3 = 29$). Em segundo lugar ficou o documento com o ID 113, que apareceu nas sete pesquisas, sendo duas delas em primeiro lugar, uma em segundo lugar, três em terceiro lugar, e numa das pesquisas no quarto lugar. Isto resulta em uma pontuação de 25 pontos ($2 \times 5 + 1 \times 4 + 3 \times 3 + 1 \times 2 = 25$). O mesmo cálculo é aplicado aos outros três documentos candidatos.

Sendo que o documento com o ID 112 o de maior pontuação, ele será considerado o melhor candidato e utilizado como resposta.

Na figura 5.6 é possível verificar com um exemplo real do sistema desenvolvido a forma como este apresenta o par encontrado e as respostas alternativas.

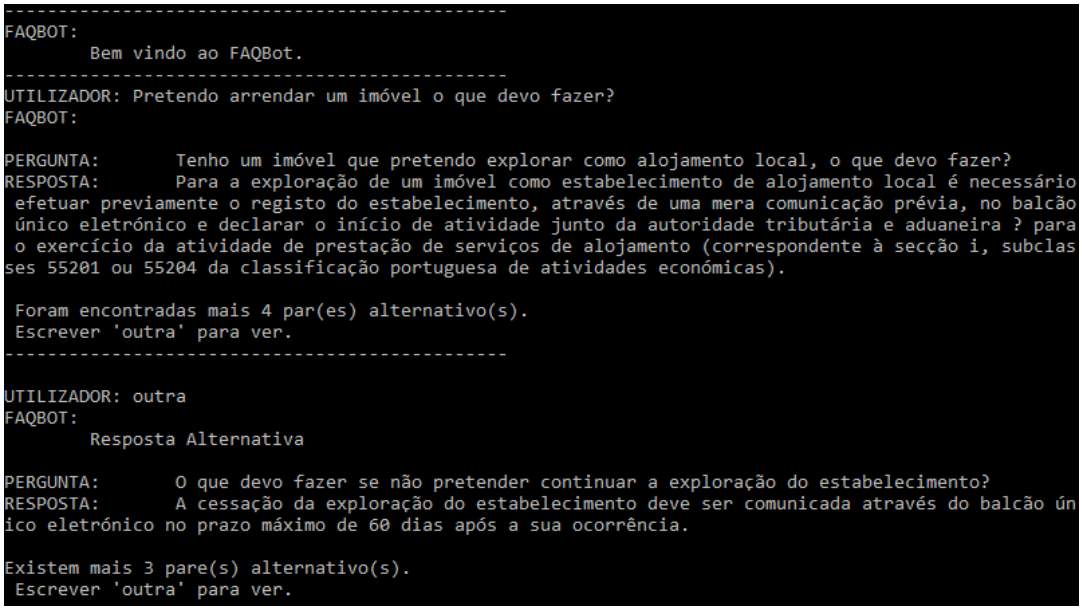


FIGURA 5.6: Exemplo Real do projeto FAQBot - Apresentação das Respostas Alternativas

De forma a poder entender melhor o exemplo apresentado na tabela 5.2, serão apresentados todos os pares candidatos segundo o nosso sistema para a pergunta do utilizador "Pretendo arrendar um imóvel o que devo fazer?", sendo eles:

1. **Doc ID:**112
    - **Pergunta:** Tenho um imóvel que pretendo explorar como alojamento local? O que devo fazer?

- **Resposta:** Para a exploração de um imóvel como estabelecimento de alojamento local é necessário efetuar previamente o registo do estabelecimento através de uma mera comunicação prévia ...

2. **Doc ID:**113
   - **Pergunta:** O que devo fazer se não pretender continuar a exploração do estabelecimento?
   - **Resposta:** A cessação da exploração do estabelecimento deve ser comunicada através do balcão único eletrónico no prazo máximo de 60 dias após a sua ocorrência.
3. **Doc ID:**156
   - **Pergunta:** Tenho um imóvel e gostaria de o ceder para exploração como alojamento local? o que devo fazer?
   - **Resposta:** Se é o proprietário e pretende que a prestação de serviços de alojamento no imóvel seja realizada por outra pessoa deve celebrar um contrato de arrendamento...
4. **Doc ID:**105
   - **Pergunta:** Pretendo fabricar gelados que procedimento devo efetuar?
   - **Resposta:** A fabricação de gelados enquadra-se no CAE 10520 pelo que o exercício dessa atividade está submetido ao regime jurídico do licenciamento industrial decreto-lei...
5. **Doc ID:**65
   - **Pergunta:** O que devo fazer para poder participar na feira da ladra?
   - **Resposta:** A participação de modo habitual na feira da ladra obriga à apresentação da mera comunicação prévia MCP no BDE após o que é emitido o comprovativo...

Após se ter o sistema a funcionar foi necessário decidir como o sistema iria lidar com as perguntas fora do domínio do sistema, ou seja, como é que iríamos lidar com as pesquisas do utilizador que o sistema não tem dados suficientes para apresentar uma resposta. Para isso foram pensadas em duas hipóteses para lidar com esse problema.

A primeira hipótese seria utilizar os scores da similaridade, como foi apresentado anteriormente. Cada correspondência entre o que o utilizador pesquisou e o que é encontrado na base de conhecimento tem atribuído um score de correspondência pela similaridade, sendo o candidato com maior correspondência o que terá maiores chances de ser o escolhido como sendo uma possível resposta a apresentar ao utilizador. Por isso foi pensado que a abordagem a um score mínimo fosse uma possível solução para lidar com perguntas fora do domínio, isto é, se nenhum dos candidatos obtivesse no mínimo um certo score não seria considerado como candidato.

Para tentar entender qual o valor que este score mínimo deveria ter, pensou-se em criar um novo conjunto de possíveis perguntas alternativas às respostas da base de conhecimento. Estas novas alternativas foram criadas com a ajuda de diferentes

voluntários, a quem foi pedido que analisassem os pares pergunta-resposta originalmente usados, e escrevessem a pergunta de formas distintas que, no entanto, mantivessem o significado original. Ao total foram criadas 401 novas perguntas alternativas e de seguida foram analisadas todas as pesquisas e verificado a exatidão do sistema, analisando os scores obtidos em cada correspondência.

É possível ver o gráfico desta análise na figura 5.7, onde para cada forma de pesquisa são associadas duas linhas no gráfico: uma linha azul que representa os scores das perguntas alternativas que conseguiram ser uma correspondência correta; e a linha vermelha que representa os scores quando não foi encontrada a correspondência correta.

Após se analisar o gráfico, entendeu-se que esta não seria uma boa abordagem, uma vez que existem scores muito baixos para certas perguntas alternativas e mesmo assim foi possível encontrar a correspondência correta e scores muito altos e a correspondência não é a correta, isto deve-se porque o score atribuído está diretamente relacionado com o numero de termos (palavras) que uma pesquisa tem, ou seja, uma pesquisa com poucos termos irá ter um score baixo em comparação a uma pesquisa com muitos termos que irá ter um score muito alto, sendo que em alguns casos uma correspondência com muitos termos, ter um score muito alto e ao mesmo tempo não ser o correto par a apresentar ao utilizador.

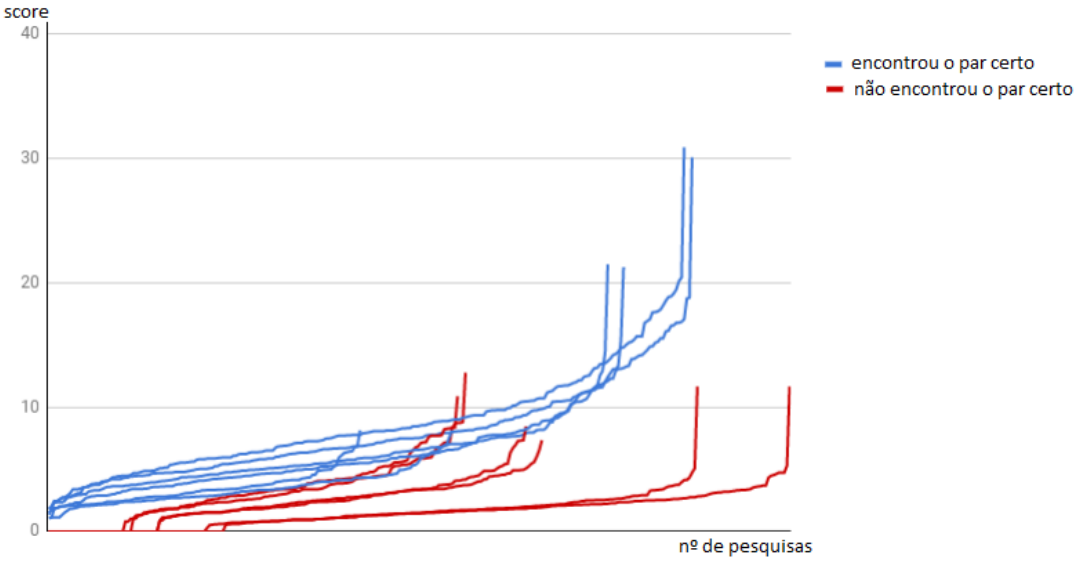


FIGURA 5.7: Gráfico dos scores de todas as pesquisas

Após se analisar que a abordagem pelo score mínimo não seria a melhor para lidar com as perguntas fora do domínio, passou-se para uma segunda abordagem. Esta segunda abordagem passou por considerar apenas candidatos a respostas os pares que aparecem no mínimo em metade das pesquisas.

Como foi dito no início desta secção, a primeira filtragem dos possíveis pares candidatos passa por utilizar sete métodos de pesquisa no Lucene, em que cada método irá encontrar cinco possíveis candidatos, obtendo no final sete listas com cinco pares candidatos cada.

Após se ter todas as listas, serão considerados como candidatos apenas os pares que se encontram no mínimo em metade das pesquisas, ou seja, em pelo menos 3

das listas devolvidas pelos métodos de pesquisas.

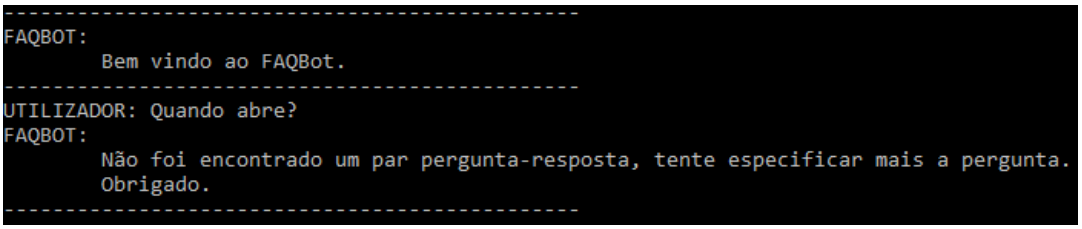


FIGURA 5.8: Exemplo Real do projeto FAQBot - Perguntas fora do domínio

Assim foi criado um novo filtro para a escolha do par a apresentar ao utilizador. Uma vez que, todas as pesquisas utilizam métodos diferentes serão criadas listas com diferentes pares e caso não se encontre o mesmo par pergunta-resposta em pelo menos metade das listas é considerado com uma pergunta fora do domínio e é apresentada ao utilizador a resposta "Não foi encontrado um par pergunta-resposta, tente especificar mais a pergunta. Obrigado", como é possível ver na figura 5.8, onde é apresentado um exemplo real do sistema desenvolvido a lidar com perguntas fora do domínio.

5.4 Testes

Nesta secção são apresentados alguns testes realizados com vista a fornecer indicações acerca do desempenho do sistema.

São apresentados três testes. O primeiro foca-se nos tempos de resposta do sistema. O segundo é uma avaliação da capacidade de apresentar a resposta correta. O último teste confirma a possibilidade de reutilização do sistema num domínio diferente do originalmente escolhido.

5.4.1 Desempenho Temporal

Um aspeto importante num sistema de conversação é a rapidez com que este consegue dar uma resposta ao utilizador. Para testar este aspeto, foram feitas algumas pesquisas e registado o tempo entre o momento em que o utilizador envia a sua pergunta até ao momento em que o sistema apresenta a resposta.

Na tabela 5.3, são apresentadas 10 pesquisas feitas consecutivamente e onde foi guardado o tempo de resposta, assim como o número de palavras que a cada pesquisa apresentava.

De forma explicar de uma forma mais clara o que é apresentado na tabela, são apresentados alguns dos exemplos das pesquisas efetuadas:

1. **Tempo de resposta:** 1 segundo; **Pesquisa:** Como posso realizar a mera comunicação prévia?
    - **Pergunta:**Sendo o titular da exploração tenho de ser eu a efetuar a mera comunicação prévia no balcão único eletrónico ou posso pedir a um terceiro que a faça?

Pesquisa	Tempo(segundos)	Nº palavras na pesquisa
1	1	7
2	<1	8
3	1	5
4	1	15
5	1	4
6	1	2
7	1	17
8	1	8
9	2	9
10	1	20

TABELA 5.3: Tempo de resposta do sistema

- **Resposta:** A submissão da mera declaração prévia no balcão único eletrónico pode ser efetuada por qualquer pessoa desde que mandatada pelo titular da exploração...

2. **Tempo de resposta:** 1 segundo; **Pesquisa:** "O que é um hostel"
   - **Pergunta:** O que é um hostel?
   - **Resposta:** O hostel é um estabelecimento de hospedagem cuja unidade de alojamento predominante é o dormitório...
3. **Tempo de resposta:** 1 segundo; **Pesquisa:** "A lista de preços deve estar visível e onde a devo afixar?"
   - **Pergunta:** Onde deve existir e o que deve constar da lista de preços?
   - **Resposta:** Devem existir listas de preços redigidas em português junto à entrada do estabelecimento e no seu interior...
4. **Tempo de resposta:** 1 segundo; **Pesquisa:** "O exercício de atividades de comércio a retalho à distância está sujeito a algum procedimento ou comunicação?"
   - **Pergunta:** O exercício de atividades de comércio a retalho à distância ao domicílio ou de forma automática está sujeito a algum procedimento ou comunicação?
   - **Resposta:** Não com a entrada em vigor do RJABSR este tipo de atividades deixa de estar sujeito a qualquer tipo de comunicação.

Como é possível verificar, o sistema tem um tempo médio de resposta de 1 segundo, o que vai ao encontro dos requisitos definidos. Um grande contributo para este desempenho é a capacidade de indexação e pesquisa da ferramenta Lucene.

5.4.2 Desempenho na Resposta Dada

Após a criação do sistema foi necessário testar a precisão do mesmo nas respostas dadas ao utilizador.

A realização deste teste baseou-se num conjunto de dados criado com a ajuda de diferentes pessoas, a quem foi pedido que analisassem alguns dos pares pergunta-resposta originalmente usados, e escrevessem a pergunta de formas alternativas que,

no entanto, mantivessem o significado original. Isto resultou num conjunto com 401 novas formas de fazer perguntas da base de conhecimento, associadas às suas respostas.

Com estas novas perguntas foi possível testar os sete métodos de pesquisa e perceber a sua taxa de sucesso na seleção da resposta correta, mesmo quando a pergunta não é feita exatamente como na base de conhecimento. Com estas novas perguntas foi possível analisar também a taxa de acerto da utilização do método BordaCount para escolher a resposta mais correta a dar ao utilizador.

1ª Filtragem	Taxa de acerto
PesquisaPergunta	0.61
PesquisaPerguntaLematizada	0.60
PesquisaPerguntaFuzzy	0.53
PesquisaPerguntaLematizadaFuzzy	0.54
PesquisaPerguntaResposta	0.67
PesquisaPerguntaLematizadaResposta	0.65
PesquisaPerguntaRespostaFuzzy	0.64
2ª Filtragem	**Taxa de acerto**
BordaCount	0.62

TABELA 5.4: Performance de cada filtragem na pesquisa

Os resultados obtidos encontram-se na tabela 5.4, onde se verifica para a primeira filtragem uma taxa de acerto média entre os 53% (obtidos com a forma de pesquisa PesquisaPerguntaFuzzy) e os 67% (com a PesquisaPerguntaResposta), é também demonstrado na tabela a taxa de acerto da filtragem final que utiliza o método BordaCount, método que como foi dito anteriormente, será o responsável por entre todas as listas de candidatos escolher a mais correta a apresentar ao utilizador, apresentando uma taxa de acerto de 62%.

Depois de uma análise aos resultados, entendeu-se que existem três fatores chave para que a taxa de acerto não seja mais elevada. São eles:

- **Perguntas muito complexas e específicas:** num ficheiro de perguntas e respostas, a pergunta deve ser simples e focada num único objetivo. No entanto, no ficheiro de FAQs do BDE existem muitas perguntas complexas, longas e demasiado específicas para serem pesquisadas, como é o exemplo da pergunta "Antes da entrada em vigor do decreto-lei n.º 128/2014, de 29 de agosto, já tinha registado na modalidade de apartamento no mesmo edifício mais de 9 frações autónomas, sendo este número superior a 75 porcento do número de frações existentes no edifício. Não se aplicando os limites previstos no artigo 11.º do dl 128/2014 posso registar mais algum apartamento do mesmo edifício?". Esta pergunta é muito longa, aborda muitos temas e é especifica para um determinado exemplo, o que torna difícil de encontrar uma correspondência feita por uma pesquisa do utilizador, acabando por nenhuma das sete pesquisas utilizadas ter encontrado a resposta correta a dar.

- **Perguntas que não têm contexto:** será difícil de obter uma correspondência se a pergunta em si não tiver o contexto do que está a ser perguntado, por exemplo as perguntas "Quais as oficinas compreendidas?" ou "É obrigatório afixar informações?" são perguntas cujo contexto é insuficiente para perceber

realmente o que se está a perguntar, sendo assim difícil de conseguir fazer a correspondência.

- **Respostas que abrangem mais do que é perguntado:** uma vez que algumas das novas perguntas alternativas foram criadas com base na resposta, seria muito difícil encontrar a resposta correta se a pergunta não referisse ao que é apresentado na resposta. Exemplos destas respostas são as que no final referem para outros pontos, como por exemplo às coimas aplicadas, ou identidades abrangidas ou requisitos necessários para criar algo que não foi perguntado previamente. Todos estes temas abordados nas respostas poderiam dar origem a novas perguntas, criando assim pares perguntas-respostas mais focadas num único contexto. Um exemplo deste caso é o caso da pergunta "Qual o regime para a identificação de veículos ligeiros que utilizam GPL ou GN como combustível?" com a resposta "Os modelos de vinhetas/dísticos identificadores constam da portaria n.º 196-b/2015, de 2 de julho. Compete às entidades que exercem as atividades de fabrico, adaptação e reparação de veículos movidos a GPL e GN disponibilizar os elementos de identificação dos veículos. Aplicação de sanções: por falta de identificação: de €60 a €300, no caso de pessoa coletiva, os montantes mínimo e máximo das coimas são elevados ao triplo.". Neste exemplo, a resposta abrange muitos mais temas do que os que são perguntados sendo difícil para um sistema deste género encontrar a resposta correta a dar ao utilizador.

A falta de precisão em relação à resposta dada à pergunta deveu-se ao facto de estas perguntas apesar de estarem num documento oficial da entidade denominadas de FAQ, não parecem ter sido devidamente analisadas, pois muitas delas são muito complexas e difícil para um utilizador procurar e esclarecer as suas dúvidas, causando alguns erros ao procurar num sistema baseado em perguntas e respostas.

5.4.3 Reutilização do Sistema

Um dos objetivos deste sistema é a sua possibilidade de ser reutilizado por novas entidades, apenas alterando o ficheiro das FAQs utilizado para criar a base de conhecimento.

Para confirmar que esse objetivo tinha sido atingido, o ficheiro com as FAQs do BDE foi substituído por uma lista de FAQs da operadora Vodafone [4]. Após a extração de 80 perguntas e respostas, foi criado um novo documento, utilizado na criação da base de conhecimento do sistema de conversação. Esse documento de texto segue a mesma estrutura já referida na secção 5.1.4, ou seja, cada pergunta é apresentada em linhas começadas por "P:", e a respetiva resposta aparece na linha seguinte, começada por "R:".

Para conseguir testar a taxa de sucesso do sistema, foram criadas perguntas alternativas às perguntas já presentes no documento, este após o teste conseguiu uma taxa de sucesso de 85%, conseguindo com as perguntas alternativas encontrar a resposta correta que se procurava. Esta elevada taxa deveu-se às perguntas que se encontram no documento de FAQs serem curtas e objetivas e em todas elas era possível identificar o contexto do que se estava a perguntar.

Verificou-se também que outros componentes como as stopwords, a base de sinónimos e os campos criados, estavam a ser devidamente utilizados, concluindo-se

[4] `https://ajuda.vodafone.pt/` (Dezembro 2018)

assim que o sistema continuou a funcionar corretamente mesmo tendo-se alterando o domínio em que este atua, nomeadamente o documento de FAQ que este utiliza.

Alguns exemplos para confirmar a sua nova utilização no novo domínio são por exemplo, estas novas pesquisas feitas, simulando o que um utilizador poderia perguntar. É apresentado um exemplo de pesquisa feita pelo utilizador e o par pergunta-resposta encontrado como sendo o melhor a apresentar:

1. **Pesquisa:** "Como aderir ao serviço Vodafone TV Net Voz"
   (a) **Pergunta:** Como posso aderir ao serviço Vodafone Tv Net Voz?
   (b) **Resposta:** Pode iniciar o seu pedido de adesão no nosso site para isso basta verificar a cobertura da sua morada aqui e indicar-nos o seu número de telemóvel...
2. **Pesquisa:** "Não sei o PIN, o que devo fazer"
   (a) **Pergunta:** Como sei o meu PIN inicial?
   (b) **Resposta:** Deve aceder ao my vodafone informação adicional caso não consiga aceder pode obter o PUK através do atendimento automático ligando 16 9 12 e seguindo as instruções.
3. **Pesquisa:** "Como posso pagar as faturas por referencia multibanco?"
   (a) **Pergunta:** Como pago as faturas por referencia multibanco?
   (b) **Resposta:** Pode obter as referências multibanco em na 1 página da fatura vodafone as referências multibanco são apresentadas na...

É possível verificar na figura 5.9, um exemplo de como o sistema desenvolvido lida com as novas perguntas feitas no novo domínio. Podendo assim afirmar que que este sistema funciona independentemente do ficheiro de FAQs inicial. Assim, é possível utilizar este sistema para uma entidade diferente daquela que foi inicialmente utilizada como base.

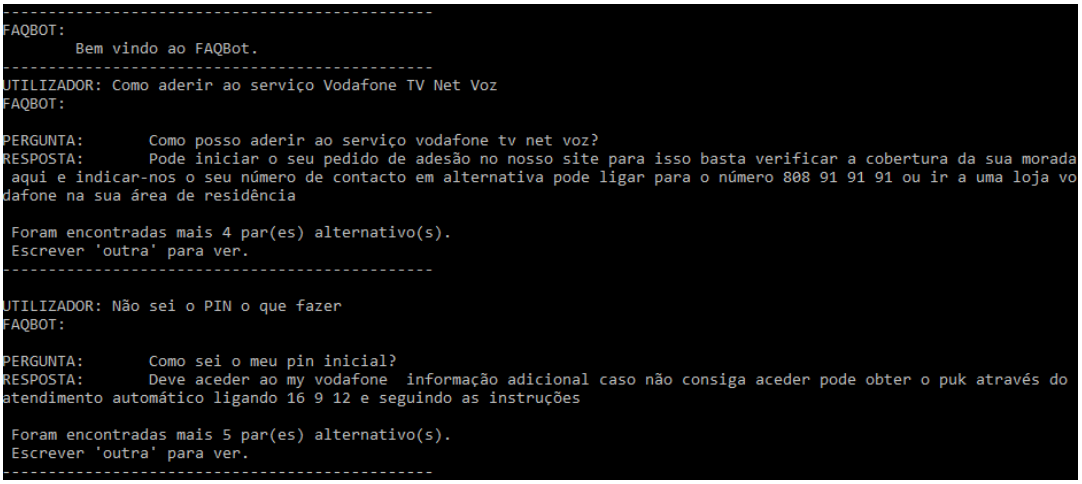


FIGURA 5.9: Exemplo Real do projeto FAQBot - Novo domínio Vodafone

Capítulo 6

Conclusões

Neste capítulo será apresentada a conclusão do trabalho elaborado. Inicialmente será feita uma revisão do tema proposto, em seguida são analisados os objetivos propostos e discutidos até que ponto foram atingidos. Posteriormente são apresentadas as contribuições deste trabalho, seguida pela apresentação das suas limitações e problemas encontrados. Para terminar, são apresentadas as direções para trabalhos futuros.

6.1 Revisão Geral do Tema

Muitas empresas e instituições têm nos seus sites um local onde são listadas perguntas e respostas frequentes (em inglês, FAQs). Estas devem incluir as principais dúvidas que os clientes e outros utilizadores possam ter relativamente à entidade em questão.

Apesar disso, a lista pode ser grande e o processo de esclarecimento da dúvida pode ser demorado, ou o utilizador pode simplesmente não conseguir encontrar a resposta que procurava, acabando por ter de enviar um email ao suporte de clientes, em que a resposta poderá demorar ainda mais tempo, ao ponto do utilizador poder perder o interesse na entidade.

Por isso, cada vez mais empresas apostam no desenvolvimento de um sistema conversacional para chegar de forma mais eficiente aos seus clientes. Este desenvolvimento é hoje em dia facilitado por algumas empresas como Google, Facebook, Microsoft, IBM e Amazon. Estas disponibilizam plataformas online capazes de simplificar o processo de criação de um sistema conversacional capazes de responder às perguntas do utilizador.

Apesar destas plataformas conseguirem criar sistemas conversacionais de uma forma simples, são muito difíceis de personalizar ficando o utilizador dependente da forma como elas fazem o seu processamento de linguagem natural (PLN) para obter a melhor resposta ou comportamento desejado. Para casos em que se pretende um maior controlo será necessário começar o desenvolvimento de um sistema conversacional de raiz ou, pelo menos, a partir de um nível mais baixo.

Neste trabalho começou-se por abordar várias formas para a criação de sistema conversacional capaz de entender a linguagem portuguesa. Estas passaram por inicialmente analisar as plataformas online que facilitam esta criação, de seguida os standards mais utilizados e por fim às ferramentas de pesquisa em textos.

Após esse estudo, foi desenvolvido um sistema de respostas automáticas a perguntas frequentes capaz de responder as perguntas do utilizador cuja resposta se encontra num documento de perguntas e respostas frequentes (FAQs). Este sistema é ainda versátil ao permitir responder a diferentes perguntas, em português, consoante o documento de FAQs que lhe é fornecido. Ou seja, poderá ser usado por qualquer entidade para a qual exista uma lista de FAQs.

6.2 Revisão dos Objetivos e Trabalho Realizado

Listam-se agora os objetivos inicialmente delineados e analisa-se o trabalho desenvolvido para alcançar cada um deles.

- O primeiro objetivo proposto neste trabalho foi o de analisar as várias formas disponíveis para se criar um sistema conversacional.

  Foram analisadas varias plataformas online disponíveis, desenvolvidas por grandes empresas, foram também analisados os standards mais utilizados para a representação de perguntas e respostas, e ferramentas de recuperação de informação.

  Na análise das plataformas começou-se por apresentar cinco das grandes empresas que disponibilizam plataformas online para a criação destes sistemas, nomeadamente DialogFlow (Google), Wit.ai (Facebook), LUIS (Microsft), Watson(IBM) e Amazon Lex (Amazon). Analisando cada uma, verifica-se que a maior limitação do uso destas plataformas é a dependência do uso das suas funções de processamento de linguagem natural (PLN). Apesar de cada uma destas plataformas disponibilizar um SDK, não é possível alterar o código fonte das funções de PLN.

  De seguida foram analisadas as formas de criar um sistema de conversação de raiz, que fosse baseado nos standards mais utilizados para este tipo de sistemas, sendo eles o AIML e Chatscript. Estes foram analisados conforme a sua curva de aprendizagem, a forma como fazem as sua pesquisas, a quantidade de programas disponíveis capazes de interpretar as suas regras, assim com as linguagens em que foram escritas e por fim o seu uso em sistemas do género mais recentes.

  A terceira análise foi dedicada ao Lucene, sendo esta uma ferramenta de Recuperação de Informação (RI). Esta análise foi importante pois parte do sistema de conversação proposto passa por analisar a informação nas perguntas e recuperar respostas, por isso o uso de uma ferramenta de RI de pesquisa de textos foi muito importante para a realização de um projeto deste género.

  Após a análise de todas estas plataformas e ferramentas, conclui-se que este objetivo foi devidamente atingido.

- O segundo objetivo deste trabalho era o de criar um sistema de conversação capaz de receber como fonte para a criação da sua base de conhecimento um documento de FAQs.

  Foi realizada uma análise de todas as plataformas e ferramentas para se construir um sistema deste tipo e, no final, foi escolhida a ferramenta de recuperação de informação Lucene. O sistema construído é capaz de receber uma FAQ

como fonte para a criação da sua base de conhecimento e responder às perguntas feitas pelo utilizador com um tempo de resposta médio de 1 segundo, independentemente da complexidade da pergunta feita, para além de conseguir entender erros ortográficos, como palavras escritas de forma errada ou com falta de acentuação gráfica. O sistema construído é também capaz de entender as pesquisas feitas pelo utilizador utilizando sinónimos. Assim sendo, conclui-se que este objetivo foi devidamente atingido.

- O último objetivo proposto para o trabalho foi que o sistema deverá ter uma arquitetura flexível, adaptável a outros domínios. Isto é, deverá ser possível alterar apenas o ficheiro das perguntas e respostas que alimenta a base de conhecimento do sistema por um outro de um domínio diferente e mesmo assim o sistema continuar a responder de forma natural às novas perguntas sobre o novo domínio. Uma vez que este trabalho foi realizado com o intuito de conseguir responder a perguntas feitas em língua portuguesa, espera-se que os ficheiros de perguntas e respostas utilizados neste sistema estejam escritos nessa língua.

  Para completar este objetivo foi necessário entender que o sistema não deverá ter nada específico do domínio em que vai atuar ligado à sua base do conhecimento. Dito isto, nenhuma variável nem ligações externas à entidade que fornecia o documento das FAQ foram criadas. O sistema foi proposto e implementado de forma independente da sua base de conhecimento e, para comprovar a sua capacidade de reutilização, foi alterado o documento dos pares perguntas e respostas por outras de uma entidade diferente e o sistema continuou o seu funcionamento normal, sendo capaz de responder de forma rápida e o mais correto possível às perguntas do utilizador sobre o novo domínio. Conclui-se que este objetivo foi devidamente atingido.

6.3 Principais contribuições

O trabalho desenvolvido tem como principais contribuições:

- Uma análise das plataformas online, assim como uma analise dos standards e ferramentas mais utilizadas, capazes de criar um sistema de conversação para a língua portuguesa.
- Uma arquitetura para um sistema conversacional para a língua portuguesa, capaz de filtrar stopwords, considerar sinónimos, capaz de tolerar erros ortográficos e falta de acentuação gráfica e ainda capaz de entender as pesquisas independentemente do seu género, número ou tempo verbal.
- Um sistema flexível capaz de ser utilizado por qualquer entidade empresarial, bastando que esta tenha uma lista de perguntas e respostas frequentes (FAQs).

Para além destas contribuições, é de referir que este trabalho deu origem a um artigo científico, recentemente submetido à conferência SLATE e aceite para publicação [1](Symposium on Languages, Applications and TEchnologies), este é um simpósio internacional dedicada à pesquisa do estudo da linguagem natural.

[1] `http://slate-conf.org/2019/hhl` (Dezembro 2018)

6.4 Limitações

Neste subcapítulo pretende-se identificar e apresentar algumas das limitações presentes no trabalho desenvolvido assim como os problemas encontrados para a criação de um sistema deste género.

Como o foco do projeto era a criação de um sistema conversacional para o português, uma das limitações deste projeto é a utilização de um documento de perguntas e respostas noutra língua. Este sistema não deve ser utilizado como um sistema conversacional em outra língua porque todo o processamento é feito por ferramentas (analisador, lematizador) e com base em recursos (stopwords, sinónimos) para esta língua.

Outra das limitações é a necessidade do documento de perguntas e respostas frequentes precisar de ter uma estrutura específica. Cada pergunta tem de começar com "P:" e cada resposta por "R:". Apesar de cada entidade ter uma forma de apresentar o documento das suas perguntas e respostas frequentes (FAQ), não é viável criar um algoritmo que consiga extrair os pares perguntas e respostas independentemente dessa estrutura.

Por fim, a última limitação encontrada é a incapacidade deste sistema estabelecer um diálogo com o utilizador, não tendo sido programado para guardar informações sobre o contexto de uma conversa, podendo assim estar a responder à mesma pergunta de forma igual inúmeras vezes não tirando partido do *feedback* do utilizador.

6.5 Problemas encontrados

Neste subcapítulo são apresentados os problemas encontrados no desenvolvimento do sistema de perguntas e respostas.

Na fase inicial em que foram analisadas ferramentas para a criação de um sistema conversacional, o maior problema foi a documentação encontrada para cada uma. Grande parte dessa documentação explica como criar um simples sistema de conversação, mas não aprofundava questões como alteração do seu método de processamento de linguagem natural (PLN) nem explicava os passos que a ferramenta percorria para a escolha da resposta ou comportamento do sistema. Foi despendido muito tempo a entender cada método da ferramenta de forma a conseguir criar o nosso próprio sistema de escolha da resposta.

Outro grande problema encontrado foi a qualidade dos dados e a importância que estes têm como fundamento para um sistema deste género. Uma vez que a base deste projeto é um documento com pares pergunta-resposta, é necessário que estes dados tenham uma certa qualidade, caso contrário o sistema poderá não ser capaz de atingir o seu objetivo final, que será ajudar o utilizador a responder às suas questões. Muitos dos pares de pergunta-resposta encontrados não estavam devidamente cuidados, sendo grande parte das perguntas muito complexas e específicas para um certo exemplo. Foram também encontradas perguntas sem o contexto e respostas que abrangem mais temas do que é perguntado, todas estes problemas foram apresentados no subcapítulo 5.4.2.

Todos estes pontos dificultam o processo de encontrar uma resposta adequada a dar ao utilizador. O ideal seria cada par ter uma pergunta específica, apresentando claramente o contexto do que está a ser perguntado, bem como uma resposta focada a esse contexto.

6.6 Direções para trabalhos futuros

Pretende-se agora apresentar algumas das questões que poderão ser abordadas futuramente, no seguimento deste trabalho.

Tendo em conta as limitações já referidas, um dos pontos chave para a construção deste sistema é o documento de perguntas e respostas frequentes (FAQs), por isso, este documento deve ter algum cuidado a ser construído. De forma a aumentar a qualidade desse documento, foram pensadas duas soluções: a primeira seria pedir à entidade que deseja utilizar o sistema de conversação que faça uma revisão ao documento, tendo em consideração que cada par deve ter uma pergunta curta que apresente claramente o contexto do que está a ser perguntado e a resposta focada nesse contexto; outra solução passa por uma parceria entre o programador e a entidade, em que seja possível esclarecer dúvidas com a entidade num período relativamente curto, de forma a criar um documento com maior qualidade.

Relativamente ao sistema, existe ainda uma falta de testes com utilizadores reais, podendo assim num trabalho futuro e em colaboração com a entidade que se proponha a utilizar o sistema a permissão de utilizar este sistema no seu website, permitindo que os seus clientes possam utilizar o sistema e avaliá-lo com base na ajuda por ele fornecida.

Por último, um dos aspetos que poderá ser relevante para o futuro do sistema é a utilização de aprendizagem automática (*machine learning*) de forma a se conseguir entender melhor os padrões dos utilizadores e conseguir entender o que os utilizadores realmente procuram nas suas pesquisas, criando assim um sistema mais robusto e capaz de ajudar mais eficazmente os utilizadores. Atualmente o sistema já consegue apresentar várias respostas alternativas à resposta apresentada ao utilizador, estas respostas podem ser apresentadas ao utilizador caso ele deseje e entenda que o par apresentado não seria o mais correto para lhe esclarecer a dúvida. Um dos pontos em que se poderia melhorar, a utilização de *machine learning*, seria treinar o sistema para adaptar as métricas de similaridade atualmente utilizadas consoante o *feedback* do utilizador, melhorando a taxa de acerto no primeiro par pergunta-resposta apresentado.

Bibliografia

Abdul-Kader, Sameera e John Woods (2015). «Survey on Chatbot Design Techniques in Speech Conversation Systems». Em: International Journal of Advanced Computer Science e Applications.

Ameixa, David et al. (2014). «Luke, I am Your Father: Dealing with Out-of-Domain Requests by Using Movies Subtitles». Em: *Intelligent Virtual Agents*. Ed. por Timothy Bickmore, Stacy Marsella e Candace Sidner. Cham: Springer International Publishing, pp. 13–21. ISBN: 978-3-319-09767-1.

Batacharia, B. et al. (1999). «CONVERSE: a Conversational Companion». Em: *Machine Conversations*. Ed. por Yorick Wilks. Boston, MA: Springer US, pp. 205–215. ISBN: 978-1-4757-5687-6. DOI: 10.1007/978-1-4757-5687-6_17. URL: https://doi.org/10.1007/978-1-4757-5687-6_17.

Braun, Daniel et al. (2017). «Evaluating Natural Language Understanding Services for Conversational Question Answering Systems». Em: *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*. Saarbrücken, Germany: Association for Computational Linguistics, pp. 174–185. URL: http://aclweb.org/anthology/W17-5522.

Canonico, Massimo e Luigi De Russis (2018). «A Comparison and Critique of Natural Language Understanding Tools». Em: CLOUD COMPUTING 2018 : The Ninth International Conference on Cloud Computing, GRIDs, e Virtualization.

Colby, Kenneth M (1973). «Simulations of belief systems». Em: *Computer models of thought and language*, pp. 251–286.

Dutta, Diptavo (2017). «Developing an Intelligent Chat-bot Tool to assist high school students for learning general knowledge subjects». Em:

Emerson, Peter (2013). «The original Borda count and partial voting». Em: *Social Choice and Welfare* 40.2, pp. 353–358. ISSN: 1432-217X. DOI: 10.1007/s00355-011-0603-9. URL: https://doi.org/10.1007/s00355-011-0603-9.

Geethanjali e Birunda Antoinette Mary J (2017). «Towards Building a Competent Chatbot – An Analogy of Development Framework, Design Techniques and Intelligence». Em: International Journal of Innovative Research in Science, Engineering e Technology.

Giakoumakou, Vasiliki (2018). «Development of a web application for an automated user assistant». Em: International Hellenic University.

Gonçalo Oliveira, Hugo (2018). «A Survey on Portuguese Lexical Knowledge Bases: Contents, Comparison and Combination». Em: *Information* 9.2. ISSN: 2078-2489. DOI: 10.3390/info9020034. URL: http://www.mdpi.com/2078-2489/9/2/34.

Gregori, Eric (2017). «Evaluation of Modern Tools for an OMSCS Advisor Chatbot». Em: Georgia Institute of Technology Atlanta, Georgia, USA.

Heller, Bob et al. (2005). «Freudbot: An Investigation of Chatbot Technology in Distance Education». Em: *Proceedings of EdMedia + Innovate Learning 2005*. Ed. por Piet Kommers e Griff Richards. Montreal, Canada: Association for the Advancement of Computing in Education (AACE), pp. 3913–3918. URL: https://www.learntechlib.org/p/20691.

Ingason, Anton Karl et al. (2008). «A Mixed Method Lemmatization Algorithm Using a Hierarchy of Linguistic Identities (HOLI)». Em: *Advances in Natural Language Processing*. Ed. por Bengt Nordström e Aarne Ranta. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 205–216. ISBN: 978-3-540-85287-2.

Ji, Zongcheng, Zhengdong Lu e Hang Li (2014). «An Information Retrieval Approach to Short Text Conversation». Em: *CoRR* abs/1408.6988. arXiv: 1408.6988. URL: http://arxiv.org/abs/1408.6988.

Jurafsky, Daniel e James H. Martin (2009). *Speech and Language Processing (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc. ISBN: 0131873210.

Kolomiyets, Oleksandr e Marie-Francine Moens (2011). «A survey on question answering technology from an information retrieval perspective». Em: Elsevier.

Krantz, Amandus e Petrus Lindblom (2017). «Generating Topic-Based Chatbot Responses». Em: Faculty of Computing Blekinge Institute of Technology, Karlskrona, Sweden.

Liddy, Elizabeth D. (2001). «Natural Language Processing». Em:

McCandless, Michael, Erik Hatcher e Otis Gospodnetic (2010). *Lucene in Action, Second Edition: Covers Apache Lucene 3.0*. Greenwich, CT, USA: Manning Publications Co. ISBN: 1933988177, 9781933988177.

Mell, Peter e Tim Grance (2011). «The NIST definition of cloud computing». Em: Computer Security Division, Information Technology Laboratory, National Institute of Standards e Technology.

Niculescu, A. I. e Rafael E. Banchs (2015). «Strategies to cope with errors in human-machine spoken interactions: using chatbots as back-off mechanism for task-oriented dialogues». Em: *ERRARE 2015 - Errors by Humans and Machines in multimedia, multimodal and multilingual data processing*.

Pedro Fialho Lúısa Coheur, Sergio Curto Pedro Cláudio Ángela Costa Alberto Abad Hugo Meinedo e Isabel Trancoso (2013). «Meet EDGAR, a tutoring agent at MONSERRATE». Em: Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL.

Quarteroni, S. e S. Manandhar (2009). «Designing an interactive open-domain question answering system». Em: vol. 15. 1. Cambridge University Press, 73—95. DOI: 10.1017/S1351324908004919.

Risso, Mercedes Sanfelice e Vandersí Pereira Sant'Ana (1973). «Variabilidade na língua». Em: *Revista de Letras* 15, pp. 195–204. ISSN: 01013505. URL: http://www.jstor.org/stable/27666188.

Robertson, Stephen (2004). «Understanding inverse document frequency: on theoretical arguments for IDF». Em: vol. 60. 5, pp. 503–520. DOI: 10.1108/00220410410560582. eprint: https://doi.org/10.1108/00220410410560582. URL: https://doi.org/10.1108/00220410410560582.

Rodrigues, Ricardo, Hugo Gonçalo Oliveira e Paulo Gomes (2018). «NLPPort: A Pipeline for Portuguese NLP (Short Paper)». Em: *7th Symposium on Languages, Applications and Technologies (SLATE 2018)*. Ed. por Pedro Rangel Henriques et al. Vol. 62. OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 18:1–18:9. ISBN: 978-3-95977-072-9. DOI: 10.4230/OASIcs.SLATE.2018.18. URL: http://drops.dagstuhl.de/opus/volltexte/2018/9276.

Shawar, Bayan Abu e Eric Atwell (2007). «Different Measurements Metrics to Evaluate a Chatbot System». Em: *Proceedings of the Workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technologies*. NAACL-HLT-Dialog '07. Stroudsburg, PA, USA: Association for Computational Linguistics, pp. 89–96. URL: http://dl.acm.org/citation.cfm?id=1556328.1556341.

Turing, A. M. (1950). «I.—COMPUTING MACHINERY AND INTELLIGENCE». Em: *Mind* LIX.236, pp. 433–460. DOI: 10.1093/mind/LIX.236.433. eprint: /oup/backfile/content_public/journal/mind/lix/236/10.1093_mind_lix.236.433/1/433.pdf. URL: http://dx.doi.org/10.1093/mind/LIX.236.433.

Wallace, Richard S (2009). «The anatomy of ALICE». Em: *Parsing the Turing Test*. Springer, pp. 181–210.

Weizenbaum, Joseph (1966). «ELIZA - a computer program for the study of natural language communication between man and machine.» Em: *Commun. ACM* 9.1, pp. 36–45. URL: http://dblp.uni-trier.de/db/journals/cacm/cacm9.html#Weizenbaum66.

Yu, Jianfei et al. (2018). «Modelling Domain Relationships for Transfer Learning on Retrieval-based Question Answering Systems in E-commerce». Em: *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*. WSDM '18. New York, NY, USA: ACM, pp. 682–690. ISBN: 978-1-4503-5581-0. DOI: 10.1145/3159652.3159685. URL: http://doi.acm.org/10.1145/3159652.3159685.

Zhao Yan Nan Duan, Junwei Bao Peng Chen Ming Zhou Zhoujun Li Jianshe Zhou (2016). «Docchat: An information retrieval approach for chatbot engines using unstructured documents». Em: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers).

Apêndice A

Proposta de estágio

PROPOSTA DE PROJETO

MESTRADO em INFORMÁTICA E SISTEMAS
Especialização em Desenvolvimento de Software

Ano Letivo de 2017/2018

TEMA

FAQBot.PT: sistema de conversação e resposta a perguntas em português

Palavras-chave: Processamento da Linguagem Natural (PLN), Resposta Automática a Perguntas, Extração de Informação, Recuperação de Informação

1. ÂMBITO

Seja em serviços de apoio ao cliente, comércio eletrónico, ou para outros tipos de consulta, agentes inteligentes artificiais com os quais é possível interagir através da linguagem humana são cada vez mais comuns, o que explica também a procura cada vez maior deste tipo de sistemas pelas mais variadas entidades.

Ainda que alguns tenham a capacidade de desenvolver qualquer tipo de conversa, os *chatbots* estão normalmente focados num domínio ou conjunto finito de perguntas a que conseguem responder. Por exemplo, podem servir como interface alternativo para uma base de dados ou *web service*.

2. OBJECTIVOS

Este trabalho tem como principal objetivo desenvolver um agente conversacional que suporte interação em português.

Para além de interações genéricas, pretende-se que ele consiga responder a perguntas sobre um determinado domínio.

Contudo, ele deve ter uma arquitetura o mais genérica possível, que permita usar o mesmo sistema para responder a perguntas sobre diferentes domínios.

O conhecimento acerca do domínio deve ser fornecido, numa primeira fase, através de uma lista de perguntas-já-respondidas (FAQs) e, no futuro, através de outras fontes de conhecimento, como, por exemplo, texto acerca do domínio.

Referir ainda que há várias plataformas sobre as quais já é possível desenvolver um sistema deste tipo, no entanto há dúvidas relativamente à sua flexibilidade com vista ao objetivo pretendido (mesma arquitetura para diferentes domínios / listas de FAQs), e à sua qualidade para lidar com o português.

Considerando a experiência dos orientadores em PLN, nomeadamente do português, interessa estudar até que ponto interessa desenvolver o sistema a partir de uma das plataformas disponíveis ou implementar algo de raiz, ainda que reutilizando várias ferramentas e recursos que temos desenvolvidos para português.

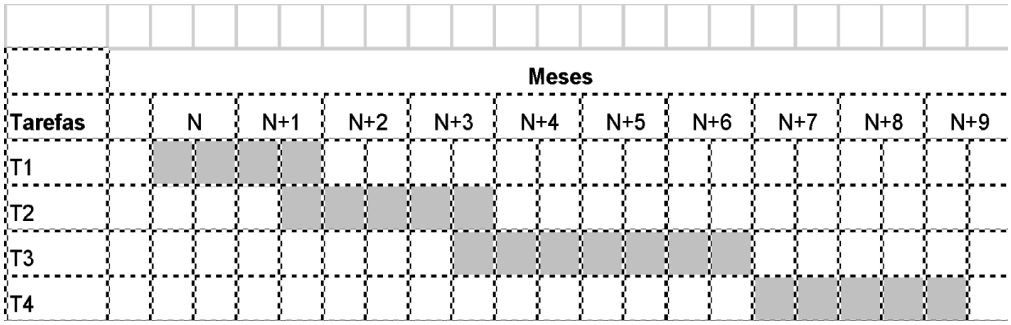
3. PROGRAMA DE TRABALHOS

O Projeto consistirá nas seguintes atividades e respetivas tarefas:

- T1 – Levantamento do estado da arte;
- T2 – Análise de requisitos de um sistema de conversação simples (v1) onde o conjunto de perguntas e respostas é previamente fornecido e métricas básicas de similaridade entre perguntas;
- T3 – Desenvolvimento do sistema (v1) e testes;
- T4 – Análise de requisitos do módulo extração de informação a partir de texto (mais avançado - V2) e utilização de métricas de similaridade semântica entre perguntas;
- T5 – Desenvolvimento do sistema (v2) e testes;
- T6 - Elaboração do relatório final e escrita de artigo científico

4. CALENDARIZAÇÃO DAS TAREFAS

As Tarefas acima descritas, incluindo os testes de validação de cada módulo, serão executadas de acordo com a seguinte calendarização:



O plano de escalonamento dos trabalhos é apresentado em seguida:

INI		Início dos trabalhos
M1	(INI + 2 meses)	Tarefa T1 terminada
M2	(INI + 4,5 meses)	Tarefa T2 terminada
M3	(INI + 9,5 meses)	Tarefa T3 terminada
M4	(INI + 10 meses)	Tarefa T4 terminada

5. RESULTADOS

Os resultados do estágio serão consubstanciados num conjunto de documentos a elaborar pelo aluno de acordo com o seguinte plano:

M1:

Relatório com o estado da arte

M2:

Relatório técnico com o levantamento de requisitos do sistema de conversação;

M3:

Relatório técnico com a descrição do sistema e plano de testes;

M4:

Relatório final de projeto e artigo

6. LOCAL DE TRABALHO

DEIS

7. METODOLOGIA

Organização de um Dossier de Projeto na *cloud* e reuniões quinzenais.

8. ORIENTAÇÃO

Ana Cristina Oliveira Alves (aalves@isec.pt)
Professora Adjunta

Hugo Oliveira (hroliv@dei.uc.pt)
Professor Auxiliar - DEI/FCTUC

Orientando: Ricardo Ferreira Filipe (a21250815@alunos.isec.pt)
Aluno do Mestrado de Informática e de Sistemas

9. CARACTERIZAÇÃO

- Data de início: Dezembro de 2017
- Data de fim: Setembro de 2018

Apêndice B

Tabela dos vencedores do Prémio Loebner

Ano	Nome do Sistema	Autor	Técnicas Utilizadas
1991	PC:Therapist	Joseph Weintraub	Respostas guardadas sem qualquer sequencia, depois de analisar as frases palavra por palavra utilizada correspondência por padrões
1992	PC:Therapist	Joseph Weintraub	Mesmo que em 1991
1993	PC:Therapist	Joseph Weintraub	Mesmo que em 1991
1994	TIPS	Thomas Whalen	Utiliza correspondência por padrões
1995	PC:Therapist	Joseph Weintraub	Mesmo que em 1991
1996	HeX	Jason Hutchens	Usou correspondência de padrões, modelo de cadeia Markov e usa uma base de dados para guardar todas as respostas anteriores
1997	Converse	David Levy	Usou um Wordnet com sinónimos,correspondência de padrões, uma base de dados com factos, uma lista de nomes próprios, e um modelo de pesos para cada resposta
1998	Albert One	Robby Garner	Utilizou uma estrutura parecida com alguns chatbots conhecidos como o Eliza e correspondência de padrões
1999	Albert One	Robby Garner	Mesmo que em 1998
2000	A.L.I.C.E	Richard Wallace	Correspondência de padrões, usou AIML.
2001	A.L.I.C.E	Richard Wallace	Mesmo que em 2000.
2002	Ella	Kevin Copple	Normalização das frases, correspondência de padrões, Wordnet e uma base de dados de abreviações.
2003	Jabberwock	Juergen Pirner	Cadeia de Markov, correspondência de padrões e GLC (Gramática livre de contexto).
2004	A.L.I.C.E	Richard Wallace	Mesmo que em 2000.
2005	George	Rollo Carpenter	Sem correspondência de padrões, apenas uma enorma base de dados de respostas em linguagem natural, baseado no chatbot Jabberwacky.
2006	George	Rollo Carpenter	Mesmo que em 2005.
2007	UltraHAL	Robert Medeksza	Scripts de correspondência de padrões combinados com código VB.
2008	Elbot	Fred Roberts	Sistema de interação comercial em linguagem natural.
2009	Do-Much-More	David Levy	Propriedade Comercial de Brinquedos Inteligentes.
2010	Suzette	Bruce Wilcox	Baseado em AIML, com uma base de dados de triplos, variáveis e conceitos
2011	Rosette	Bruce Wilcox	Mesmo que em 2010
2012	Chip Vivant	Mohan Embar	ChatScript e inteligência artificial
2013	Mitsuku	Steve Worswick	Baseado em AIML.
2014	Rose	Bruce Wilcox	Contem uma engenharia de compreensão da linguagem natural para descobrir o sentido da frase e ChatScript.
2015	Rose	Bruce Wilcox	Contem uma engenharia de compreensão da linguagem natural para descobrir o sentido da frase e ChatScript.
2016	Mitsuku	Steve Worswick	Baseado em AIML.
2017	Mitsuku	Steve Worswick	Baseado em AIML.
2018	Mitsuku	Steve Worswick	Baseado em AIML.

TABELA B.1: Tabela dos vencedores do Prémio Loebner

Apêndice C

Diagramas e Mockups dos Requisitos Funcionais

Neste anexo serão apresentados os diagramas de atividades assim como os mockups dos requisitos funcionais apresentados no subcapítulo 4.1.1.

A representação dos mockups, não corresponde ao aspeto visual do sistema, apenas representa de uma forma clara como o requisito funcional deve atuar.

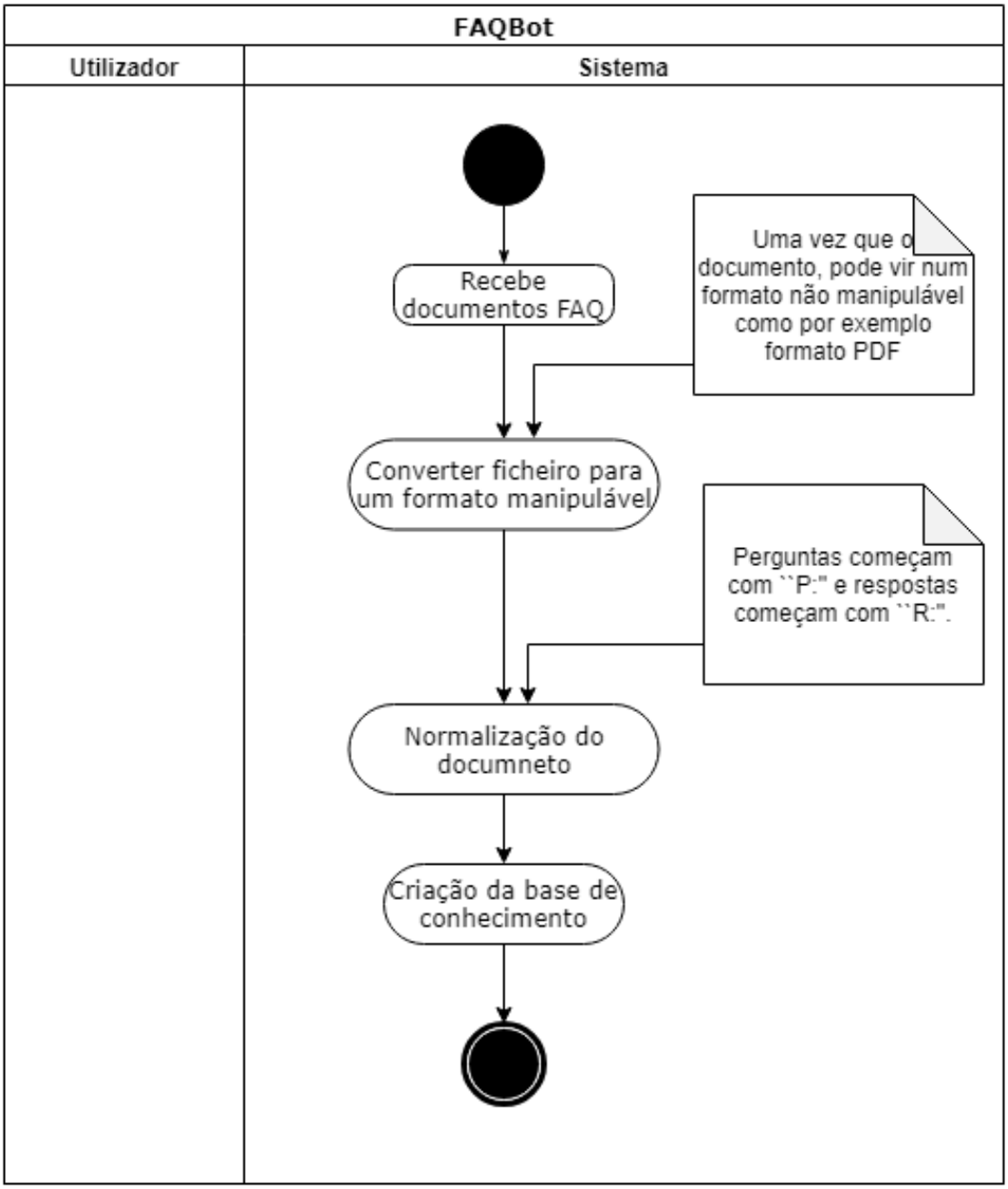


FIGURA C.1: Diagrama de atividade do requisito funcional - Obter os dados de uma FAQ

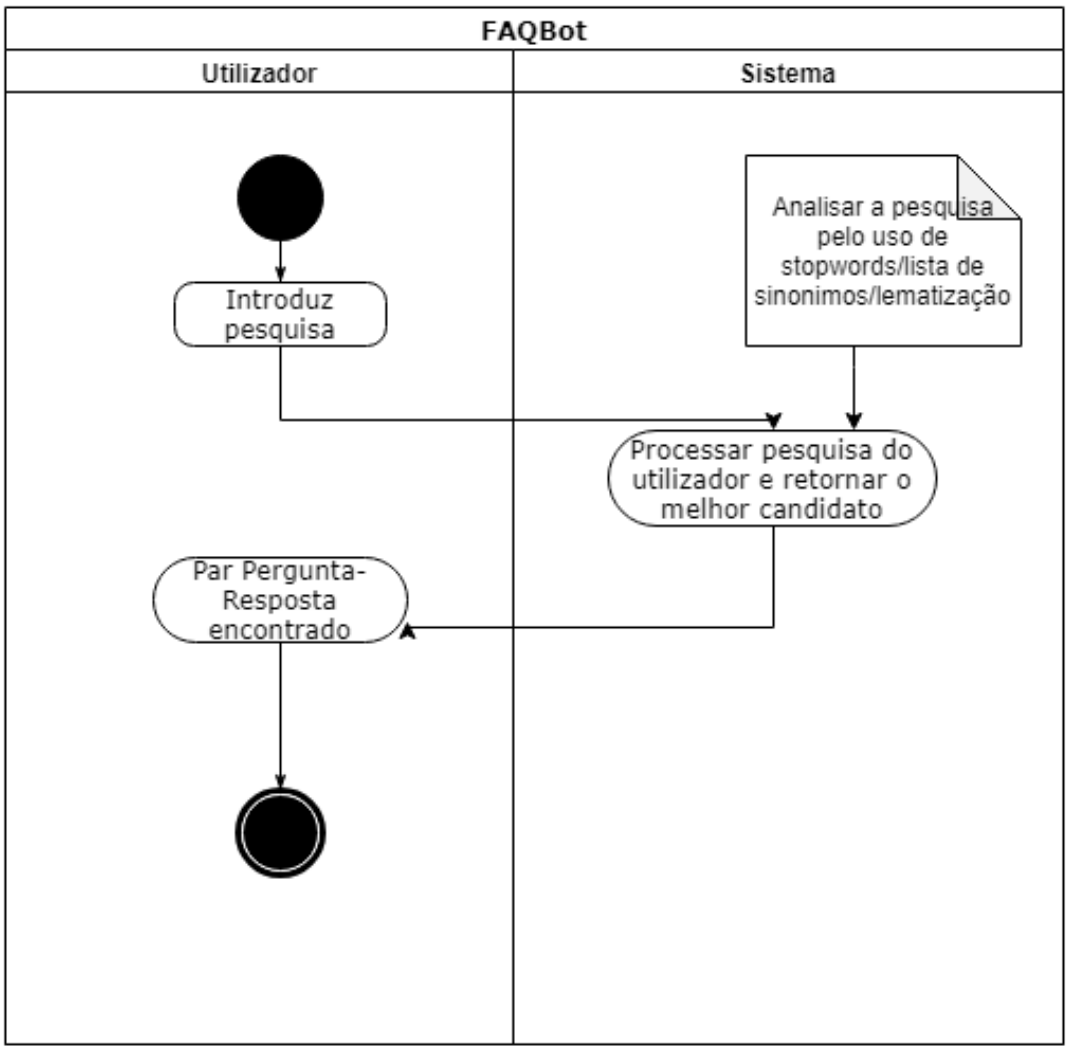


FIGURA C.2: Diagrama de atividade do requisito funcional - Capacidade de apresentar o par pergunta-resposta mais provável

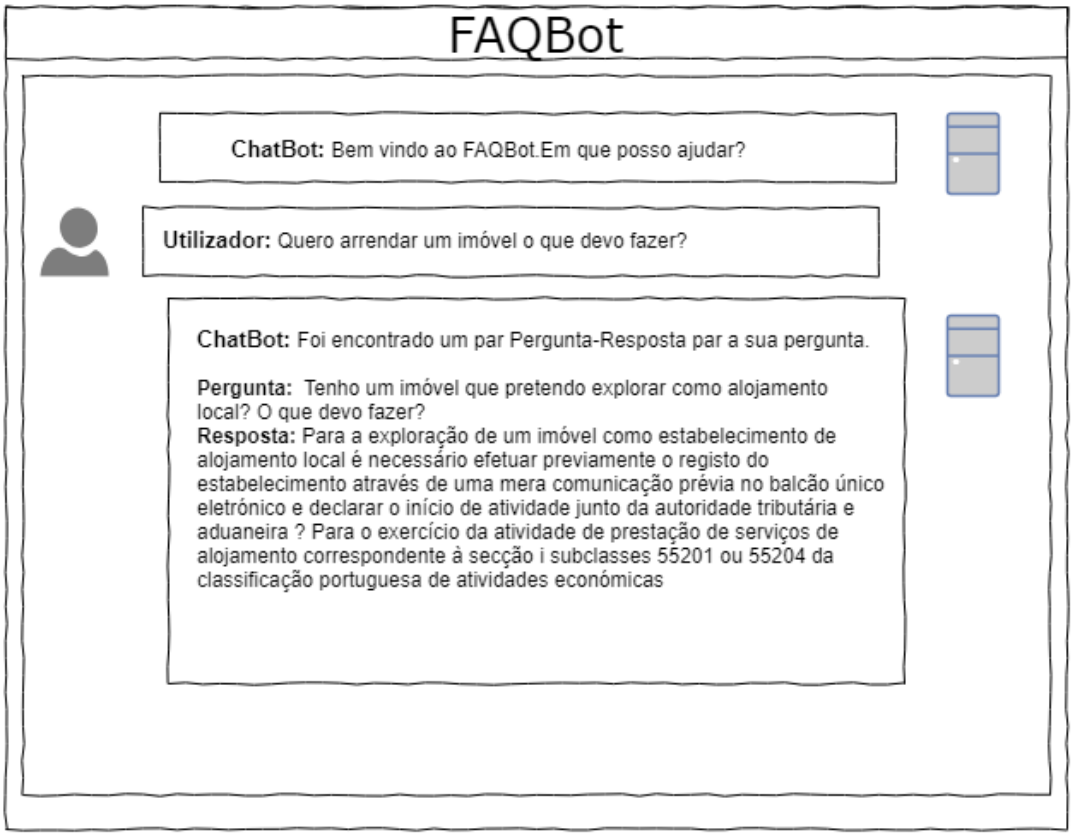


FIGURA C.3: Mockup do requisito funcional - Apresentar outros pares pergunta-resposta alternativas

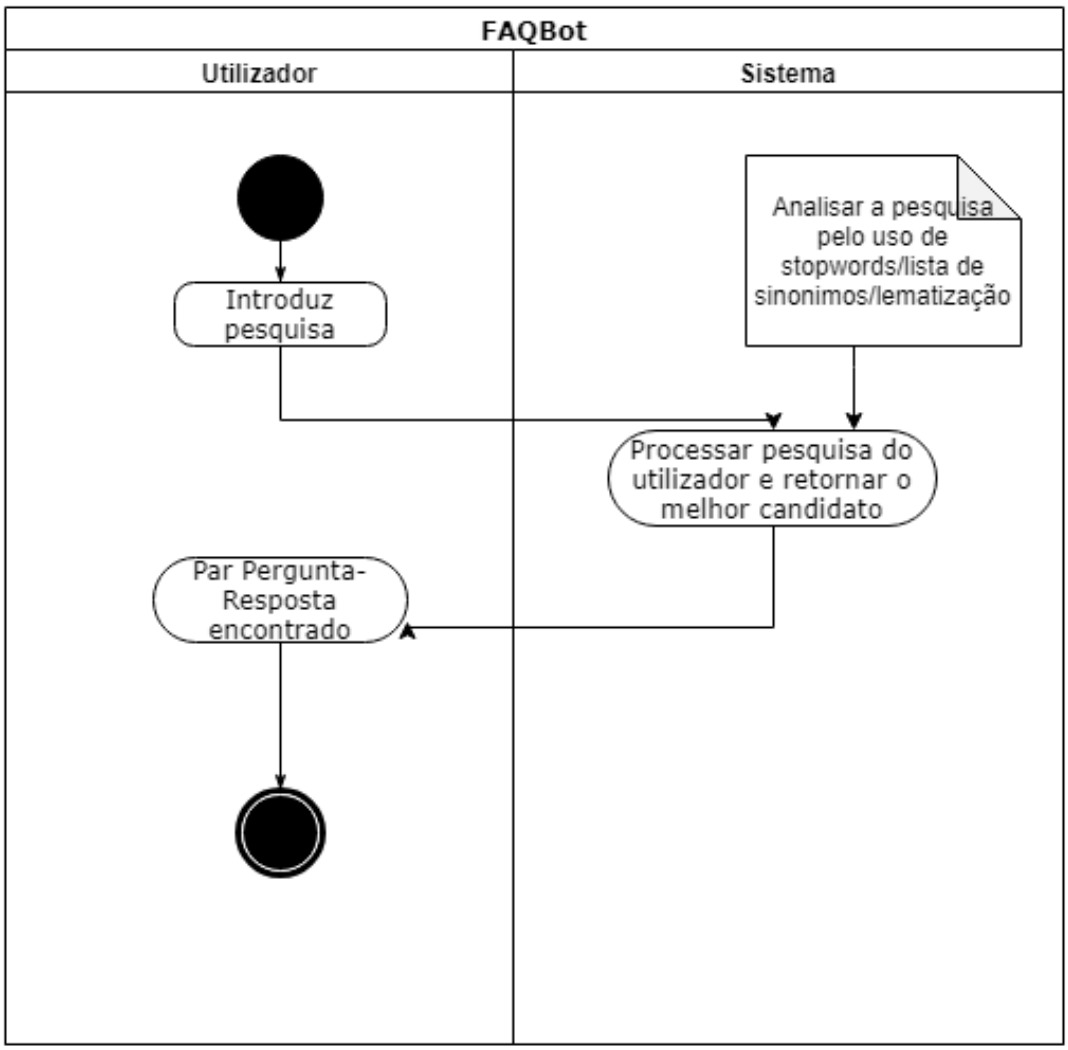


FIGURA C.4: Diagrama de atividade do requisito funcional - Capacidade de apresentar o par pergunta-resposta mais provável

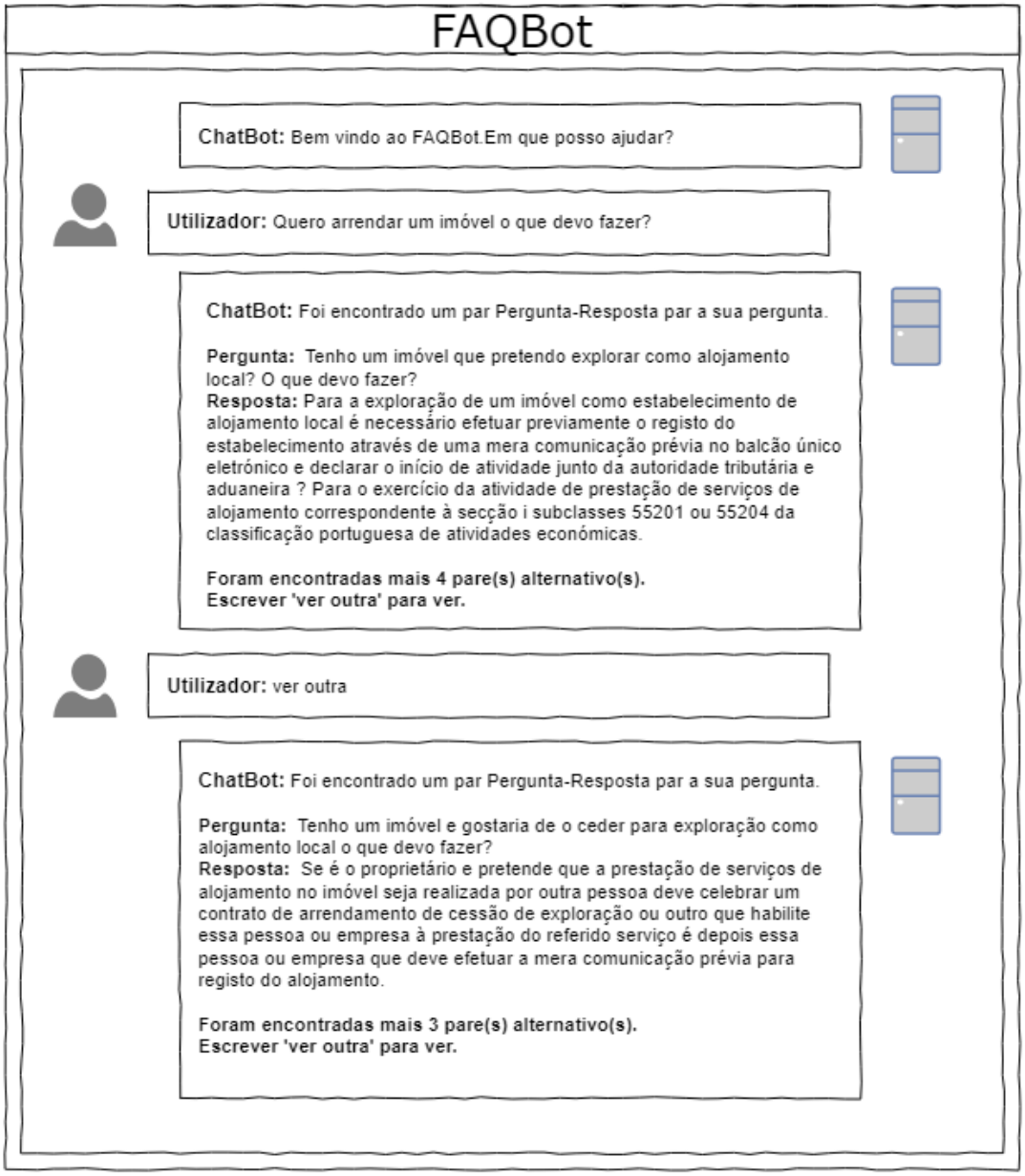


FIGURA C.5: Mockup do requisito funcional - Apresentar outros pares pergunta-resposta alternativas

Apêndice D

Email do DialogFlow

Titlo: Try to change the Match in the intents

Hi,

Sorry, Dialogflow does not provide such functionality at this moment and we cannot disclose any information about our machine learning algorithms and NLU techniques.

Regards, Akash from Dialogflow

————— Original Message —————

Sent: 5/4/2018 5:34 PM

Subject: Re: Try to change the Match in the intents

Thank you for the reply.

But what i'm asking it's if there is a way to see the algorithm that dialogflow use to do the match and change that.

2018-05-04 0:17 GMT+01:00 Dialogflow Support <support@dialogflow.com>:

————————————————

Hi,

You can improve matching results by setting a higher value for the ML Classification Threshold, as described at `https://dialogflow.com/docs/machine-learning#ml_classification_threshold`. Please note that after setting a higher value, you may need to add more examples to the intents. Alternatively, you can create a special intent for capturing words/phrases that are matched incorrectly. In this intent, you can defined the same response that you have in the fallback intent.

You can also consider to set intent priority which allows you to assign more weight to one of the intents in case an input phrase matches multiple intents. For more information, please refer our docs here: `https://dialogflow.com/docs/intents#intents_priority`.

I hope this information helps you. Let me know if you have any questions.

Regards, Akash from Dialogflow

————— Original Message —————

Sent: 5/3/2018 1:26 PM

Subject: Try to change the Match in the intents

Hello,

My name is Ricardo, can you please tell me if is possible alter the way the DiologFlow make the match of the userSays. I'm using the API in my Java project.

There is one way to change the match?

Apêndice E

Diagrama de Sequência do Sistema

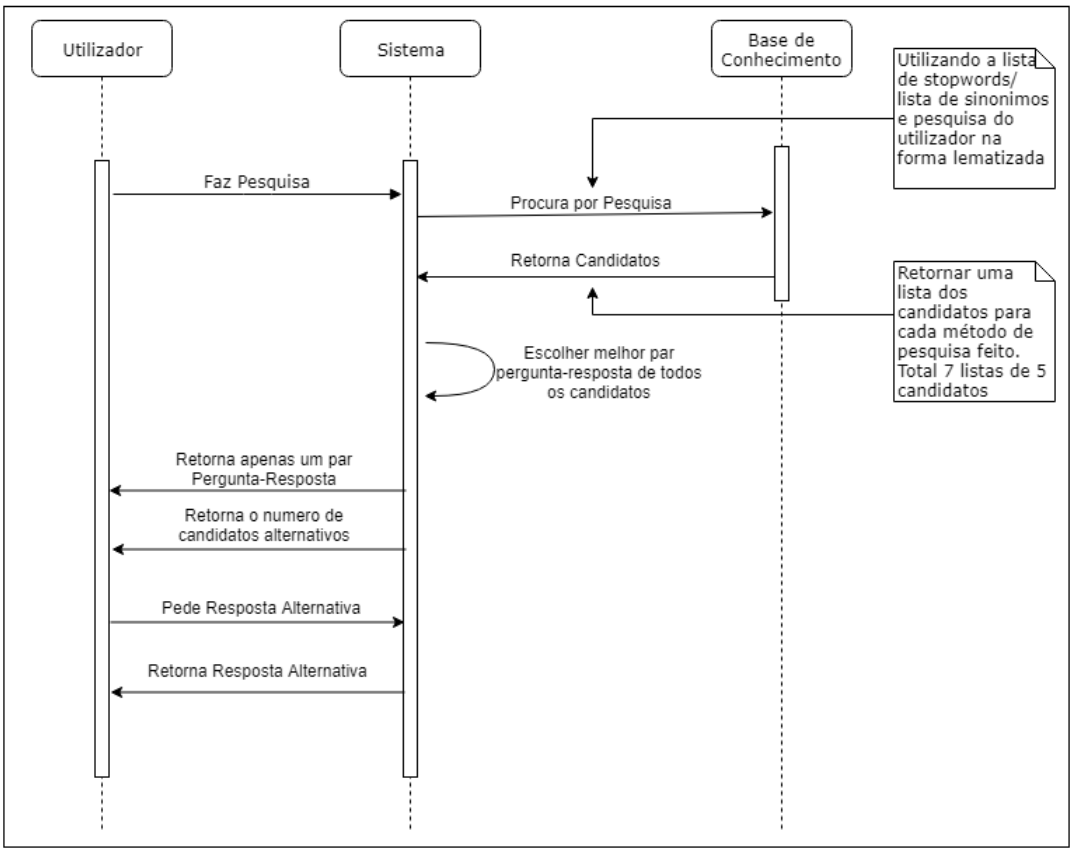


FIGURA E.1: Diagrama de sequência do sistema

Apêndice F

Código Fonte do Analisador

```
private static Analyzer CreateAnalyzer(final SynonymMap synonymMap,final CharArraySet StopWords) {

    Analyzer analyzer = new Analyzer() {

        @Override
        protected TokenStreamComponents createComponents(String fieldName) {

            org.apache.lucene.analysis.Tokenizer source = new ClassicTokenizer();
            TokenStream filter = new StandardFilter(source);

            //Meter tudo em lowcase
            filter = new LowerCaseFilter(filter);
            //StopWords
            filter = new StopFilter(filter, StopWords);
            //Sinonimos
            SynonymMap mySynonymMap = synonymMap;
            filter = new SynonymGraphFilter(filter, mySynonymMap, false);
            //Stem Portuguese
            filter = new PortugueseLightStemFilter(filter);

            return new TokenStreamComponents(source, filter);
        }
    };

    return analyzer;
}
```

FIGURA F.1: Código fonte do analisador utilizado no projeto

Apêndice G

Código Fonte da Similaridade

```
private static Similarity CreateSimilarity() {

    TFIDFSimilarity similarity = new TFIDFSimilarity() {

        @Override
        public float tf(float freq) {
            //freq the frequency of a term within a document
            return (float)Math.sqrt(freq);
        }

        @Override
        public float sloppyFreq(int distance) {
            //distance the edit distance of this sloppy phrase match
            return 0;
        }

        @Override
        public float scorePayload(int doc, int start, int end, BytesRef payload) {

            //doc The docId currently being scored.
            //start The start position of the payload
            //end The end position of the payload
            //payload The payload byte array to be scored
            return 0;
        }

        @Override
        public float lengthNorm(int length) {
            //length: the number of terms in the field, optionally discounting overlaps
            return ((float) (1.0 / Math.sqrt(length)));
        }

        @Override
        public float idf(long docFreq, long docCount) {
            //docFreq the number of documents which contain the term
            //docCount the total number of documents in the collection
            return (float)(Math.log((docCount+1)/(double)(docFreq+1)) + 1.0);
        }

        public String toString() {
            return "Custom TFIDFSimilarity";
        }
    };

    return similarity;
}
```

FIGURA G.1: Código fonte da similaridade utilizada no projeto