INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




Chatbot para apoio à Divisão Académica

JOÃO MIGUEL RESENDE DA FONSECA
Junho de 2022

POLITÉCNICO
DO PORTO

Instituto Superior de Engenharia do Porto

Chatbot para apoio à Divisão Académica do ISEP

João Miguel Resende da Fonseca

Dissertação para obtenção do Grau de Mestre em Engenharia Informática, Área de Especialização em Engenharia de Software

Orientador: Fátima Rodrigues

Porto, Junho de 2022

"With having failed, you must accept yourself as fallible in order to love others and life"

Thomas Vinterberg

Resumo

O Instituto Superior de Engenharia do Porto é uma Escola constituída por um número elevado de estudantes, cujo processo de suporte é fulcral para o bom funcionamento da sua comunidade. A disponibilidade do atendimento presencial da Divisão Académica é atualmente reduzida, dificultando o esclarecimento de dúvidas. Isto resulta não só no constrangimento dos processos virtuais, como *e-mail e* linhas telefónicas, como também na diminuição do nível de satisfação do estudante.

Para resolver este problema, foi idealizado um agente conversacional com o intuito de responder às perguntas mais frequentes relativas à Divisão Académica da Escola. O *chatbot* é constituído por um módulo de interpretação de linguagem, portuguesa, recorrendo a *machine learning*, particularmente técnicas de compreensão de texto. De forma complementar, para manter o sistema abstraído de qualquer especificação técnica, desenvolveu-se um serviço *web* que efetua a gestão da base de conhecimento do agente.

A avaliação da solução comparou o desempenho de vários modelos de classificação, compostos internamente pelo agrupamento de métodos de pré-processamentos de dados, modelos de linguagem (transformadores), e, finalmente, algoritmos de aprendizagem. Os resultados obtidos acentuam a superioridade do modelo de linguagem da Google, BERT, em junção com o classificador DIET, obtendo um *F1-Score* de 0.965.

O sistema foi implantado através de um protótipo exposto não só aos estudantes da Escola, como também aos membros da Divisão Académica. Para um total de 256 mensagens, obtiveram-se cerca de 30% de respostas não enquadradas. Em contrapartida, as avaliações persistidas foram positivas, com uma média acima de 4.20 e um desvio padrão de 1.23.

***Palavras-chave**: Chatbot, Universidade, Apoio ao Cliente, Natural Language Understanding, Deep Learning*

Abstract

Since Instituto Superior de Engenharia do Porto is a school with a high number of students, client support is very important to having a healthy community. This is handled by the academic department, who works under a very limited presential schedule. Virtual support, such as e-mails and phone lines, end up queuing and students are often dissatisfied with the duration of this process.

To solve the problem, a chatbot is idealized with the objective of answering the most frequent student questions. The system is conceptually composed of a portuguese language interpreter module based on machine learning, particularly natural language understanding. In order to have the application abstracted from the technical specifications of the chatbot framework, a web service was also developed with functionalities to maintain and modify the agent's information store.

The solution evaluation compared several intent classifying models, particularly text pre-processing methods, language models (transformers) and deep learning algorithms. The best combination was Google's language model, BERT, combined with DIET, obtaining a 0.965 F1-Score.

The chatbot was deployed through a public prototype exposed to both students from ISEP and to the student support department. The final result was about 30% of non fitted responses, while the remaining 70% were evaluated with a mean of 4.20 and a standard deviation of 1.23.

***Keywords**: Chatbot, College, Client Support, Natural Language Processing, Deep Learning*

Agradecimentos

Agradeço aos meus pais, Joaquim Fonseca e Graça Resende, por todo o apoio prestado ao longo da minha vida e pelo suporte constante em todas as minhas decisões.

Agradeço ao meu irmão, José Fonseca, pela orientação dada ao longo do meu percurso académico.

Agradeço a todos os meus amigos, particularmente os meus colegas de licenciatura e mestrado, José Ferreira, André Madureira, Gustavo Ferreira, Tomás Santos, Eduardo Carneiro e João Dias. Sem a vossa colaboração, motivação e competitividade, os anos curriculares teriam sido menos fruitivos.

Agradeço à minha namorada, Patrícia Ferreira, pela ajuda e pela motivação prestada no decorrer do último ano do Mestrado.

Agradeço à minha orientadora, Fátima Rodrigues, pelo acompanhamento regular no desenvolvimento da dissertação.

Por fim, agradeço ao Instituto Superior de Engenharia do Porto pela oportunidade de implementar o projeto.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Acrónimos e Símbolos

Lista de Acrónimos

ISEP	Instituto Superior de Engenharia do Porto
AIML	Artificial Intelligence Markup Language
XML	Extensible Markup Language
CLN	Compreensão de Linguagem Natural
PLN	Processamento de Linguagem Natural
ML	Machine Learning
DL	Deep Learning
ANN	Artificial Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
BOW	Bag of Words
LSA	Latent Semantic Analysis
NER	Named Entity Recognition
GRU	Gated Recurrent Unit
FFE	Fuzzy Front End
NCD	New Concept Development
SWOT	Strengths, Weaknesses, Opportunities and Threats
QFD	Quality Function Deployment
AHP	Analytic Hierarchy Process
HTTP	Hypertext Transfer Protocol
DTO	Data Transfer Object

1 Introdução

O presente capítulo descreve uma breve apresentação do projeto desenvolvido, expondo o seu contexto, o problema a resolver, os seus objetivos e os contributos que poderá ter. Denota-se ainda a metodologia seguida e a estrutura do documento.

1.1 Contexto

O apoio ao cliente é uma vertente crucial para qualquer negócio, seja uma organização com ou sem fins lucrativos, visto que serve como ponto de conexão com os seus clientes. Pode ser efetuado tanto a nível físico, presencialmente, como a nível virtual, através de linhas telefónicas ou da *internet*.

Aliado à expansão dos serviços virtuais, surgiram novos métodos de suporte ao utilizador final, nomeadamente a utilização de agentes inteligentes para o esclarecimento de dúvidas, os *chatbots*. Brevemente, estes pretendem simular um assistente humano e, através de inteligência artificial, compreender e responder adequadamente aos pedidos dos clientes. Usualmente, são capazes de estabelecer um diálogo fluído, mas também podem ser utilizados através de comunicação visual ou sonora.

A sua popularidade é notável, não só devido à transição do mercado presencial para o mercado virtual [1], mas também porque trazem vantagens em relação ao apoio tradicional ao cliente. O aumento de disponibilidade, o custo associado ao seu desenvolvimento e, ainda, a capacidade de criar uma relação única com múltiplos clientes são pontos de vantagem em relação ao comércio presencial [2].

1.2 Problema

Dado o elevado número de membros do Instituto Superior de Engenharia do Porto (ISEP), o processo de suporte ao estudante denota-se como um pilar no seu bom funcionamento. A

Escola já apresenta metodologias modernas no que toca a esta vertente, sendo que quase todos os processos podem ser realizados através da sua plataforma *online*, o portal. Contudo, apesar de trazer vantagens tanto para a Divisão Académica, como para os estudantes, também traz complicações, particularmente a dificuldade na compreensão e execução destes procedimentos virtuais.

A Divisão Académica funciona com base num horário reduzido, nomeadamente duas horas diárias para a linha telefónica e cinco horas para o atendimento presencial. Adicionalmente, com o aparecimento da pandemia, estes horários são cada vez mais limitados e necessitam de marcação prévia. Ainda que facilmente esclarecidos, a disponibilidade restringida do departamento resulta ou nos estudantes evitarem as suas questões, ou no constrangimento das filas de espera.

Ainda que expostos em documentação suporte no portal, assuntos como as matrículas, propinas e inscrições em exames são expostos repetitivamente à Divisão Académica. Isto traduz-se na ineficiência de processos que, aliados à pouca disponibilidade do departamento, acabam por penalizar estudantes que têm problemas que apenas possam ser esclarecidos com atendimento presencial.

A solução proposta baseia-se na implementação de um agente capaz de responder às perguntas mais frequentes recebidas pela Divisão Académica. Para além de simplificar o processo de esclarecimento de dúvidas, ao fornecer uma resposta imediata aos estudantes, a automatização deste processo retira carga de trabalho ao departamento.

1.3 Objetivos

O projeto desenvolvido tem como intuito a construção de um *chatbot* para a Divisão Académica do Instituto Superior de Engenharia do Porto, disponibilizando uma solução capaz de responder às perguntas dos estudantes de forma automática. Todo o desenvolvimento é aliado de processos de engenharia de requisitos, nomeadamente o levantamento de requisitos, a análise, o desenho da arquitetura do sistema e, por fim, a análise dos casos de uso.

O sistema é composto por dois componentes, nomeadamente a interface visual, a ser implantada no portal da divisão académica, e o servidor, o *chatbot*. Este funciona com recurso à inteligência artificial, utilizando processamento de linguagem natural e um modelo de classificação para compreender e gerar respostas de acordo com o pedido do utilizador.

O objetivo final é a implantação do agente conversacional num ponto de acesso público aos estudantes. No entanto, é necessário assegurar o seu bom funcionamento e a sua expansibilidade, sendo que a dissertação aborda vários objetivos:

- Análise e compreensão do conteúdo relevante para suporte ao estudante;
- Análise dos requisitos impostos pela Escola;
- Estudo de trabalhos relacionados e de plataformas de desenvolvimento de *chatbots*;

- Avaliação e determinação de qual o modelo e arquitetura mais adequada para o *chatbot*;
- Avaliação e determinação de quais as melhores métricas de análise de eficácia e eficiência do *chatbot*;
- Desenvolvimento e implementação do sistema conversacional através de algoritmos inteligentes;
- Construção de mecanismos de gestão de informação para os configuradores da Divisão Académica, abstraindo qualquer dependência técnica ou tecnológica;
- Implantação do *chatbot* no portal do ISEP e avaliação do seu sucesso;

Todos estes tópicos serão abrangidos no documento.

1.4 Contribuições

O sistema pretende facilitar a procura de informação relativa aos serviços académicos do ISEP, evitando não só o deslocamento físico, uma mais-valia na sociedade atual, mas também diminuir o número de atendimentos presenciais e chamadas telefónicas. O *chatbot* serve como ponto de entrada ao apoio ao estudante, sendo que retira a centralização do processo de assistência alocado à Divisão Académica.

Para além do melhoramento da comunicação entre o Instituto e os seus membros, o tempo que seria despendido no esclarecimento de dúvidas poderá ser utilizado para outras tarefas mais relevantes. Desta forma, aumenta-se a produtividade não só da Divisão Académica, como também dos estudantes.

Ainda que o *chatbot* tenha capacidade de resposta limitada aos tópicos definidos, estudos referem que interações baseadas em diálogo natural são marcantes [3]. A utilização de um sistema de respostas resulta numa experiência agradável e carrega uma melhor relação custo-benefício para a organização quando comparado com um serviço *web* tradicional [4].

Por fim, em termos científicos, o presente estudo é relevante para a construção de agentes conversacionais constituídos por técnicas de processamento de linguagem natural em conjunto com algoritmos de aprendizagem automática. Para além da exposição do seu processo de construção, avalia-se a influência que estas áreas têm relativamente à criação de um modelo de classificação de texto eficaz e eficiente.

1.5 Abordagem

A fase inicial da dissertação consiste no estudo de trabalhos relacionados, nomeadamente a análise de arquiteturas e algoritmos já estabelecidos de forma a enquadrar qual a melhor solução para os requisitos impostos. Com o intuito de complementar esta pesquisa, realiza-se

uma análise de valor para definir quais as funcionalidades que possam ser relevantes tanto para a organização como para o cliente.

Para que os estudantes possam ter respostas relevantes, é necessário compreender e recolher qual a informação a fornecer ao *chatbot*. Esta etapa, realizada antes do desenho da solução, decorre em parceria com a Divisão Académica através da análise de artefactos já existentes. Adicionalmente, para compreender se o agente é, ou não, útil, é necessário compreender quais as métricas de qualidade mais relevantes para um agente inteligente.

Posteriormente à análise e desenho, o desenvolvimento do *chatbot* realiza-se através de protótipos que, após obter aprovação pelo ISEP, serão implantados no portal. A sua taxa de eficácia é analisada consoante a avaliação direta dos estudantes.

1.6 Estrutura do Documento

O presente documento está dividido em seis capítulos, cada um relevante para a compreensão global do trabalho produzido. O capítulo atual apresenta uma breve introdução ao projeto, explorando o seu enquadramento, o problema que visa solucionar e, por fim, os seus contributos.

O Capítulo 0 expõe o estudo precoce ao desenvolvimento da aplicação, onde se elabora comparações do serviço a construir com outros projetos já expostos no mercado. Para além disso, introduz-se o conceito de *chatbot*, explorando a sua história, as arquiteturas normalmente utilizadas e respetivos padrões de desenvolvimento. Do mesmo modo, são apresentadas as tecnologias que são relevantes para a sua implementação.

O Capítulo 3 explica a introdução de inovação e a geração de valor proposta pelo projeto desenvolvido. Esta é suportada através da exposição de artefactos, como o modelo *Value Proposition*, *Quality Function Deployment* e ainda a seleção das ferramentas a utilizar de acordo com *Analytic Hierarchy Process*.

De seguida, o Capítulo 4 denota o estudo detalhado do processo de engenharia de requisitos, expondo, primeiramente, a análise do domínio numa perspetiva de negócio. Por fim, mas não menos importante, expõe-se toda a fase de desenho da solução desenvolvida.

O Capítulo 5 ilustra a fase de implementação e implantação da solução, especificando três passos, nomeadamente a construção do serviço de gestão de informação, a configuração dos processos de linguagem natural e, por fim, a integração entre ambos.

O Capítulo 6 evidencia a fase de apreciação posterior à implantação da solução, utilizando não só a avaliação direta do utilizador como as métricas de qualidade de modelos de classificação.

Por fim, no Capítulo 7 encontra-se a conclusão global da dissertação.

2 Estado da Arte

Neste capítulo demonstra-se a contextualização de conceitos teóricos relativos a agentes conversacionais, examinando a sua evolução e os tipos de *chatbots* existentes. Denota-se a sua importância no mercado, comparando a solução proposta com implementações já existentes. Por outro lado, ilustra-se o seu modelo de funcionamento, através de algoritmos e técnicas relacionadas com processamento de linguagem natural e a capacidade de reação destes agentes.

2.1 História e evolução de *chatbots*

Um *chatbot* classifica-se como um agente de *software* conversacional, permitindo ao emissor interagir com o sistema através de linguagem natural, projetada em texto ou voz [5]. De acordo com Joe Mayo [6], define-se como uma aplicação que interage com o utilizador através de um esquema de diálogo, normalmente disponível em plataformas de comunicação, utilizando alguma forma de inteligência.

O primeiro *chatbot*, denominado de ELIZA, foi desenvolvido em 1966 com o intuito de servir como um agente psicoterapeuta [7]. O seu funcionamento baseava-se num conjunto de pares pergunta-resposta definidos manualmente que, através de mecanismos de correspondência de palavras, geravam uma resposta. O seu maior problema era a adequabilidade de diálogo, dado que as respostas não continham qualquer contexto relativamente à conversa.

Embora mais soluções fossem propostas, criado em 1995, o *chatbot* ALICE [8] afirmou-se como um marco na investigação relacionada com a interação humano-computador, recebendo vários prémios. Inspirado por ELIZA, utiliza técnicas de correspondência de padrões com uma linguagem criada unicamente para o seu motor, a *Artificial Intelligence Markup Language* (AIML). Esta, derivada de XML, permite produzir a base de informação dos agentes baseada em blocos personalizáveis, dividindo o seu esqueleto por tópicos, que, por sua vez, são constituídos

por categorias. A categoria é a unidade atómica de informação, composta por uma questão e os modelos de resposta, como observado no extrato de Código 1.

```
1.  <category>
2.      <pattern>DO YOU KNOW WHO * IS</pattern>
3.      <template><star/> is my friend.</template>
4.  </category>
```

Código 1. Extrato de uma categoria AIML

O agente ALICE exponenciou o interesse na área do diálogo inteligente devido ao facto de, com esta arquitetura, ser possível criar um agente adequado a qualquer tema sem configurações adicionais à base de informação AIML. Por outro lado, o crescimento da popularidade das redes sociais e das plataformas de comunicação abriram portas à implantação destes *bots*: em 2001, SmarterChild, um *chatbot* inserido no MSN Messenger comunicava com os seus utilizadores, realizando tarefas como a partilha de notícias e previsões de temperatura, através de diálogo natural [9].

Recentemente, assistentes virtuais como a Siri e a Alexa estabeleceram-se como propulsionadores na evolução de interfaces conversacionais [5]. Desenhados com o objetivo de serem assistentes pessoais, são compostos por um diálogo orientado a tarefas, concretizando os pedidos requisitados pelo utilizador através de reconhecimento de voz. A sua reação é construída com a recolha de informação de motores de pesquisa, como o Google, gerando respostas através dos seus resultados.

Ainda que a maioria dos projetos sejam fundamentados em modelos semelhantes a ALICE e ELIZA, *frameworks* como a IBM Watson utilizam técnicas de inteligência artificial para substituir o processo de configuração de dados. Recorrem a mecanismos de compreensão de linguagem natural para automatizar a etapa de correspondência entre as intenções do utilizador e a resposta, suportadas com bases de informação extraídas da *internet* [10].

2.1.1 Influência no mercado

Os *chatbots* são cada vez mais populares entre organizações, sendo que, de acordo com Gartner [11], a sua comercialização aumentará exponencialmente nos próximos anos. No seu estudo, realizou a previsão que, até 2020, 85% da interação realizada entre um cliente e um negócio seria efetuada sem qualquer intervenção humana. De forma semelhante, um inquérito realizado pela Oracle em 2016 [12] referiu que 80% das organizações possuem, ou planeiam desenvolver um agente conversacional.

Atualmente, já existem inúmeros casos de uso para *chatbots*. Na indústria jornalística, são empregues como meio de transmissão de notícias, ou até de resultados de desporto. No mercado turístico, servem como ponto de marcação de voos, alojamentos, ou até como intermediadores de pagamento. No entanto, a sua maior frequência de utilização enquadra-se no contexto de suporte ao cliente [13].

De uma forma geral, os agentes pretendem substituir a assistência humana em situações onde um serviço rápido é solicitado, como perguntas frequentes ou a execução de processos simples. Por estas razões, são maioritariamente utilizados como pontos informativos [13]. A migração da atividade de suporte para a dimensão virtual não só traz vantagens para o utilizador final, como para a organização responsável pela sua manutenção. Ainda que possam ter serviço de apoio telefónico ou correio eletrónico, estes canais exibem maiores constrangimentos.

Recentemente, um ensaio realizado pelo Facebook [14] denotou que 61% dos seus utilizadores preferem comunicar com entidades comerciais através de mensagens que outro meio. Comparado com o apoio ao cliente tradicional:

- 59% dos inquiridos referem uma maior rapidez de resposta;
- 50% dos inquiridos referem um maior sentimento de preocupação pela organização;
- 49% dos inquiridos referem uma maior veracidade de resposta;
- **66% dos inquiridos** referem que construíram uma maior relação de confiança com a organização.

Para além da satisfação do cliente, o uso de agentes inteligentes também traz benefícios à organização. Respetivamente, a habilidade de resolução de problemas varia consoante a experiência de cada funcionário, que, para além de necessitar de contextualizar o conflito, ainda pode precisar de utilizar sistemas de suporte. Com a ajuda de um *chatbot* devidamente integrado, estes passos são facilmente resolvidos de forma automática. Para além disso, quanto mais contacto tiver com o cliente final, melhor será a sua capacidade de resolução [15].

No contexto académico, a comunicação entre as universidades e os seus membros acontece, por norma, através de canais virtuais, como o portal ou o correio eletrónico. Ainda que seja uma ocorrência regular, a carga de apoio ao estudante é sazonal, sendo que existem fases onde as suas necessidades de esclarecer dúvidas é maior. O início do ano letivo é um exemplo, com temas como o processo de inscrição, critérios de ingresso, horários estabelecidos, entre outros que preenchem os departamentos de apoio. Tal como a entrada anual dos estudantes, o seu conjunto de perguntas é cíclico, rotulando este cenário como ideal para o uso de um agente inteligente.

2.2 Classificação de *chatbots*

Como referido, os agentes inteligentes podem ser utilizados em diversos contextos, sendo que são construídos de forma díspar dependendo do seu objetivo. Concretamente, um *chatbot* é classificado consoante o seu domínio de conhecimento, o nível emocional, os objetivos finais e o processamento de *input* e *output*.

O **domínio de conhecimento** pondera duas opções, particularmente o domínio aberto, onde existe a capacidade de comunicar adequadamente sobre qualquer tópico introduzido, e o domínio fechado, que se foca num tema específico e não é capaz de interagir com assuntos diferentes [16].

O **nível emocional** do agente considera o índice de proximidade afetiva com o cliente final, sendo que, enquanto os agentes interpessoais não pretendem construir qualquer intimidade, os agentes intrapessoais são definidos como companheiros particulares que têm como objetivo final entender e reagir ao sentimento do utilizador [16].

O **objetivo final** avalia os *chatbots* tendo em conta o seu intuito principal. Caso sirva apenas como um meio de comunicação originário de uma base de informação fixa, classifica-se como informativo, sendo que, dentro deste espectro, existem ainda os *task-based bots*, caso a interação seja realizada através de um processo sequencial de tarefas. Adicionalmente, os agentes conversacionais são definidos como sistemas que apenas pretendem ter uma conversa fluída com o utilizador sem qualquer restrição de informação [17].

Por fim, de acordo com o **processamento de *input* e a respetiva lógica de geração de respostas**, os *chatbots* segmentam-se em três modelos, particularmente o modelo baseado em regras, o modelo baseado em busca e, por fim, o modelo generativo.

O modelo baseado em regras baseia-se na ligação entre a frase recebida e um conjunto de informação fixo. Dado que a base de conhecimento é introduzida manualmente, quanto mais conceitos existirem, maior será a sua abrangência. O problema mais comum neste tipo de agentes é a falha na reação a erros gramaticais, visto que a fase de correspondência pode não encontrar qualquer par correspondente se houver erros lexicais (Figura 1). Para além disso, como o comportamento do agente é definido através de regras, requerem um nível de manutenção elevado.

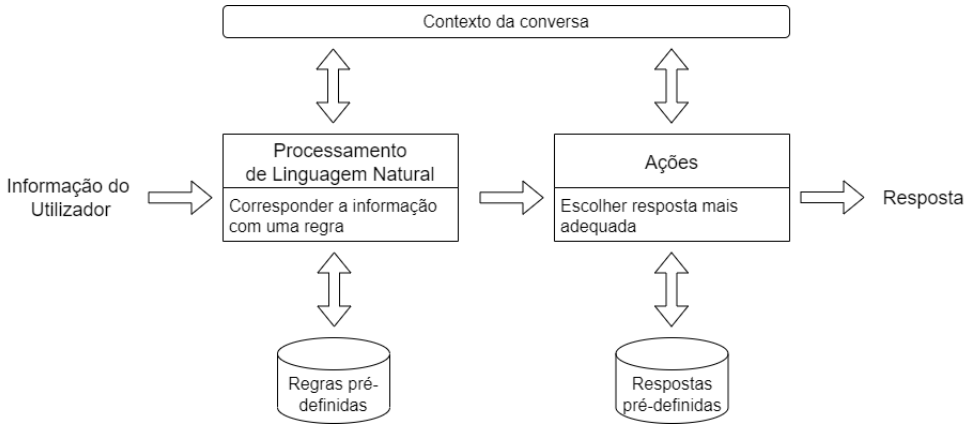

Figura 1. Visualização de alto nível do modelo baseado em regras

O modelo baseado em busca é uma ramificação do modelo demonstrado acima, contudo, em vez de usar uma base de informação fixa, utiliza mecanismos de pesquisa complementar, como a invocação de serviços *web* externos ou motores de busca.

Por último, o modelo generativo é moldado por técnicas de aprendizagem automática [18]. Para ter capacidade de resposta, o modelo necessita de passar por fases de treino onde, com um conjunto de dados inicial, aprende a comunicar. Comummente, estes agentes costumam ser treinados com conversas e discussões em fóruns virtuais. De forma igual ao modelo baseado

em regras, quanto maior o conjunto de informação fornecido ao agente, mais fluida será a sua capacidade de conversa.

As características de um agente influenciam a sua complexidade, sendo que quanto melhor a sua capacidade de reação, maior será a dificuldade de desenvolvimento. O *chatbot* mais simples classifica-se como domínio fechado, interpessoal, informativo com um modelo baseado em regras, com um funcionamento cingido à informação importada pelos desenvolvedores. Como observado pela Figura 2, a complexidade de construção de um agente aumenta de forma proporcional ao seu grau de reatividade.

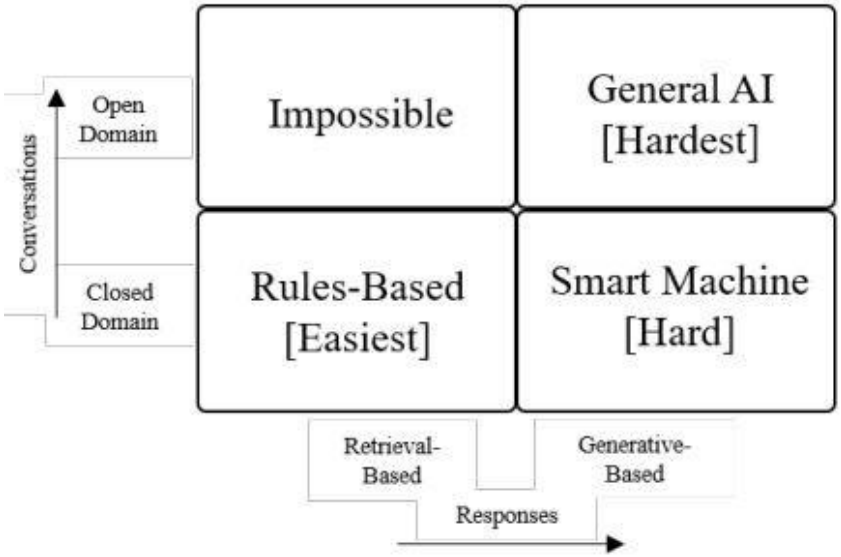

Figura 2. Comparação de complexidade dos vários tipos de *chatbots* [10]

A solução proposta é um sistema com interação do tipo pergunta-resposta, constituindo uma versão simplificada de um agente conversacional, onde o objetivo não é construir uma conversa, mas responder diretamente aos pedidos do utilizador. Tendo em conta um ponto de entrada de linguagem natural, a sua função é a procura de informação numa base de conhecimento. Ainda que possam ser utilizados para qualquer tema, a sua incidência é mais popular em agentes de domínio fechado, visto que, desta forma, a base de conhecimento é reduzida e limitada aos temas de negócio [19]. Este é o caso da dissertação.

2.3 Modelo de funcionamento de *chatbots*

Um *chatbot* é dividido em três componentes principais, nomeadamente o processamento da linguagem de entrada, o gestor de diálogo e, por fim, o gerador de resposta. Opcionalmente, pode conter um módulo de transformação entre voz e texto, que sai fora do âmbito deste projeto.

Na primeira fase, o módulo de processamento da linguagem de entrada tem como objetivo interpretar a mensagem recebida e convertê-la numa estrutura lexical pré-definida. Ao contrário de uma linguagem de programação, onde cada conceito está associado a um único propósito, a linguagem humana não é linear, sendo que uma palavra não tem sempre o mesmo significado [20]. Desta forma, é necessário compreender o seu significado em concordância com o resto da frase para que, futuramente, não existam conotações erradas entre as expressões do utilizador e o seu objetivo.

Assim que a mensagem seja interpretada, o gestor de diálogo é responsável por reagir ao seu intento, comparando-o com uma base de conhecimento populada com informação do domínio de negócio. Em consonância com a ação determinada, a informação de saída é construída através de modelos generativos ou através da utilização de respostas tipo. Em caso de insucesso, o *chatbot* pode decidir responder com um pedido de mais informação ou, em último recurso, com uma mensagem de erro.

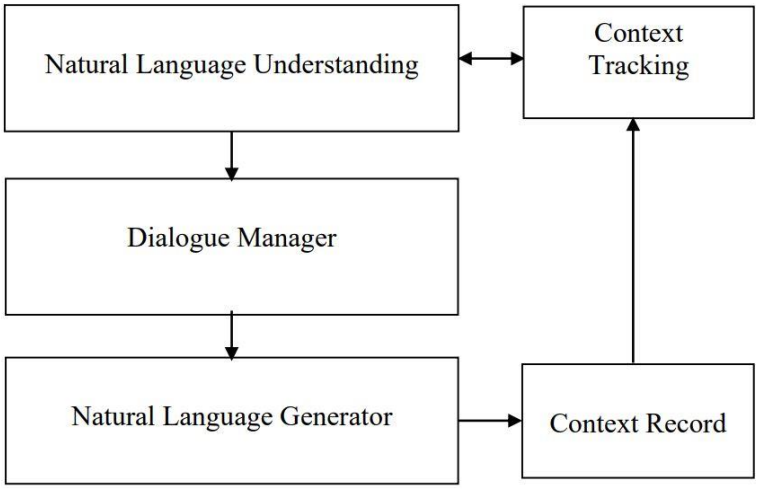

Figura 3. Arquitetura de alto nível de um *chatbot* [21]

A Figura 3 demonstra a esquematização e a integração dos componentes internos a um agente conversacional. Cada módulo é analisado detalhadamente nas seguintes secções.

2.3.1 Processamento de linguagem natural

O conteúdo transmitido pelo emissor é suportado através de texto, composto por particularidades importantes na compreensão de sentimento. Respetivamente, o discurso pode ser produzido com irregularidades como erros gramaticais ou autocorreções, sendo que o agente necessita de estruturar a informação num contexto semântico e gramatical [20].

Esta fase é denominada de compreensão de linguagem natural, ou CLN, onde o sistema pretende replicar o raciocínio do ser humano aquando da interpretação de uma frase. É um processo interno à área de investigação de processamento de linguagem natural, ou PLN.

De acordo com Elizabeth Diddy [22], PLN refere-se à investigação motivada por um conjunto de técnicas computacionais para analisar e representar texto de ocorrência natural a um nível linguístico, com o intuito de reproduzir comunicação semelhante a um ser humano. O maior problema que a área pretende resolver é a ambiguidade presente neste tipo de linguagem [23]:

- Ambiguidade lexical – múltiplos significados para a mesma palavra;
- Ambiguidade de sintaxe – significados diferentes para frases de estrutura igual;
- Ambiguidade pragmática – mudança do sentido da frase consoante a interpretação do recetor;
- Ambiguidade referencial – incapacidade de perceção da entidade a ser referida.

Apesar de variar com o tipo de idioma, estas ambiguidades estão frequentemente presentes no discurso português, como pode ser observado pela Tabela 1.

Tipo de Ambiguidade	**Exemplos**
Lexical	Estou no **canto** da sala. *(canto é a extremidade de um espaço)*
	Eu **canto** uma música. *(canto é derivado do ato de cantor)*
Sintaxe	Vi um homem com **binóculos.** *(o homem observado estava com binóculos)*
	Vi um homem com **binóculos.** *(sujeito utilizou binóculos para ver o homem)*
Pragmática	**Podes** levantar a rocha? *(pedido dirigido a uma entidade)*
	Podes levantar a rocha? *(questão de capacidade)*
Referencial	O João está com o Luís. **Ele** está doente. *(João está doente)*
	O João está com o Luís. **Ele** está doente. *(Luís está doente)*

Tabela 1. Exemplos de tipos de ambiguidade presentes na linguagem portuguesa

Ao contrário de PLN, área que se foca no processamento de texto num sentido literal, CLN é a vertente que tem como objetivo extrair o contexto e o significado da frase como um todo. Desta forma, os sistemas de interpretação de texto dividem-se em dois, nomeadamente os sistemas de correspondência de padrões, determinísticos, e os sistemas de aprendizagem automática, não determinísticos. [24].

2.3.1.1 Classificação de discurso

Para sistemas conversacionais, as técnicas de PLN são utilizadas para conseguir estruturar as várias dimensões do discurso. Atualmente, a maioria dos *chatbots* classifica o texto em entidades e intenções [25], no entanto, também pode existir a necessidade de classificar qual o ato de diálogo.

A **intenção** refere-se ao objetivo ou ao significado do discurso do utilizador. É a classificação que o componente de entrada atribui à frase que, posteriormente, é utilizada para executar

uma ação ou gerar uma resposta final. As intenções são os pilares que determinam a capacidade de resposta do *chatbot,* sendo que, normalmente, são introduzidos manualmente pelos desenvolvedores.

Uma **entidade** é considerada como um influenciador do contexto final da frase. São palavras-chave que estão sempre associadas ao mesmo valor, como por exemplo nomes pessoais, localizações, etc. No caso da Figura 4, as entidades ajudam a compreensão do objetivo final da intenção, nomeadamente a localização original e o destino de um voo.

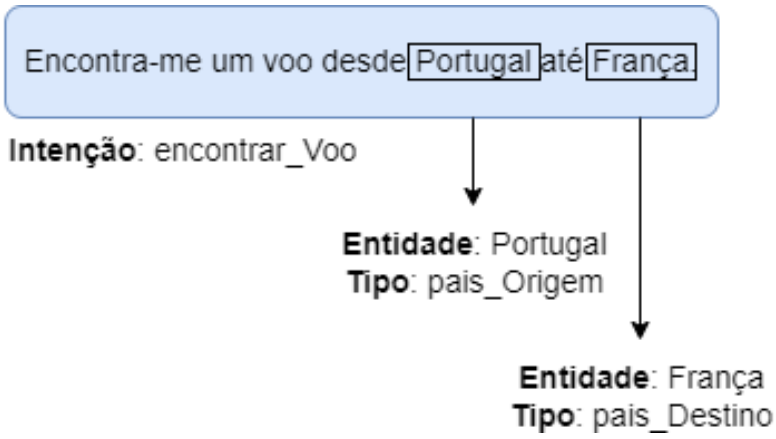

Figura 4. Exemplo da classificação de uma frase em termos de entidades e intenção para o agendamento de um voo

O **ato de diálogo** denota o tipo de função da frase, tendo em conta critérios de identificação linguística. O processo de *dialogue act recognition* [20] utiliza modelos pré-definidos para analisar expressões que indicam o objetivo final do discurso, como por exemplo o facto de *por favor* remeter a um pedido, ou uma questão terminar sempre com um ponto de interrogação (Tabela 2).

Emissor	**Expressão**	**Função da frase**
1	Olá.	Abertura de conversa
2	Olá, amiga.	Abertura de conversa
1	Está tudo bem?	Questão
2	Estou bem.	Afirmação
1	Adeus!	Conversa terminada

Tabela 2. Classificação da função da frase no discurso entre dois emissores

Cada agente conversacional pode ser constituído pela sua estrutura de diálogo, desde que as suas regras de identificação sejam especificadas. No entanto, já existem modelos frequentemente associados a *chatbots*, nomeadamente o *DAMSL* [26]. Visto que o agente desenvolvido apenas terá interações de perguntas e respostas, o ato de diálogo não é relevante para o seu desenvolvimento.

O ponto principal deste módulo é a descoberta da intenção. Normalmente, as técnicas de PLN são utilizadas na fase de preparação de algoritmos de análise textual, que, tendo em conta documentos de treino, constroem modelos de classificação automática [20]. Os agentes mais recentes utilizam técnicas de conversão de frases em valores numéricos, como *bag of words*, *latent semantic analysis*, e, *word embedding,* explorados em detalhe na secção 2.4.3.2, para que os algoritmos de aprendizagem consigam compreender qual a sua relação semântica e estabelecer formas de previsão de respostas.

2.3.2 Gestor de diálogo

O módulo de gestão de diálogo é responsável pelo processo de orientação da conversa, escolhendo qual a tarefa a executar após cada interação. As suas ações mais comuns são a tentativa de guia para temas em que o agente tenha informação, ou até o pedido de confirmações ou clarificações [27].

O critério principal do gestor de diálogo é a estratégia de interação, que define quem é que lidera a conversa: o utilizador ou o sistema [20]. Caso a origem seja o ser humano, o sistema baseia-se numa comunicação do estilo cliente servidor, onde o agente apenas retorna respostas aos seus pedidos. O maior problema deste tipo de interação é o facto de o agente ter a possibilidade de não conseguir interpretar a intenção emitida. Apesar de menos frequente, a segunda vertente incide na orientação feita pelo sistema, resultando em menos erros de diálogo.

2.3.2.1 Estado do Diálogo

O rastreio do estado de diálogo é efetuado para estimar o contexto da conversa entre o sistema e o utilizador, tendo em conta não só a informação recebida, como todas as rondas de interação anteriores. Williams [28] define o estado como uma estrutura de dados que sumariza toda a história da conversa a um nível que permita ao sistema definir qual o próximo passo a tomar. O rastreio é suportado por todos os elementos armazenados até ao instante, particularmente os resultados do componente de PLN, as ações do sistema e, por fim, a base de informação exterior.

Um método prominente de alcançar o controlo do contexto são os algoritmos baseados em estado – *finite state machine* [29]. Estes são constituídos por um grafo de diálogo que, dependendo da informação de entrada e de um conjunto de regras definido, transitam de fluxo em fluxo. Uma derivação desta metodologia é o *slot-fill Model* [29], algoritmo baseado em blocos que representam o estado do diálogo através de critérios de informação (*slots*) que são preenchidos através das respostas do utilizador (Figura 5).

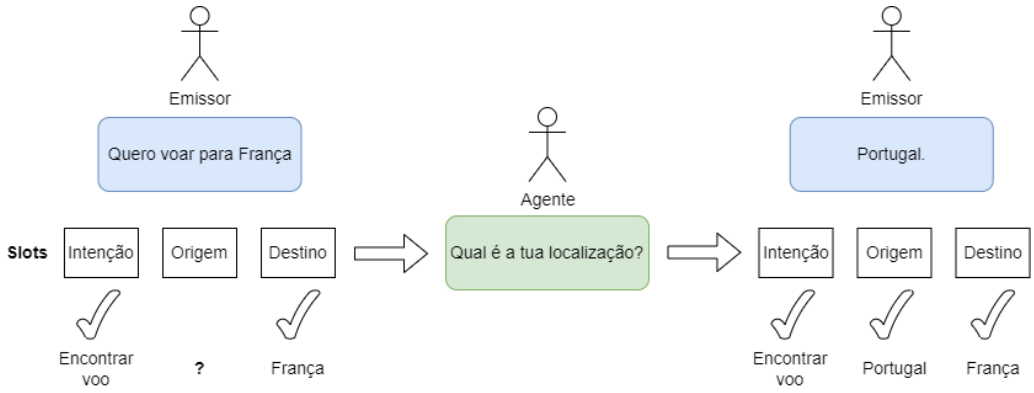

Figura 5. Importância de um gestor de diálogo na descoberta de intenção, utilizando um algoritmo baseado em *slots*

Com o aumento de capacidade de processamento, modelos generativos ganharam preferência relativamente a métodos baseados em estados [30]. Ainda que num ponto inicial os procedimentos de aprendizagem estejam em desvantagem, a utilização dos agentes inteligentes e o *feedback* do utilizador melhoram o modelo ao longo do tempo. Por outro lado, acabam com a dependência da definição manual de regras de decisão e resolvem o problema de cobertura insuficiente.

2.3.3 Geração de Resposta

O módulo de geração de resposta tem como objetivo receber uma representação da intenção interpretada e convertê-la em linguagem natural [20]. Fruto das fases anteriores, informação como a história do diálogo, o contexto e o objetivo do utilizador é utilizada para a construção da resposta. De forma complementar, pode ainda utilizar fontes de dados exteriores, particularmente a sua base de informação, mecanismos de pesquisa ou algoritmos generativos. Os modelos generativos usualmente são mais adequados para agentes que pretendem alcançar uma conversa mais fluída, enquanto os modelos baseados em pesquisa encontram a sua utilidade em diálogo orientado a tarefas [24].

2.3.3.1 Modelo baseado em pesquisa

O modelo baseado em pesquisa depende de uma base de dados externa pré-populada com respostas que, através da extração da intenção do utilizador, define qual a mais provável de ser adequada. A representação estruturada de texto, nestes modelos, é resultado das técnicas de PLN, referidas na secção 2.4.3.

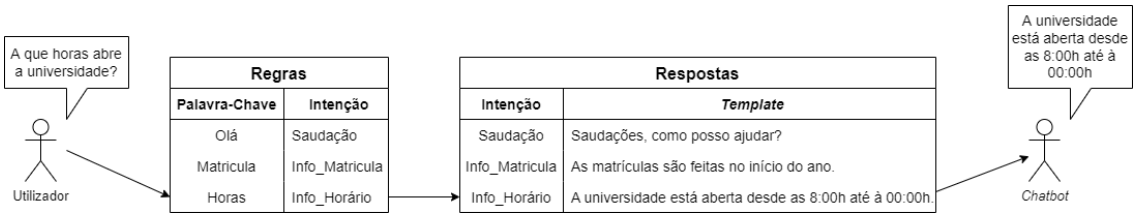

Figura 6. Funcionamento típico de *chatbots* baseados em regras

A base de informação pode ser pré-definida através de respostas candidatas, tratando-se de um **modelo baseado em regras** (Figura 6). Ainda que o exemplo referido contenha apenas um exemplo por intenção, existem métodos que contêm conjuntos que são individualizados aleatoriamente.

Por outro lado, é possível fazer a seleção de respostas através de modelos de classificação. Ao contrário da correspondência direta de palavras-chave, a avaliação da resposta é realizada através da sua comparação com corpos de texto externos. Isto é estabelecido através de sistemas de pontuação e o cálculo da proximidade de significado. A metodologia mais simples é a contagem de palavras comuns entre ambos, sendo que a resposta candidata será a que tiver o maior valor [31]. Alternativamente, redes neuronais são utilizadas para o processo de correspondência, tomando como *input* uma frase e um candidato de resposta, estimando a probabilidade deste ser adequado tendo em conta a similaridade entre cada palavra [20].

2.3.3.2 Modelo generativo

Os modelos generativos pretendem substituir a escolha de respostas pré-definidas com a geração dinâmica de texto. Ainda que continuem a carecer de possíveis respostas a escolher, o método não requer a instrução de todos os cenários possíveis.

A geração de *output* é feita no momento de processamento através de um modelo treinado: para um pedido P, com N respostas candidatas possíveis, o modelo gera N respostas embebendo o contexto da conversa [32]. Desta forma, para além de conseguir evitar a repetição de respostas candidatas, ainda possibilita a introdução de informação relevante ao histórico do diálogo.

Um modelo típico generativo é o *Seq2Seq model* [33] que, através de redes neuronais recorrentes, captura as semânticas do *input* através de vetores (*encoder*) e depois realiza a tradução de cada valor para uma resposta (*decoder*).

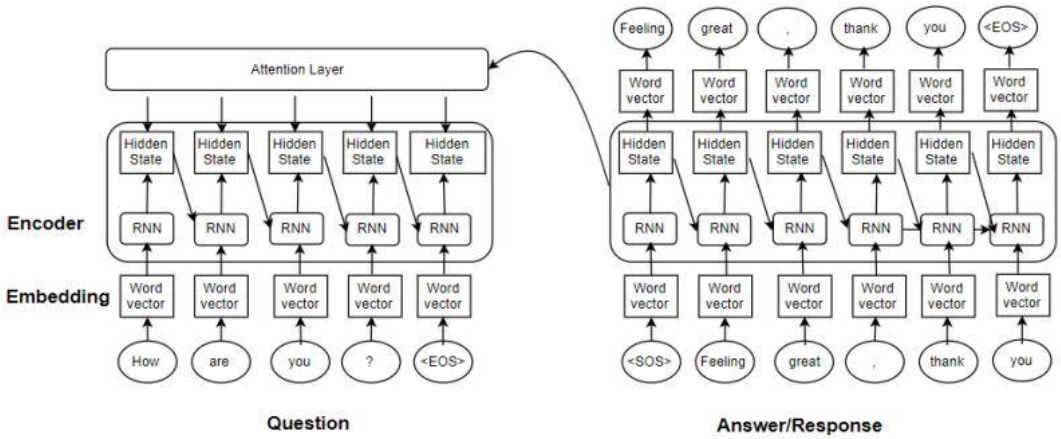

Figura 7. Arquitetura de um agente *Seq2Seq* [33]

A maior diferença entre este tipo de geração de resposta e o modelo de pesquisa é o facto de não depender de um repositório pré-definido, fornecendo mais flexibilidade no ponto de saída de informação.

2.4 Machine Learning

Machine learning (ML) é um pilar para a construção de agentes conversacionais, particularmente as áreas de aprendizagem profunda e de compreensão de texto. A presente secção pretende explorar a sua utilidade, em conjunto com os detalhes de implementação.

Para isto, introduz-se a definição de ML, os tipos de aprendizagem existentes e a respetiva contextualização com o esquema de funcionamento de *chatbots*. Por fim, e mais importante, refere-se a sinergia entre todas estas áreas aquando da construção de um modelo de classificação capaz de associar um segmento de discurso a uma ação.

ML constitui um subconjunto de inteligência artificial que pretende estudar e compor algoritmos computacionais que simulam a inteligência humana ao aprender com o seu ambiente de envolvimento [34]. De acordo com Tom Mitchell [35], definem-se como programas que aprendem com a experiência *E* relativa a uma tarefa *T* caso a sua performance em *T* seja positiva. Recentemente, tornou-se numa das áreas mais importantes e prolíficas de inteligência artificial, sendo utilizada em vários setores da indústria, como na construção de agentes conversacionais [36].

Os sistemas tradicionais são produzidos com um conjunto definido de funções, normalmente conformados a uma área de domínio. Com a popularização de inteligência artificial, muitos programas foram influenciados a migrar o seu comportamento, convertendo o núcleo não adaptável para um sistema reativo ao seu *input*. Uma representação genérica desta transformação pode ser definida através da Figura 8.

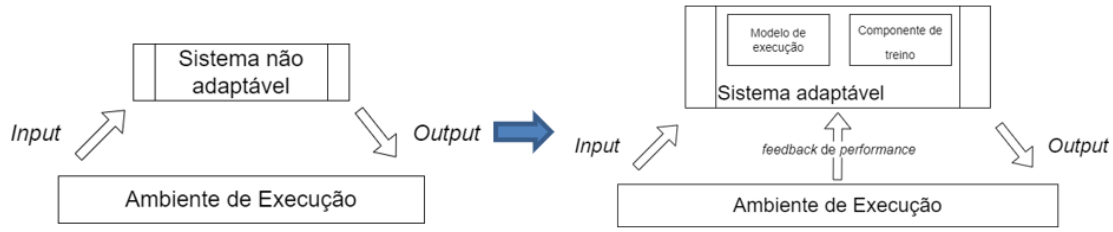

Figura 8. Arquitetura de um sistema tradicional em comparação com um sistema adaptável

Os sistemas reativos não são compostos por procedimentos estáticos de lógica, mas sim por componentes que possuem a habilidade de adaptar o seu comportamento tendo em consideração os dados fornecidos e o *feedback* do ambiente de execução.

2.4.1 Modos de aprendizagem

O funcionamento de um algoritmo de ML baseia-se em dois conceitos chave, nomeadamente a ação que está a ser aprendida, o rótulo (*label*), e as respetivas características que definem as justificações da rotulação (*features*), compostas por um ou mais atributos. As metodologias dividem-se em quatro tipos, dependendo do seu tipo de aprendizagem, nomeadamente a aprendizagem supervisionada, não-supervisionada e a aprendizagem por reforço.

A **aprendizagem supervisionada** divide-se em duas fases, o treino e o teste**.** Utiliza um conjunto de dados de entrada *x* já rotulados e um modelo f como motor para a previsão de novas entradas *y*: $y = f(x)$. O objetivo destes métodos supervisionados é compilar a função *f* através de um conjunto de informação corretamente classificado, gerando o modelo de previsão.

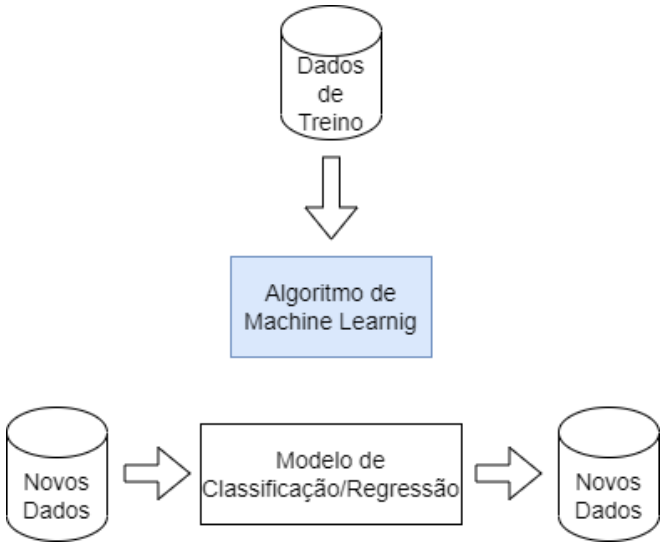

Figura 9. Enquadramento do algoritmo de *machine learning* para uma aprendizagem supervisionada

A finalidade deste modo de aprendizagem divide-se em dois casos, particularmente os problemas de classificação e de regressão. O primeiro atribui os dados de teste a uma categoria específica enquanto o último pretende compreender e gerar uma função de relação entre as variáveis interdependentes. Técnicas normalmente utilizadas para estes problemas *são Support Vector Machines, Decision Trees, K-Nearest Neighbor*, entre outras [37].

Em contrapartida, a **aprendizagem não supervisionada** tem como objetivo a descoberta autónoma de padrões e funcionalidades através de um conjunto de dados sem rotulação.

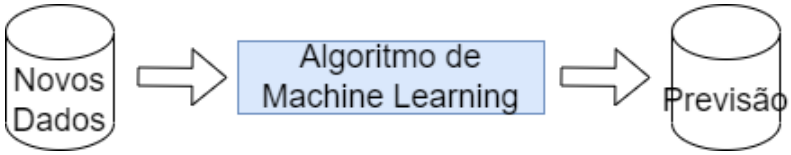

Figura 10. Enquadramento do algoritmo de *machine learning* para uma aprendizagem não supervisionada

A **aprendizagem por reforço** é constituída por agentes operacionais que pretendem identificar o melhor procedimento para alcançar um prémio**,** ou melhorar a prestação até esse momento**.** A metodologia é baseada em iterações classificadas de forma binária, positiva ou negativa, até conseguir obter um nível que atinja o objetivo final. A classificação é cumulativa, disponibilizando o agente com avaliações do que deve, ou não, realizar após cada fase de treino.

A sua aprendizagem divide-se em três conceitos chave: a *política de aprendizagem*, a função que define o comportamento do agente num instante de tempo, o *sinal de prémio*, a função que define qual a meta da aprendizagem e, por fim, a função de valor, o modelo que classifica se o comportamento do agente é positivo para futuras iterações [38].

Figura 11. Esquema de funcionamento para o processo de aprendizagem por reforço

Os agentes conversacionais informativos recorrem ou à aprendizagem supervisionada ou não supervisionada. Para o primeiro caso, o processo de treino é efetuado através de um conjunto de tópicos anotados com a intenção correta, enquanto a aprendizagem não supervisionada recorre a uma base de informação maior para deduzir independentemente quais as intenções do utilizador. Para um sistema de perguntas e respostas, a aprendizagem supervisionada é mais relevante dado que é possível utilizar exemplos de perguntas e respostas corretas para alcançar o melhor desempenho possível.

2.4.2 Aprendizagem Profunda

Aprendizagem profunda, ou *deep learning*, é uma subárea de ML que modela abstrações de dados com altos níveis de abstrações utilizando redes neuronais com camadas de processamento suportadas por transformações lineares e não lineares [39].

As metodologias convencionais de ML são limitadas pela capacidade de processar dados sem qualquer pré-processamento ou rotulação. Por oposição, a aprendizagem profunda estabelece métodos que, para além da capacidade de previsão, pretendem aprender o procedimento de classificação de forma automática. Internamente, os modelos ampliam a compreensão de informação entre cada nível de processamento, utilizando transformações de dados.

Artificial Neural Networks (ANN) são sistemas computacionais que pretendem emular o funcionamento biológico do cérebro humano. Podem ser considerados como grafos constituídos por vários níveis, internamente compostos com nós, conectados através de pesos matemáticos. As redes neuronais são constituídas por três camadas, nomeadamente a camada de *input*, a camada de *output* e a camada escondida, sendo que esta pode ter um número variado de níveis internos.

Os nós pertencentes a estes sistemas, denominados de neurónios, estão associados a um valor numérico, a ativação, que é resultado de uma função que recebe todos os pesos de neurónios imediatamente anteriores - $peso \times ativação$ - e modifica o *output* para as camadas seguintes.

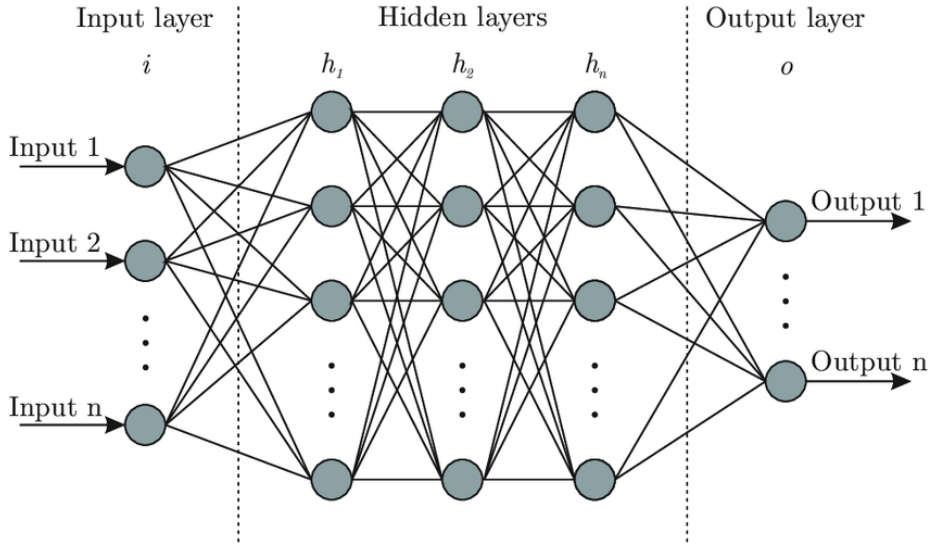

Figura 12. Núcleo de uma rede neuronal [40]

Os conceitos principais de uma rede neuronal são a célula e o respetivo estado de ativação, as conexões entre unidades, *a função de ativação* que determina qual o *output* de uma célula, a função de custo, e, por fim, o método de recolha de informação (Figura 13).

O papel da função de custo é, iterativamente, calcular a diferença entre o resultado obtido e o resultado esperado, indicando, através de um valor numérico, se os pesos de cada conexão devem ser aumentados, se positivos, ou diminuídos, se negativos. Tendo em conta estes valores, a reconfiguração da rede neuronal é efetuada através do algoritmo de *backwards propagation.*

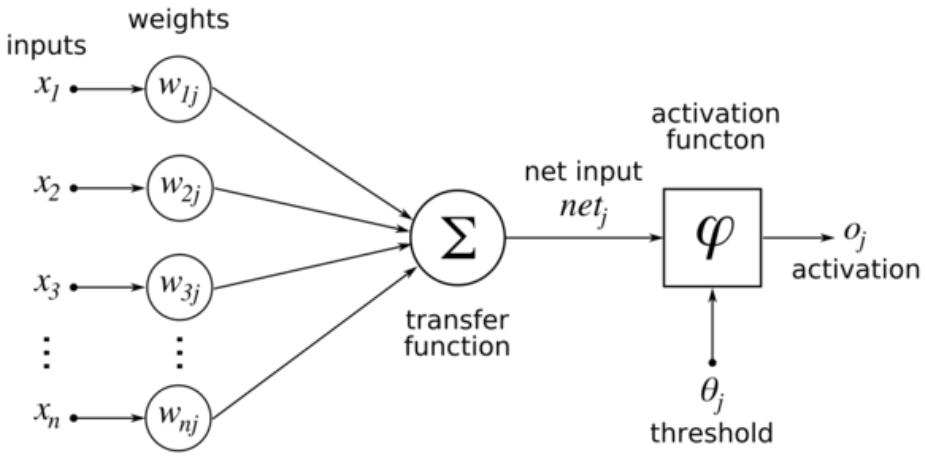

Figura 13. Funcionamento interno de um neurónio [41]

Dependendo da orientação de execução, existem dois tipos de redes neuronais, particularmente as *feed-forward networks,* que funcionam num único sentido desde a entrada

à saída, e as *recurrent networks,* que implicam conexões entre os seus nós. As *recurrent neural networks* (RNN) são caracterizadas pelo facto de os neurónios incluírem ligações cíclicas para os anteriores, sendo que desenvolvem uma noção de contexto com etapas diretamente antecedentes.

Long Short-Term Memory (LSTM) representa um modelo de rede neuronal recorrente que é frequentemente utilizada em PLN. Para combater os problemas de falta de contexto para sequências de dados, sendo que as redes recorrentes tradicionais apenas contêm conexão com a célula anterior, o nó de processamento é ampliado para armazenar informação passada – *hidden state* [42].

As células de memória são os pontos de consciência que decidem quais os dados a preservar, utilizando *gates*, particularmente o *input gate,* responsável por decidir se a nova informação deve ser guardada, o *output gate,* com o cargo de partilhar ou não a memória com a rede neuronal e, por fim, o *forget gate,* delegado para apagar a memória presente.

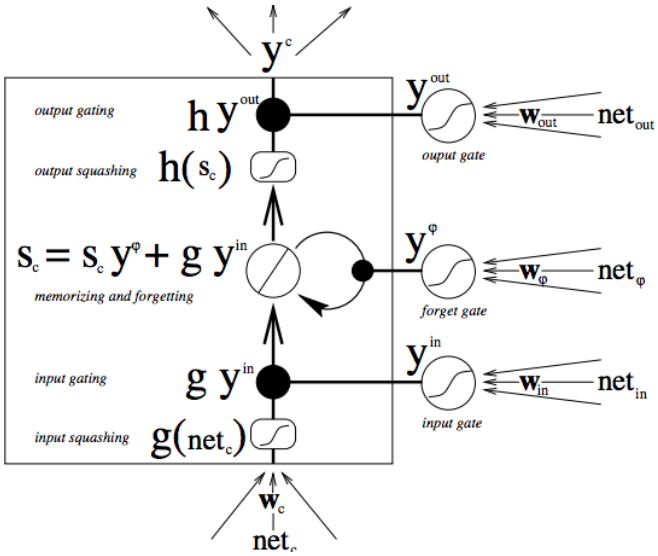

Figura 14. Comportamento interno das unidades de LSTM [43]

Como observado pela Figura 14, a informação da rede neuronal é fornecida para os três pontos constituintes da unidade LSTM, sendo que cada terá a sua influência no *output* a enviar.

Transformers é uma arquitetura de redes neuronais descoberta recentemente, em 2017, pela equipa da Google [44]. Tem como objetivo substituir o uso das LSTM, que eram consideradas o estado da arte de CLN, devido ao seu maior problema, o processamento sequencial de palavras. Embora obtivesse métricas positivas, a complexidade de treino e, consequentemente, a sua duração, aumentavam exponencialmente com um número elevado de corpos de texto. A sua inovação consiste em dois novos mecanismos, nomeadamente a codificação de posição, ou *positional encoding*, e a introdução do conceito de atenção, ou *self-attention*. Em conjunto, permitem analisar cada frase em ambos os sentidos, contextualizando o significado dos seus elementos independentemente da ordem.

O objetivo inicial dos transformadores era servir como um tradutor, dai o foco em conseguir compreender não só o contexto da frase, como a sua ordem [44]. A sua arquitetura divide-se em dois componentes, o codificador, que recebe a mensagem, e o descodificador, que faz a tradução, como exposto na Figura 15.

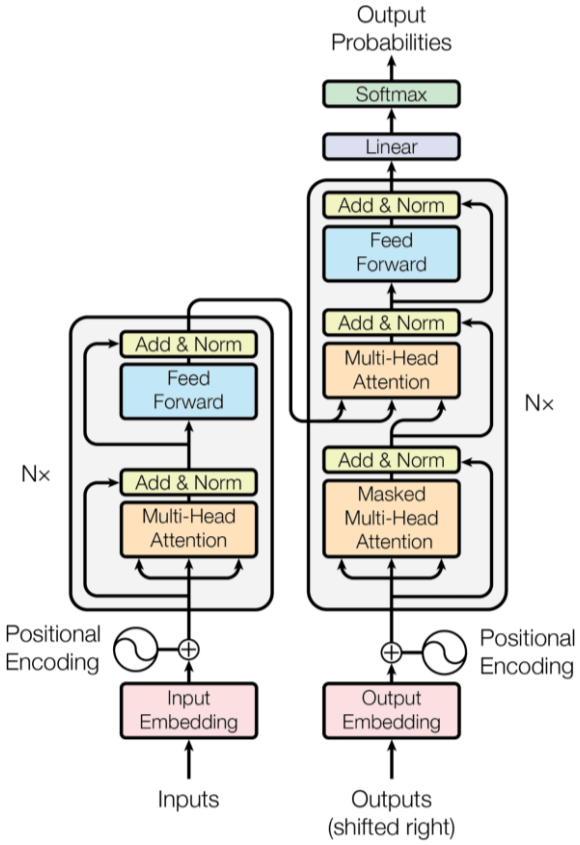

Figura 15. Arquitetura geral de um transformador [44]

O ponto de entrada são os *tokens* de uma frase, imediatamente convertidos em *input embeddings*. Tendo em conta um dicionário semântico pré-treinado, são atribuídos com um vetor espacial (*word embeddings*, secção 2.4.3.2). De seguida, o codificador de posição adiciona, através do cálculo de outro vetor numérico, o contexto de cada termo consoante o seu índice. A sequência abaixo representa um exemplo da codificação do termo *"dog"*.

$$\text{dog} = \begin{bmatrix} 0.37 \\ 0.99 \\ 0.01 \\ 0.08 \end{bmatrix} + Positional\ Encoding\ Function = \begin{bmatrix} 0.42 \\ 0.82 \\ 0.12 \\ 0.81 \end{bmatrix}$$

A função de codificação pode ser definida de forma personalizada, sendo que a Google [44] utiliza cossenos e senos. O vetor final representa a palavra contextualizada através da sua posição e significado.

O bloco de codificação é constituído por uma camada de atenção e uma rede neuronal *feed-forward*. A primeira relaciona todas as palavras de uma frase através da criação e atualização

de pesos. A Figura 16 ilustra o resultado deste procedimento, onde se liga o pronome *"it"* com os restantes termos. Neste caso, quanto mais escuro o laranja, maior a probabilidade de os elementos estarem relacionados. Posteriormente, a rede neuronal é utilizada para normalizar o conteúdo recebido, originando uma estrutura numérica preparada para a etapa de descodificação.

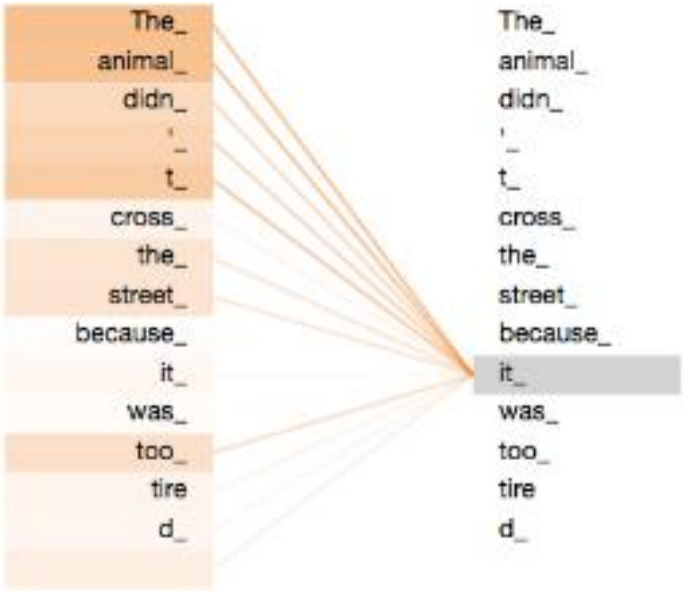

Figura 16. Exemplo de uma camada de atenção com a relação do pronome "*it"* com os restantes elementos da frase

Para o caso de agentes conversacionais com apenas uma linguagem, a utilidade dos transformadores corresponde apenas aos codificadores, sendo que a tradução não apresenta qualquer utilidade. Desta forma, os descodificadores não são abordados nesta dissertação.

Após a publicação do artigo, vários modelos de transformadores, denominados de modelos de linguagem, foram criados. O melhoramento substancial do desempenho em relação às redes LSTM originaram, através de corpos de texto enormes, codificadores pré-treinados expostos ao público. A área de compreensão de linguagem natural foi revolucionada, uma vez que estes modelos, para além de terem melhores resultados, podem ser implementados com os pesos pré-definidos, evitando a fase de treino pesada. Atualmente, os mais populares são o BERT [45], RoBERTa [46] e o GPT2 [47].

2.4.3 Processamento de Linguagem Natural

Anterior à execução de qualquer método de aprendizagem profunda, é necessário fazer a preparação de dados. Esta fase é constituída por métodos de PLN, com o intuito de transformar o conteúdo recebido em texto para uma forma previsível e adequada. Pode-se segmentar em duas etapas, nomeadamente o pré-processamento de dados e a extração de significado.

2.4.3.1 Pré-processamento de dados

A fase de pré-processamento de dados tem como objetivo preparar a compreensão de texto através da sua generalização. É importante considerar que a conversão para um formato que o sistema possa compreender, naturalmente, reduz a fidelidade da informação introduzida, sendo que a utilização destes procedimentos implique uma decisão ponderada.

A ***tokenization*** consiste na divisão de texto em estruturas reduzidas, ou *tokens*. O seu objetivo principal é a conversão de frases em elementos individuais, utilizando espaços, sinais de pontuação ou carateres especiais para a sua separação (Tabela 3). De forma complementar, também pode ser utilizada para a conversão de parágrafos nas suas frases internas. É a etapa principal de pré-processamento visto que prepara as palavras para os restantes métodos, denotados de seguida.

Frase	**Frase dividida em tokens**				
Onde fica o ISEP?	Onde	fica	o	ISEP	?

Tabela 3. Exemplo de *tokenization* para uma frase portuguesa

Existem cenários onde a *tokenization* pode corroer o significado da palavra, como por exemplo abreviações, nomes com pontuação, o uso de conjunções com pronomes e até especificações particulares a um idioma. Contudo, os algoritmos de *tokenization* já são apropriados à maioria das linguagens.

Anterior à interpretação semântica das palavras, podem-se processar técnicas como a **capitalização** e a **remoção de caracteres irrelevantes ao significado do texto**. Respetivamente, a primeira refere-se à normalização da frase, convertendo todos os caracteres para letra minúscula ou maiúscula.

Por outro lado, métodos como a eliminação de pontuação e a remoção de números (ou a sua transformação para representações textuais) também podem ser aproveitados, ainda que muitas vezes já estejam incluídos na *tokenization*.

A **remoção de *stopwords*** denota a eliminação de palavras de conexão, visto que estas não apresentam qualquer adição ao significado da frase. Usualmente, trata-se de classes gramaticais como pronomes, articuladores, conjunções e proposições. Como observado pela Tabela 4, ainda que reduzida, a frase final é mais percetível que a inicial. Para além de resultar na poupança de processamento, a remoção destas palavras apoia na definição do intuito.

Frase	***Stopwords***			**Final**
A Joana foi para a piscina?	A	para	a	Joana foi piscina?

Tabela 4. Exemplo da remoção de *stopwords* para uma frase portuguesa

Por fim, com o objetivo de normalizar o seu significado, existem algoritmos de manipulação morfológica de palavras. Em termos linguísticos, a morfologia refere-se ao significado, estrutura

e a relação entre vocábulos, sendo que, para a fase de pré-processamento, estas técnicas são utilizadas para reduzir as derivações de palavras. Ao convertê-las para uma forma base, o sistema consegue não só reconhecer os termos originais e respetivos desvios, como também aprender novas ramificações.

Existem dois tipos de transformações morfológicas, nomeadamente a conversão da palavra para um *stem,* a unidade base de uma palavra sem prefixos ou sufixos, ou então para um *lemma,* a forma base da palavra no dicionário.

O processo de ***stemming*** define-se como a eliminação de sufixos e prefixos de palavras para obter a sua raiz. A Tabela 5 referencia um exemplo da utilidade deste processo, onde duas questões, efetuadas com o mesmo intuito através de palavras diferentes, são normalizadas para uma frase semelhante. Isto possibilita a interpretação de forma igual por parte do sistema.

Frase	**Frase após *stemming***
Como posso correr?	Com poss corr?
Como faço corrida?	Com faç corr?

Tabela 5. Exemplo de *stemming* para uma frase portuguesa

A ***lemmatization*** é uma alternativa ao método de *stemming,* utilizando dicionários e a análise morfológica das palavras para as converter na sua forma canónica.

Frase	**Frase após *lemmatization***
Eu estou a ***estudar***	Eu estou a **estudar**
Eu estou ***estudando***	Eu estou **estudar**

Tabela 6. Exemplo de *lemmatization* para uma frase portuguesa

Enquanto os algoritmos de *stemming* dependem de um dicionário de sufixos e prefixos, os algoritmos de *lemmatization* necessitam de reconhecer o vocabulário completo de uma linguagem. Ainda que a sua construção carregue mais complexidade, os algoritmos de *lemmatization* são mais abrangentes e precisos. Adicionalmente, a maior diferença entre os dois métodos é o facto de este último retornar sempre um vocábulo verdadeiro, enquanto, como demonstrado na Tabela 6, o *stemming* pode retornar palavras irregulares.

2.4.3.2 Métodos de extração de significado

Para além de limpeza de dados, os sistemas de interpretação de texto necessitam de compreender o seu valor semântico. Estes métodos são empregues na fase de aprendizagem de modelos inteligentes, convertendo texto sem qualquer estrutura, os *tokens*, para representações que possam ser utilizadas para comparações de valor.

Bag of words (BOW): O método tem como objetivo contar a ocorrência de cada palavra independentemente da estrutura, da ordem e da sintaxe da frase. É indicado para uso em

conjunção com redes neuronais para conseguir recolher qual a intenção do discurso do utilizador, recorrendo à comparação de palavras através de valores numéricos (Tabela 7).

Frase	Bom	Filme	Gostei	Do	Fui	Ao	Cinema
Bom filme	1	1	0	0	0	0	0
Gostei do filme	0	1	1	1	0	0	0

Tabela 7. Exemplo de utilização de *Bag of Words*

É importante referir que esta técnica é normalmente processada em conjunto com a remoção de *stopwords* e a *lemmatization* para que, respetivamente, termos insignificantes e derivações da mesma palavra não influenciem o processo de correspondência.

Latent Semantic Analysis (LSA): LSA é um algoritmo **s**emelhante a*o* imediatamente acima*,* contudo, utiliza conceitos compostos como unidade de comparação. Após agrupar palavras que ocorram frequentemente, através de *Single Value Decomposition* [48], constrói uma matriz que é definida horizontalmente por tópicos analisados e, verticalmente, pelos documentos de estudo. Cada célula representa a frequência destes tópicos nos corpos (Figura 17).

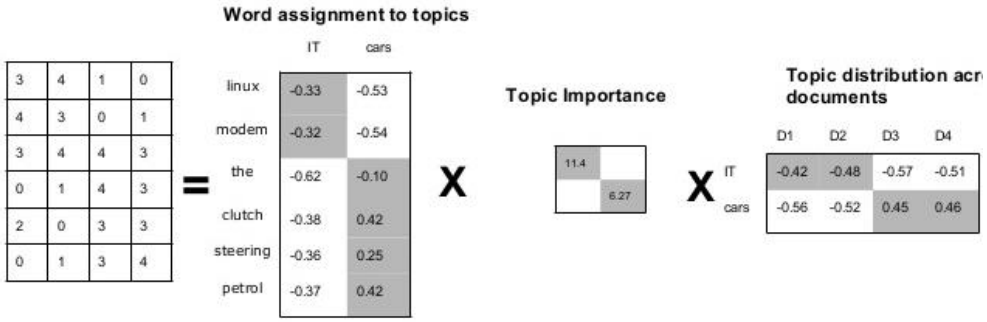

Figura 17. Fluxo de processamento do método de *Latent Semantic Analysis* [49]

Part-of-Speech tagging (POS-Tagging): O método efetua a categorização de cada palavra ou *token* num corpo de texto tendo em conta a sua classe sintática. Dado que cada linguagem tem as suas próprias estruturas gramaticais, é um processo que é bastante dependente da linguagem utilizada [24]. Ao contrário da *tokenization*, *POS-tagging* tenta enquadrar a palavra num contexto ainda que não saiba o seu significado, utilizando palavras precedentes ou sucessoras (Figura 18). O seu funcionamento baseia-se na correspondência das palavras com um dicionário.

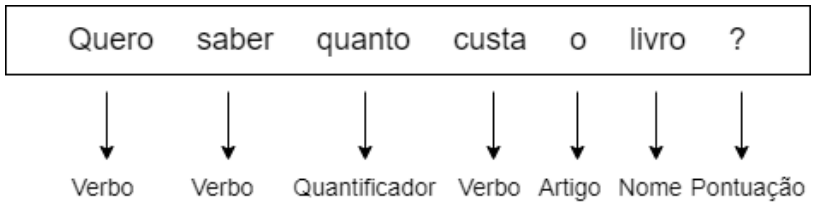

Figura 18. Exemplo de *POS-Tagging* para uma frase portuguesa

Named Entity Recognition (NER): Ao contrário de *POS-Tagging*, que identifica as funções sintáticas de cada palavra, NER pretende categorizar os termos como entidades (Figura 19). É útil para manter o contexto da conversa do utilizador, processado pelo componente de gestão de diálogo. Normalmente, os algoritmos de NER utilizam a deteção de capitalização e de limites da frase para conseguir determinar possíveis referências [50].

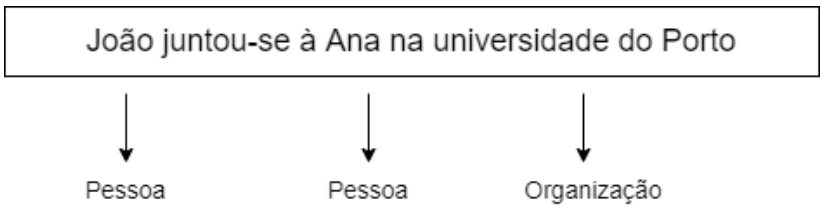

Figura 19. Exemplo de *Named Entity Recognition* para uma frase portuguesa

Word Embedding: O termo refere-se ao ato de transformação de uma palavra para forma vetorial, com o intuito de agrupá-las espacialmente através do seu valor semântico. Os algoritmos utilizam corpos de texto para conseguir obter contexto relativamente ao significado de cada palavra, atribuindo valores numéricos para o seu índice de similaridade. Word2Vec [51] é uma técnica que calcula a proximidade através da frequência de ocorrência das palavras, utilizando redes neuronais (Figura 20).

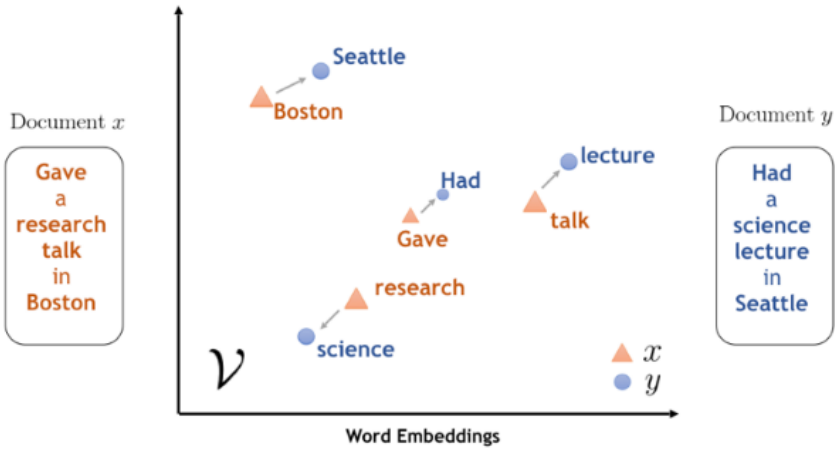

Figura 20. Demonstração espacial do resultado de *Word Embedding* [52]

2.4.4 Compreensão de Linguagem Natural

Dado que a utilização independente de métodos PLN não é suficiente para a compreensão de discurso, estes são empregues em conjunto com modelos de aprendizagem profunda. Ao converter as palavras para valores numéricos, estes são capazes de induzir a sua similaridade.

Visto que o PLN é altamente dependente do contexto da frase, as redes neuronais mais relevantes são as recorrentes, particularmente, as LSTM. No entanto, recentemente, os transformadores têm ganho bastante popularidade devido não só ao seu desempenho, mas também pela sua facilidade de uso [44].

A Figura 21 denota, concetualmente, a sinergia entre o uso de PLN e de aprendizagem profunda, através das duas fases cruciais, o treino do agente e a sua execução. É importante denotar que os métodos relativos à fase de treino são meramente ilustrativos, dado que estes são definidos de acordo com as intenções do desenvolvedor.

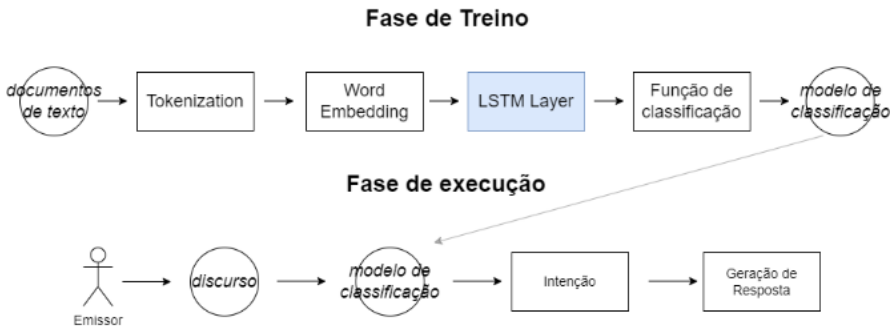

Figura 21. Exemplo de um processo de extração de intenção

Assim que o modelo de classificação esteja parametrizado, após a fase de treino, o agente utiliza-o para converter o discurso do utilizador na intenção mais provável. Naturalmente, a precisão do agente depende de vários fatores, especificamente a fase de PLN empregue, o método de aprendizagem profunda e, por fim, o conjunto de informação de treino introduzido.

2.4.4.1 Classificador DIET

O classificador DIET, ou *Dual Intent Entity Transformer*, é um tipo de arquitetura de transformadores desenvolvido pela equipa RASA, capaz de interpretar entidades e intenções em simultâneo [53]. Utilizando o exemplo da frase *"play ping pong",* a arquitetura é explicada através da Figura 22.

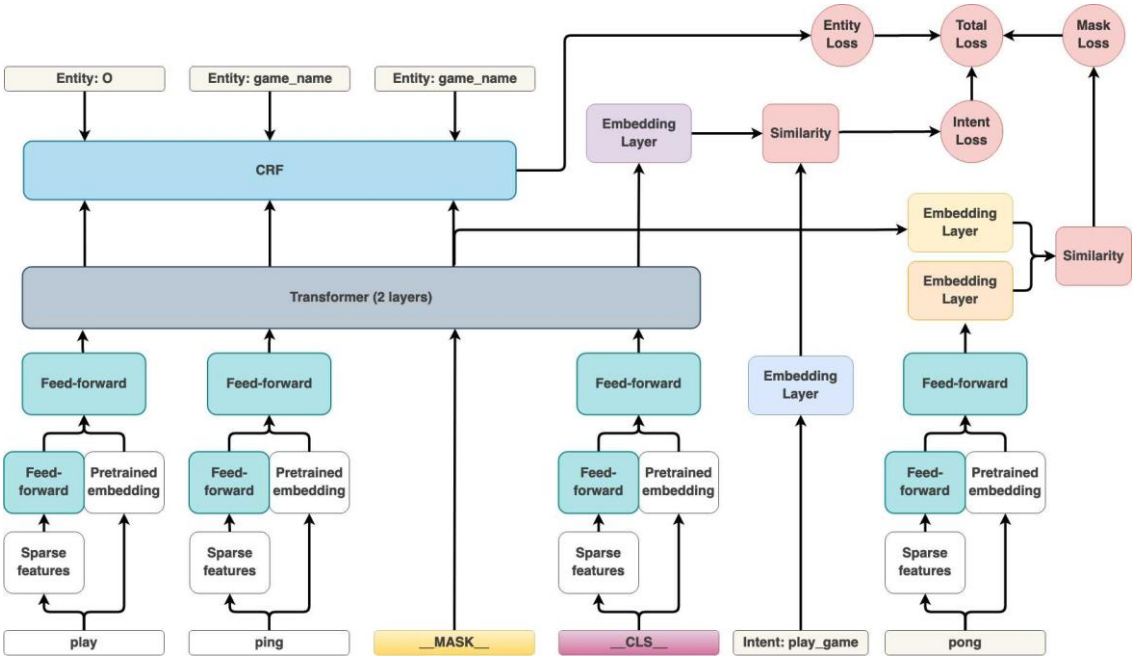

Figura 22. Arquitetura do classificador DIET tendo em conta um exemplo de frase [53]

Os blocos brancos simbolizam os algoritmos de PLN, que tem como intuito a conversão de elementos textuais para vetores numéricos, adicionando algum tipo de significado. Em contrapartida, os blocos azuis representam a vertente inteligente do modelo, nomeadamente as redes neuronais *feed-forward* e os transformadores, que aplicam transformações a estes vetores. É importante referir que os pesos de cada rede são partilhados entre si, sendo que o intuito do modelo é receber exemplos de treino e, consoante a comparação das previsões com os exemplos reais, ajustar estas ligações.

A metade inferior da figura compõe os blocos de processamento de *tokens,* divididos em duas ramificações, as funcionalidades dispersas, no lado esquerdo, e o uso de redes pré-treinadas, no lado direito (Figura 23). Respetivamente, a primeira representa a normalização das funções de PLN através de uma rede neuronal, enquanto a segunda demonstra a conversão de uma palavra para um *word embedding*, consoante um modelo de linguagem já definido, como o BERT (secção 2.4.2).

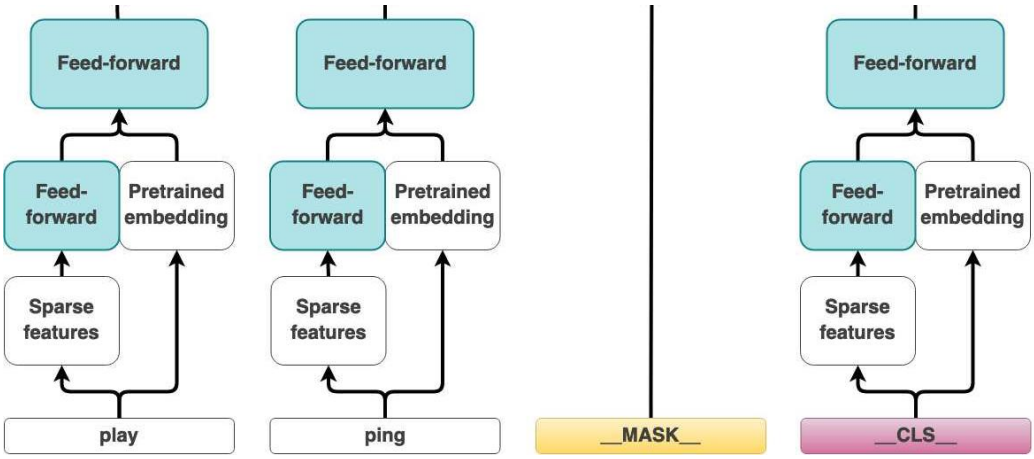

Figura 23. Blocos de processamento de *tokens* [53]

Caso DIET esteja configurado para utilizar ambas as ramificações, todos os *tokens* são processados repetidamente. Desta forma, cada uma produz um vetor numérico, que é normalizado através das camadas *feed-forward* finais (Figura 24).

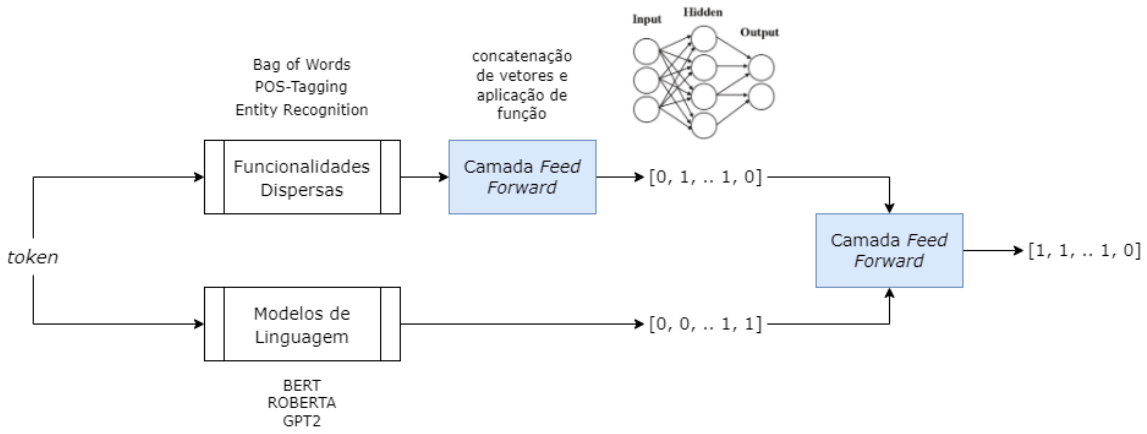

Figura 24. Funcionamento interno dos blocos de processamento de *tokens*

Para cada exemplo de treino, para além das suas palavras, existem dois elementos especiais, a máscara __*MASK*__ , definida a amarelo, e a classe __*CLS*__, definida a rosa. A máscara representa um *token* escondido propositadamente pelo classificador, de forma a adivinhar qual a palavra em falta, enquanto a classe refere um valor numérico da frase como um todo.

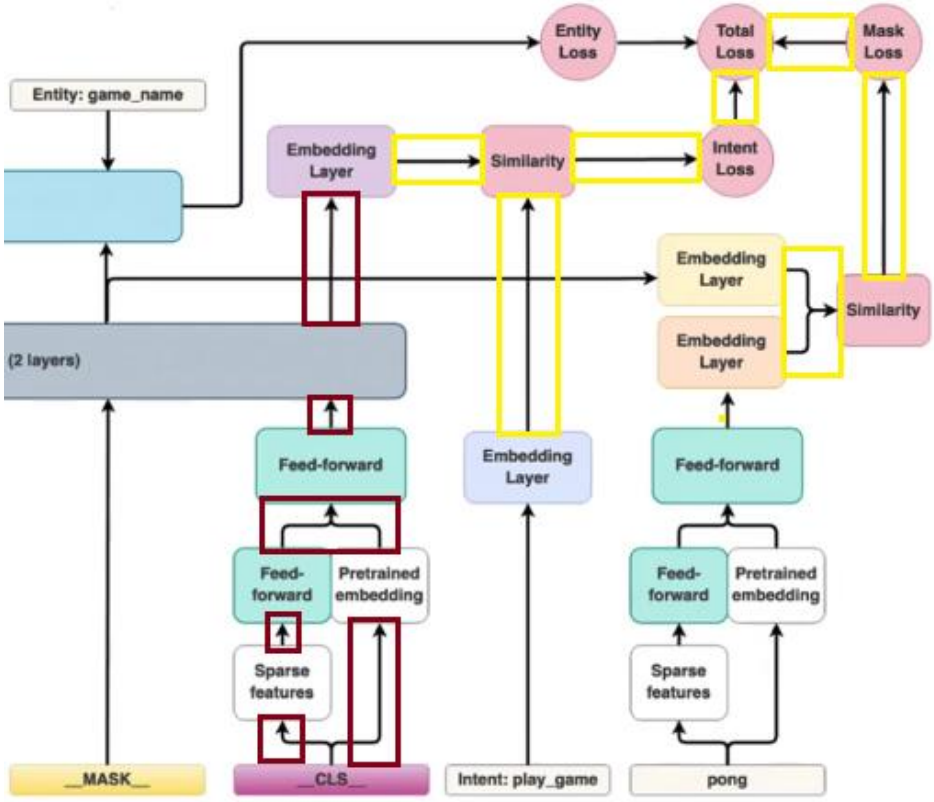

Figura 25. Sequência de processamento para os elementos especiais, com o caminho de comparação delineado a amarelo e a obtenção de intenção a vermelho

O critério de atualização das camadas *feed-forward* é a comparação de similaridade entre os elementos de validação e a máscara e a classe (caminho amarelo na Figura 25). Respetivamente, a primeira refere a perda de intenção, enquanto a segunda de classe. Estas duas medidas, em conjunto com o desvio de entidades, compõe a perda total, que é a métrica utilizada para a atualização dos pesos até ao fim das iterações de treino. No caso atual, as entidades não foram exploradas porque não são utilizadas.

Assim que esteja treinado, a obtenção da intenção final é realizada através do elemento especial __*CLS*__. Como delineado a vermelho na Figura 25, após a adição do contexto pelos transformadores, a previsão realizada pelo modelo é a intenção resultante, transformada pela camada de *embedding*. Em vez de ser comparada com os exemplos de validação, é devolvida ao utilizador.

Esta arquitetura é vantajosa uma vez que apresenta uma elevada modularidade em relação aos algoritmos de linguagem natural, suportando a utilização tanto de métodos de pré-processamento de texto como de modelos de linguagem.

2.5 Ferramentas de desenvolvimento de chatbots

No que toca a ferramentas de desenvolvimento de *chatbots*, é importante fazer a distinção relativamente ao seu objetivo final. As plataformas de construção rápida pretendem disponibilizar um fluxo pré-definido de decisões para a construção de diálogo em alto nível, através de regras, sem qualquer conhecimento de tecnologias ou técnicas, enquanto as plataformas e *frameworks* de desenvolvimento disponibilizam funções e algoritmos adequados à sua personalização. Para além disso, existem canais de comunicação que possibilitam a configuração de agentes conversacionais. A presente seção irá explorar todos os casos, sendo que o maior detalhe será associado à vertente de desenvolvimento.

2.5.1 Canais de Comunicação

Com a popularização de *chatbots*, os canais de comunicação começaram a disponibilizar recursos para a sua configuração. Embora o projeto desenvolvido já tenha um local de implantação estipulado, o estudo precoce destas plataformas é realizado para compreender a ligação entre o ponto de entrada de informação e o sistema de resposta.

O **Slack** [54] é uma plataforma digital dedicada maioritariamente a ambientes profissionais, permitindo a criação e manutenção de canais de conversa direta. No que toca a interação inteligente, suporta dois tipos de *chatbots*: o *reply bot e o interaction bot.*

O *reply bot é* a derivação mais básica de um agente baseado em regras, com o objetivo de intercetar mensagens introduzidas através da explicitação manual de expressões. Cada uma terá respostas associadas, sendo que não existe qualquer tipo de adaptação de contexto.

O *interaction bot* é um agente que depende unicamente de componentes externos para definir o seu comportamento, agindo apenas como um intermédio de comunicação. A ligação entre o agente e o módulo externo é realizado através de *webhooks* [55], ou seja, a interação com outros sistemas *web* definidos pelo utilizador através de comunicação HTTP.

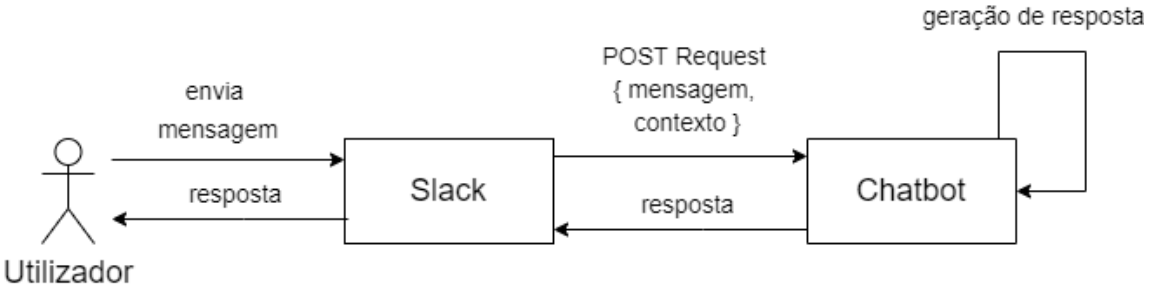

Figura 26. Funcionamento de alto nível da ligação entre o *chatbot* e o Slack

Estes pedidos são espoletados através de eventos nativos do Slack, especificamente o uso de comandos específicos, menções ou, normalmente, o envio de mensagens para o utilizador definido como agente. Para além disso, com o intuito de ajudar o *chatbot* com o controlo de contexto, gera um identificador único para cada conversa em processamento.

2.5.2 Plataformas de construção rápida

As plataformas de construção rápida permitem ao cliente desenvolver um *chatbot* totalmente funcional sem qualquer preocupação das tecnologias e/ou algoritmos que o constituem. Por norma, são baseados em interfaces visuais simples que admitam a um utilizador sem conhecimentos técnicos fazer a definição do seu comportamento. O fluxo é básico, conduzido por decisões do utilizador em vez de compreensão de linguagem natural.

De acordo com um estudo efetuado em 2020 [56], considerando a facilidade de implementação, a integração com canais de comunicação, a recolha de informação do utilizador e os tipos de resposta, os exemplos mais populares são o *Wotnot* [57] e *Landbot* [58]. Para efeitos de simplificação, apenas uma plataforma será explorada, visto que esta análise apenas servirá para a compreensão de quais as funcionalidades que poderão ser úteis no ponto de vista dos configuradores de informação. Para além disso, a escolha destes serviços não é viável visto que, para além de apresentarem taxas de pagamento, não permitem um nível alto de detalhe.

O Landbot.io [58] é uma solução que admite o desenvolvimento de agentes interativos exclusivamente a partir do estabelecimento de regras, destinado ao consumo de perguntas. O desenho do sistema assenta num modelo condicional, utilizando uma árvore de decisão visual.

A Figura 27 representa a interface do serviço, utilizada para compor um cenário semelhante ao do projeto. A definição do comportamento segue um fluxo, que, dependendo da escolha do utilizador, denotados através dos botões vermelhos, resultará na apresentação de tópicos específicos. Cada tópico tem um grupo de perguntas e respostas que, novamente, será retornado caso o utilizador o escolha.

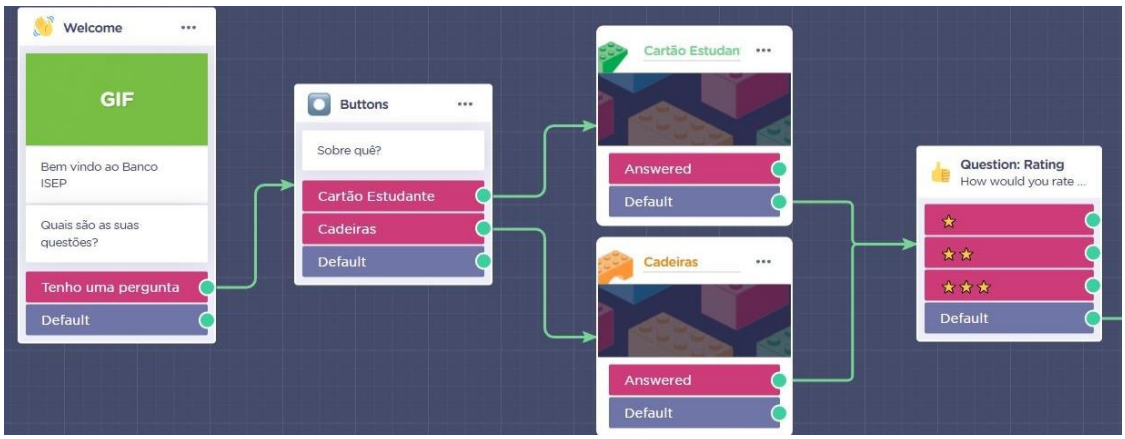

Figura 27. Esquematização de um fluxo conversacional em Landbot.ai

A simplicidade é o ponto forte da plataforma, dado que facilmente se cria uma corrente de decisões com a informação necessária para o cliente final. Em contrapartida, a sua complexidade aumenta exponencialmente com o número de ramificações na conversa.

2.5.3 Frameworks de desenvolvimento

A presente secção apresenta as *frameworks e* bibliotecas mais relevantes em termos de *chatbots,* tendo em consideração a sua aplicação de PLN e o seu suporte para funcionalidades de aprendizagem profunda. O alvo de estudo será a combinação das técnicas apresentadas na seção 2.3.

Devido à falta de revisão sistemática de serviços de *chatbots*, Braun [25] realizou um estudo em que avaliou sistemas de interfaces conversacionais tendo em conta o seu entendimento de linguagem natural. Destacou os cinco serviços mais populares, nomeadamente LUIS, IBM Watson, Wit.ai, Dialogflow, Amazon LEX, baseados em implementações na nuvem e, por fim, RASA, uma solução local. Adicionalmente, um artigo realizado por Thorat [59] em 2020 comparou as tecnologias existentes para um contexto semelhante ao da dissertação: um agente de perguntas e respostas. Avaliou IBM Watson e Dialogflow através da sua integração com canais de comunicação, a capacidade de recolher dados do utilizador através de interação contínua e, por fim, o suporte de linguagem natural. Ainda que não existam muitos artigos que refiram quais as mais populares, utilizou-se a IBM Watson, a Dialogflow e RASA como alvo de estudo, uma vez que também foram referidas como as mais proeminentes por outros autores [60].

2.5.3.1 Dialogflow

Dialogflow [61] é uma solução da Google baseada em ML, implantada totalmente na sua nuvem. Ainda que a organização não exponha detalhes da sua implementação, internamente, o agente utiliza algoritmos de aprendizagem profunda para treinar modelos através dos seus motores de busca.

É possível dinamizar qual a informação de moldagem, introduzindo diálogos customizados que são considerados na fase de treino. Em execução, estes modelos são empregues para a extração de informação do discurso do utilizador, convertendo-os para a lista de intenções pré-definidas e a respetiva probabilidade de confiança (Figura 28). O sistema aprende com a interação contínua, adicionando informação à base de treino automaticamente

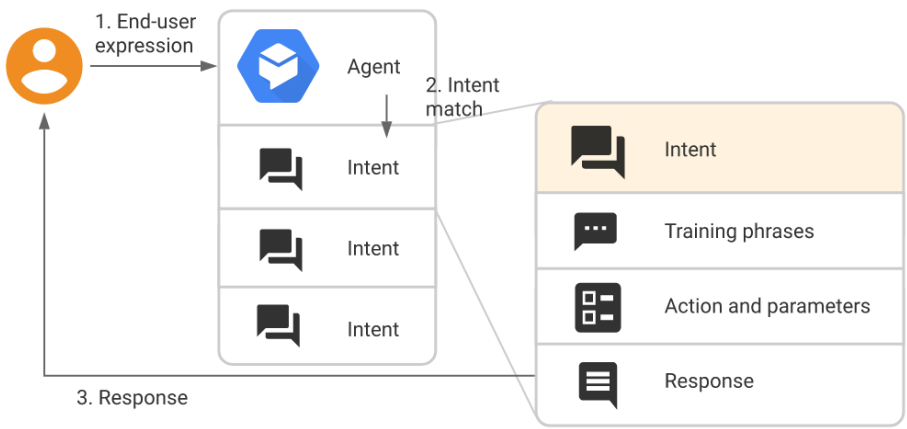

Figura 28. Diagrama funcional de alto nível relativo a Dialogflow [61]

Por fim, Dialogflow fornece já um conjunto pré-configurado de entidades, como organizações, e nomes, permitindo ainda a sua personalização manual. Toda a configuração referida nesta seção é realizada através de uma interface visual na plataforma Google.

2.5.3.2 IBM Watson

IBM Watson Assistant [62] é um projeto sustentado pela IBM com o objetivo de construir agentes virtuais através de CLN. É considerada a melhor solução do mercado, suportada por estudos que demonstram que tem a melhor taxa de acerto na classificação de significado [63] [25]. O seu conceito principal é o nó de diálogo, onde se instrui o comportamento do agente caso seja detetada uma intenção ou uma entidade. Este modelo permite conectar os nós entre si, sendo que motiva o uso de modularidade e a utilização de rotinas frequentes (Figura 29).

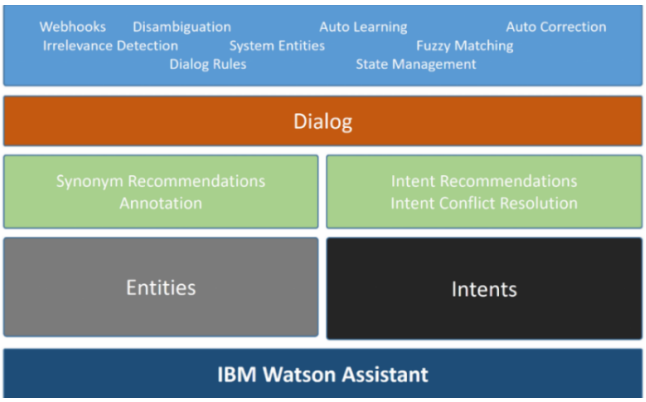

Figura 29. Arquitetura concetual de IBM Watson [62]

A classificação de intenção é realizada através da indução de exemplos de frases, enquanto as entidades são introduzidas através de valores específicos.

2.5.3.3 RASA Stack

RASA é uma solução *open-source* que, tal como as anteriores, suporta a criação de chatbots com foco em CLN. Respetivamente, segmenta-se em dois módulos, o RASA Core e o RASA NLU. O primeiro destina-se à gestão de diálogo, possibilitando a análise de contexto e a escolha de ações, enquanto o segundo é responsável pela extração de entidades e de intenções. De forma a possibilitar o uso dos componentes individualmente, o projeto é constituído através de uma arquitetura modular, como apresentado na Figura 30.

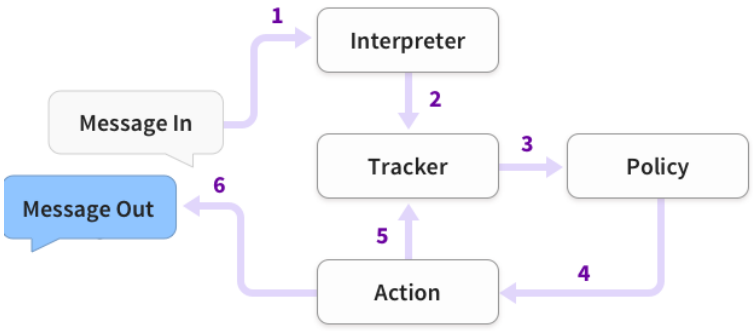

Figura 30. Arquitetura interna de Rasa Stack através de uma perspetiva sequencial [64]

A sua estrutura é uma derivação da solução apresentada na seção 2.3, sendo que *Interpreter* assume a função de interpretador da linguagem de entrada, o par *Tracker* e *Policy* são o gestor de diálogo e, por fim, *Action* representa a geração de resposta.

RASA NLU é o componente que fornece os métodos associados à fase de CLN, com algoritmos nativos da *framework* ou através da integração com outras bibliotecas Python. A responsabilidade do desenvolvedor cinge-se à enumeração de quais os métodos a usar e a sua parametrização. Embora possa ser alterado, o fluxo de interpretação de texto normalmente segue as seguintes fases:

- *Word Vector Sources*: implantação de modelos semânticos pré-definidos com a relação de proximidade entre palavras;
- *Text Featurizer*: conversão de texto para estruturas adequadas a funções de ML;
- *Intent Classifier*: modelos de ML para a extração de intenção;

```
1. nlu:
2.   - intent: PROCEDIMENTO_REALIZACAO_MATRICULA
3.     examples: |
4.       - Que papéis preciso para fazer a matrícula
5.       - O que fazer depois de aceite na universidade
6.       - Quais são as credenciais de acesso ao portal para matricular
7.
8. pipeline:
9.   - name: SpacyNLP
10.    model: pt_core_news_md ##wikipedia portuguesa
11.  - name: SpacyTokenizer
12.  - name: LexicalSyntacticFeaturizer ##pos-tagging
13.  - name: CountVectorsFeaturizer ##bag of words
14.  - name: CountVectorsFeaturizer
15.    analyzer: "char_wb"
16.    min_ngram: 1
17.    max_ngram: 4
18.  - name: DIETClassifier
19.    epochs: 200
```

Código 2. Extrato do ficheiro de configuração em RASA, utilizando funcionalidades da biblioteca SpaCy e um dicionário pré-definido Português baseado na Wikipedia [65]

Por fim, suporta outras funcionalidades como a definição de expressões regulares, entidades e a conversão de palavras para sinónimos pré-definidos.

2.5.4 Comparação das ferramentas

O estudo de Braun [25] estabelece duas hipóteses iniciais: 1) os serviços comerciais são melhores que os gratuitos e 2) a qualidade de classificação depende da informação de domínio. Na fase de verificação, empregou dois corpos de texto, um com questões anotadas com intenções e outro com pares questões respostas. A primeira hipótese não foi verificada, dado que a solução RASA reagiu tão bem como as restantes. A segunda foi averiguada visto que denotou que certos corpos de texto tinham uma pior prestação dependendo do domínio

inserido nas ferramentas. Em contrapartida, para um corpo de texto não anotado, um relatório exposto pela IBM [63], denotou que a precisão de IBM Watson (70.6%) na extração de intenção era superior a Dialogflow (66.6%) e RASA (62.9%).

Ainda que a precisão seja um bom apontador, a capacidade do agente acaba por depender do domínio introduzido, como referido por Braun [25]. De acordo com as necessidades da dissertação atual, o modelo da seção 2.3 expõe as funcionalidades mais importantes de cada ferramenta (Tabela 8).

Solução	**Dialogflow**	**IBM Watson**	**RASA**
Open-source			X
Suporte de PLN	X	X	X
Personalização da *pipeline* de PLN			X
Suporte linguagem portuguesa	X	X	X
Preço	0.006€/mensagem	Grátis até 10mil mensagens/mês & 123€/mês	Grátis
Baseada em Cloud	X	X	
Interface visual	X	X	
Linguagem de programação	SDK para 14 linguagens	SDK para 10 linguagens	Python

Tabela 8. Comparação de funcionalidades das três *frameworks*

No contexto de implantação, as soluções comerciais fornecem um suporte superior em relação a RASA, visto que, para além de permitirem a configuração do agente através da *web*, fornecem integração com as suas plataformas de publicação. O seu objetivo é alcançar um intermédio entre uma solução técnica e uma solução de construção rápida (secção 2.5.2). Em contrapartida, a solução *open-source* RASA oferece um nível de personalização elevado.

Ao contrário de Dialogflow e IBM Watson, que utilizam a sua própria metodologia de PLN e pré-processamento de dados, RASA transfere essa responsabilidade ao desenvolvedor. Isto implica a restrição de linguagem de programação e, posteriormente, as bibliotecas utilizadas para esta fase.

Todas as soluções baseiam-se nos mesmos princípios: a definição de entidades, intenções e o processo de treino para um modelo de classificação. O maior ponto diferencial entre todas as *frameworks* é o esquema de inteligência artificial que vai definir o processo de aprendizagem. Embora ofereçam parâmetros de entrada, o uso destas ferramentas implica a aplicação do seu modelo de aprendizagem.

Adicionalmente, existem bibliotecas que, ainda que não sejam totalmente dedicadas a agentes conversacionais, possam ser utilizadas individualmente para a compreensão de texto. Destacam-se o Tensorflow [66], uma solução *open-source* desenvolvida pela Google com funções de aprendizagem automática, notoriamente conhecida por ser o motor do seu tradutor. Keras [67] é uma biblioteca construída com base em Tensorflow, mas fornecendo funções de mais alto nível para o uso de redes neuronais recorrentes como LSTM e GRU.

2.6 Trabalhos relacionados

Desenvolvido para um serviço de comércio eletrónico, Muangkammuen [68] expõe uma abordagem mais técnica para um agente de perguntas frequentes. Em termos de pré-processamento, juntou métodos como a *tokenization* e a conversão de palavras para um número inteiro através de *word embedding.* Construiu um modelo de classificação pelo treino de uma rede neuronal recorrente de três camadas que estabelece a relação das palavras através da proximidade do seu valor numérico, utilizando um nível LSTM como intermédio.

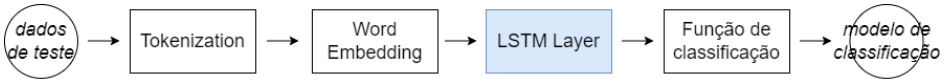

Figura 31. Ilustração da fase de aprendizagem referida em [68]

O modelo de classificação recebe uma pergunta como entrada, calculando a probabilidade de a frase estar associada a questões especificadas anteriormente. Para um universo de dados de 2600 perguntas, 60% foram utilizadas para treino, 20% para teste e, por fim, 20% para avaliação de desempenho. O resultado final incidiu em 83% de perguntas respondidas, sendo que 93% dessas resultaram em respostas corretas. É importante referir que esta verificação foi realizada através da variação de vários parâmetros, como o número de níveis escondidos da rede neuronal.

Peters [9] denota uma abordagem semelhante para a construção de um *chatbot* responsável pelo apoio ao cliente de uma organização de videojogos. Contudo, para além de uma camada interna com LSTM, realiza comparações com um nível de GRU invertida. Uma *Gated Recurrent Unit* (GRU) é semelhante a uma camada LSTM, no entanto, contendo apenas dois portões, de atualização e de eliminação.

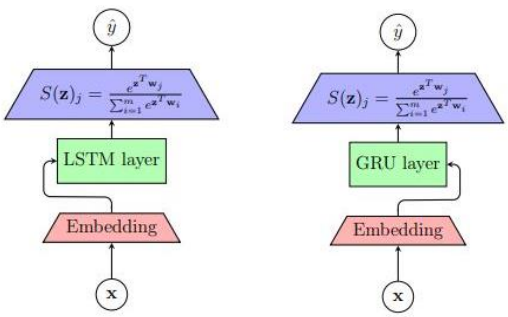

Figura 32. Arquitetura do modelo de classificação [9]

A maior diferença entre os dois casos foi a poupança de tempo no processo de aprendizagem, sendo que a camada LSTM demorou 314 minutos enquanto a GRU demorou 196. Para além disso, a segunda arquitetura denotou uma melhoria de 0.6% de precisão (87.6% contra 88.4%), sendo escolhida como solução final. O teste de desempenho definitivo foi efetuado para 3 linguagens, nomeadamente francês, holandês e inglês, das quais, respetivamente, originou uma precisão de 87%, 83% e, por fim, 70%. Atribui o mau comportamento deste último ao fato de ser a linguagem com menos dados de teste. Em termos de tecnologias, utilizou Keras para o módulo de *machine learning* e SciKit-Learn para o pré-processamento de dados.

Oliveira [10] utilizou IBM Watson para construir um agente conversacional do tipo questão-respostas para o *moodle* da sua universidade. Especificou cada pergunta como uma intenção, fornecendo respostas exemplo para o agente conseguir treinar o modelo de classificação. O seu universo de teste foi pequeno, com 12 utilizadores, sendo que 10 denotaram a experiência com 4/5 estrelas.

3 Análise de Valor

O capítulo atual representa todo o processo de análise de valor. Inicialmente, aborda-se a criação de inovação associada à dissertação, seguida do estudo do valor criado pela sua implantação. Por fim, efetua-se uma comparação das ferramentas expostas acima, estabelecendo critérios e respetivas prioridades para definir qual a *framework* mais adequada para desenvolvimento.

3.1 Criação de Inovação

A inovação é uma noção que está associada com a mudança. No contexto empresarial, refere-se à criação de produtos que se distingam da concorrência, sendo em termos de conceito, de custos, ou de qualquer outro critério. Para conseguir captar os fatores de inovação de cada produto, existem metodologias que propõe a definição de inovação através da análise de fatores associados à ideia.

A fase de criação de um produto segmenta-se em três partes, nomeadamente o *Fuzzy Front End* (FFE), o desenvolvimento do produto iterativo e, por fim, a comercialização [69]. O primeiro passo define a etapa de criação de ideias, onde se avalia oportunidades e conceitos independentes da tática de venda. É considerada a mais importante do processo, visto que as seguintes dependem exclusivamente do que seja decidido (Figura 33).

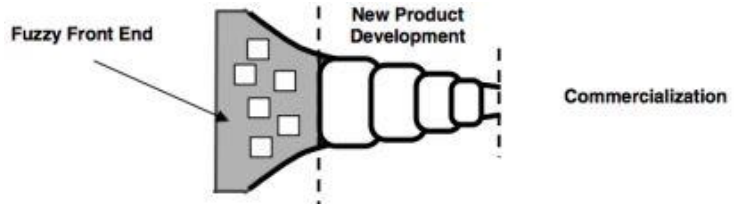

Figura 33. Processo de criação de produto [69]

Dado que esta fase não apresenta qualquer métrica nem formalidade de avaliação, existe um grande nível de abstração no que toca à classificação da inovação [69]. Para resolver este

problema, Koen [70] sugeriu um modelo que utiliza uma linguagem comum no processo de inovação, o *New Concept Development* (NCD). Segmenta-se em três partes, particularmente *wheel*, que representa as cinco atividades associadas ao FFE, *engine*, que constitui os elementos organizacionais que afetam diretamente ou indiretamente as atividades da roda, e, por fim, *rim*, que define os fatores exteriores que possam influenciar as duas fases anteriores [70].

O NCD pretende auxiliar na compreensão do produto a um nível mais detalhado, explorando fatores que possam ser ignorados aquando de uma análise técnica. Para além disso, permite avaliar quais as forças e fraquezas da organização relativamente ao conceito a desenvolver. Nas seguintes secções, exploram-se as atividades cruciais ao NCD, nomeadamente a identificação e análise de oportunidade e a geração e seleção de ideias.

3.1.1 Identificação de Oportunidade

A oportunidade principal acentua na diminuição de constrangimento nas vias de apoio ao estudante. Ainda que já exista suporte *online*, o horário da Divisão Académica não é suficientemente alargado para conseguir esclarecer os pedidos, sendo que as vias telefónicas e o correio eletrónico são limitados ao número de funcionários disponíveis. Com a resolução dos problemas mais frequentes, utilizando técnicas de linguagem natural, o *chatbot* pretende reduzir o fluxo de pessoas que necessitem de apoio humano.

Embora existam problemas que não serão passíveis de resolução, a implementação do *chatbot* permite a filtragem das questões mais simples. Para além do aumento da produtividade incutido na Divisão Académica, a experiência com o agente também pode aumentar o nível de satisfação do estudante. Independente do ISEP ser uma organização sem fins lucrativos, isto origina o melhoramento do bem-estar da comunidade escolar e, posteriormente, na sua recomendação.

Os *chatbots* inteligentes prosperam em ambientes repetitivos onde podem consumir a informação do utilizador para moldar o seu comportamento [15]. A Divisão Académica refere o seu processo de trabalho como sazonal, com conjuntos de perguntas efetuados regularmente em termos anuais, dependendo da época. Ainda que as regras possam mudar, este é um cenário ideal para o *chatbot*, possibilitando o melhoramento da sua base informação ao longo do tempo.

Por fim, um fator que não pode ser ignorado é a mudança de preferência do apoio ao cliente tradicional para um canal conversacional. Como referido anteriormente, o estudo realizado pelo Facebook denota que 66% dos seus inquiridos construíram uma maior relação de confiança com a marca quando utilizado um *chatbot* [14]. O relatório apresenta métricas de comparação entre o apoio ao cliente tradicional e o uso de canais de comunicação virtuais, abordando pessoas que os utilizaram (Tabela 9).

Canais de comunicação virtuais	Relação do agente conversacional com o negócio
59% refere um tempo de resposta mais rápido	**70%** pretendem utilizar as plataformas de comunicação mais frequentemente para dúvidas relacionadas ao negócio
50% refere um maior sentimento de auxílio	**59%** pretendem utilizar as plataformas de comunicação mais frequentemente para efetuar compras
49% refere um maior número de respostas reais	**68%** referem que o intermédio de comunicação melhorou a experiência do cliente

Tabela 9. Conclusões retiradas do estudo referido [14]

3.1.2 Análise de Oportunidade

Após a identificação de oportunidade, é necessário avaliar o seu processo de estabelecimento, particularmente as vantagens e desvantagens que a organização terá consoante esta decisão. Para isto, definiu-se uma análise SWOT, um processo que averigua as forças, fraquezas, oportunidades e ameaças relativas a um conceito [71]. Visto que o modelo aglomera a capacidade interna da organização com fatores externos que possam influenciar o resultado da solução, é um modelo relevante para esta secção.

Forças	Fraquezas	Oportunidades	Ameaças
Existência de uma base de informação com as perguntas mais frequentes	Período de adaptação e de treino	Expansão do agente para outros departamentos	Resistência ao uso do agente
Melhoramento do agente ao longo do tempo com aprendizagem automática		Adição de suporte multilinguagem para estudantes de ERASMUS	
Facilidade de escalabilidade da solução		Popularidade dos canais de comunicação	

Tabela 10. Análise SWOT relativa à implementação do agente conversacional

Como observado pela Tabela 10, denota-se um elevado número de forças, das quais se destaca a existência de uma base de informação com dados que possam ser utilizados diretamente para a moldagem do comportamento do agente. Para além disso, visto que a Divisão Académica é frequentemente utilizada pelos estudantes, o melhoramento da precisão do agente é expectável visto que quanto mais interações tiver, melhor a sua capacidade de reação.

No que toca às oportunidades, aliado à fácil escalabilidade da solução, apresenta-se a expansão do *chatbot* para outros departamentos e o possível uso da ferramenta para suporte à receção de membros estrangeiros. Para além disso, a situação pandémica e consequente popularidade dos canais de comunicação são uma aposta pertinente para a universidade.

Em termos de fraquezas, refere-se o período inicial em que o agente pode não ter capacidade para responder a todas perguntas, resultando no seu desuso. No entanto, com a quantidade de informação já disponível no departamento, induz-se como pouco provável. A única ameaça que se considera é a resistência dos estudantes ao uso do agente virtual, sendo que, caso aconteça, a sua capacidade não será melhorada ao longo do tempo.

De qualquer forma, a implementação do agente está associada a mais vantagens que desvantagens.

3.1.3 Geração e Seleção de Ideias

O próximo passo é a geração de ideias, ainda que de forma simplista, para selecionar as soluções que satisfaçam os requisitos propostos. O agente a desenvolver segue a estrutura definida pelos três componentes referidos na secção 2.3, aliado à implementação de uma interface visual no portal da universidade. Ainda que esta fase seja vítima de um processo de pesquisa e experimentação, as três alternativas que se colocam são:

1. A criação dos três componentes de raiz, implementando os processos de linguagem natural e os algoritmos de aprendizagem automática;
2. A utilização de uma plataforma de construção rápida, utilizando um sistema baseado em regras;
3. A utilização de *frameworks de* aprendizagem automática e PLN para a construção de um sistema baseado na análise e reação inteligente à linguagem natural.

Dado que os algoritmos de aprendizagem automática são bastante complexos, a primeira opção é naturalmente descartada. O tempo necessário à criação das técnicas ultimamente iria desviar o foco dos requisitos de negócio. Ainda que a segunda escolha seja interessante, não apresenta personalização suficiente para conseguir fazer experimentação. Desta forma, a opção três é a ideia selecionada, sendo que, para além de permitir a abstração da complexidade dos algoritmos de aprendizagem profunda, admite a sua configuração através de parâmetros de entrada.

3.2 Valor da Solução

De acordo com Neap [72], o termo valor reflete o nível de sentimento que o cliente tem para reter ou obter um produto. Está associado com as condições que o item pode sortir ao utilizador, dependendo de um número de fatores, como o preço. No entanto, o conceito também pode ser utilizado na perspetiva do produtor, onde se refere o retorno que os clientes podem ter para o negócio. De forma complementar, no seu estudo, Miles [73] refere que o valor que um produto oferece ao seu utilizador segmenta-se em quatro: o valor de uso, que está relacionado com as características que satisfaçam o objetivo do produto, o valor de estima, derivado das qualidades que atraem o cliente, o valor de custo, que representa a soma dos custos de produção e, por fim, o valor de troca, que refere o valor de um produto relacionado à troca com outros objetos do mercado. Ainda que não exista um consenso da definição de valor, é possível utilizar metodologias que normalizem o contexto e a perspetiva na análise de produtos.

3.2.1 Proposição de Valor

A proposição de valor é um modelo que apresenta o produto ou o serviço através das vantagens que traz para o cliente [74]. É um método conciso que permite à organização avaliar quais os ganhos e as perdas, em termos de valor, que o produto está associado através dos seus criadores.

No contexto do agente conversacional, o cliente direto é, efetivamente, o estudante. No entanto, o produto também traz ganhos para o departamento da Divisão Académica. Desta forma, o modelo de proposição de valor é segmentado entre dois pontos de análise, o estudante e a Divisão Académica (Figura 34).

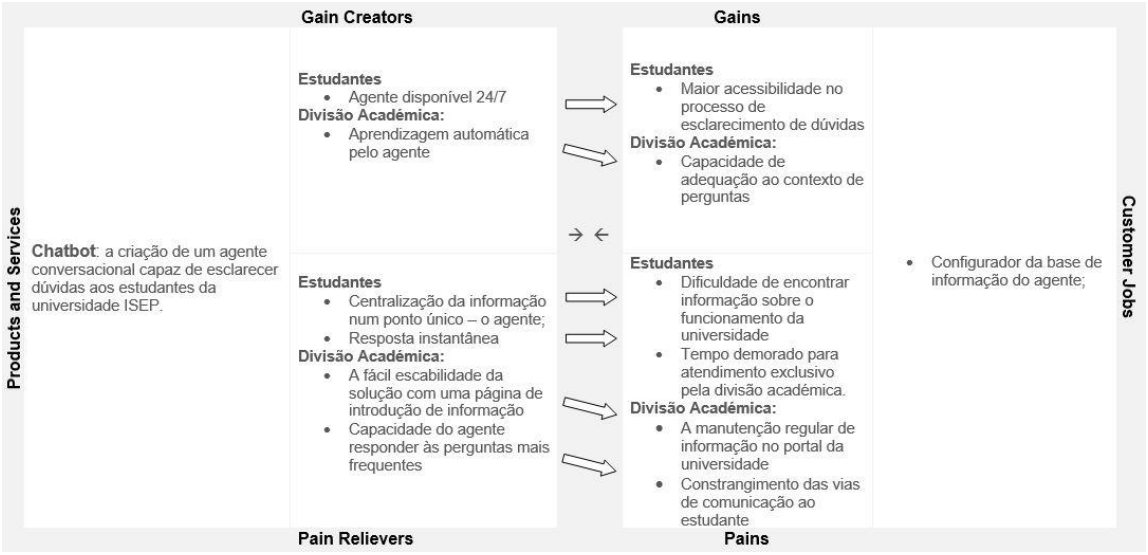

Figura 34. Modelo de proposição de valor para os estudantes e a Divisão Académica

Em termos de ganhos, refere-se o aumento da acessibilidade do agente comparada com o atendimento presencial, sendo que os estudantes terão um ponto de apoio disponível permanentemente. Para além disso, acentua-se a capacidade de adaptação do agente através dos mecanismos de aprendizagem automática.

Os maiores problemas associados ao estudante é o facto de não haver nenhum local centralizado com a informação da universidade, sendo que o agente vem a resolver esse problema. Para além disso, relacionado com a sua disponibilidade, o estudante terá sempre uma resposta instantânea. No contexto da Divisão Académica, denota-se a escalabilidade fácil da solução e ainda a diminuição do constrangimento das vias presenciais com a capacidade de resposta aos problemas mais frequentes. Por fim, existe ainda a criação de novos trabalhos, nomeadamente o configurador da base de informação do *chatbot*.

3.3 Priorização dos Requisitos

Quality Function Deployment (QFD) é uma metodologia de análise de produtos através da ligação dos requisitos de negócio e as características de engenharia da solução. Desta forma, para além de assegurar que todos os requisitos são endereçados no desenho do projeto, ordena-se a sua implementação através de um sistema de prioridades [75]. O artefacto final, denominado de *House Of Quality*, é constituído por uma matriz cujas linhas representam os benefícios e as colunas as funcionalidades a implementar. Para além deste núcleo, expõe ainda as prioridades de cada requisito e a correlação de cada funcionalidade, como observado pela Figura 35.

Correlations	
Positive	+
Negative	−
No Correlation	

Relationships	
Strong	●
Moderate	○
Weak	▽

Direction of Improvement	
Maximize	▲
Target	◇
Minimize	▼

Row #	Weight Chart	Relative Weight	Customer Importance (1-5 rating)	Customer Requirements \ Product Characteristics	Deep Learning	Natural Language Processing	Serviço CRUD para tópicos	Sistema de *rating* para interações	Componente de chat visual
				Column #	1	2	3	4	5
				Direction of Improvement	◇	◇	◇	◇	◇
1	■	18%	5	Importação do FAQ existente			●		
2	■	11%	3	Introdução de novos tópicos informativos			●		
3	■	11%	3	Entendimento de linguagem natural	●	●	○		
4	■	18%	5	Capacidade de resposta automática	●	●	○		
5	■	11%	3	Habilidade do utilizador dar feedback				●	○
6	■	18%	5	Entendimento de linguagem Portuguesa	●	●	○		
7	■	14%	4	Inteface visual intuitiva no Portal					●
				Technical Importance Rating	417,86	417,86	396,43	96,429	160,71
				Relative Weight	28%	28%	27%	6%	11%

Figura 35. *House of Quality* para a solução proposta

Customer Requirements referem os requisitos associados à Divisão Académica, sendo que se denotaram sete:

1. A importação da informação das perguntas mais frequentes, já existente numa página *web* no portal;
2. A possibilidade de introdução de novas perguntas e respostas para o processo de aprendizagem do agente;
3. O entendimento de linguagem natural, sem utilizar fluxos pré-definidos;
4. A capacidade de o agente responder adequadamente;
5. Um sistema de avaliação para o estudante conseguir qualificar a sua experiência;
6. O entendimento do dialeto português (mesmo que seja a partir de um fluxo de regras);
7. A implantação de uma interface visual para o *chat* na página da Divisão Académica.

Em termos de prioridade, o ponto 1, 4 e 6 obtêm pontuação máxima, visto que são a base de funcionamento do agente. Ainda que a introdução de novos tópicos seja importante para escalabilidade de solução, não é necessária para o produto mínimo viável. O mesmo se aplica ao sistema de avaliação e o ao entendimento de linguagem natural, visto que, respetivamente, se trata de melhorias ao ecossistema do *chatbot*.

Para responder às funcionalidades propostas, expõe-se cinco qualidades técnicas:

1. A utilização e configuração de métodos de aprendizagem profunda para o agente conseguir reagir às perguntas dos utilizadores. Isto será possível tendo em conta a base de informação existente, nomeadamente os tópicos informativos.
2. A implementação de um módulo de PLN para o agente conseguir interpretar as frases recebidas;
3. A criação de um serviço CRUD, com criação, atualização e eliminação de tópicos informativos para a base de conhecimento do agente;
4. O uso de um sistema de avaliação, revisto em cada interação, para melhorar a aprendizagem do sistema inteligente;
5. O componente visual associado ao *chatbot*, permitindo ao utilizador enviar e receber informação.

As primeiras duas funcionalidades apresentam uma correlação elevada, dado que ambas contribuem para a satisfação dos requisitos 3, 5 e 6. A interpretação das mensagens do utilizador, através de linguagem natural, e respetiva capacidade de resposta é resultado dos algoritmos de aprendizagem automática e da etapa processamento de PLN. Por outro lado, o serviço de gestão de tópicos informativos também tem uma influência acentuada para a maioria dos requisitos, sendo um ponto de dependência para o 1, 2, 3, 4 e 6.

A moldagem do comportamento do agente vai ser realizada através desta base de informação, sendo que é vital para os pontos das características anteriores. Adicionalmente, os tópicos informativos são utilizados como ponto de geração de resposta. Por fim, o sistema de avaliação e a interface visual respondem diretamente aos requisitos 4 e 6, respetivamente.

Em suma, o artefacto refere a importância relativa às três funcionalidades principais, nomeadamente o módulo de PLN, a capacidade de aprendizagem e, por fim, o serviço de gestão de tópicos. Estes consistem no esqueleto de funcionamento do agente conversacional, sendo que sem eles não existe aptidão de resposta.

3.4 Decisão de tecnologias

Com o intuito de decidir quais as tecnologias a recorrer, utiliza-se o método de decisão multicritério *Analytic Hierarchy Process* (AHP). O seu objetivo é a priorização de possíveis hipóteses, neste caso, as ferramentas definidas na secção 2.5, tendo em conta um conjunto de critérios quantitativos e qualitativos [76]. A decisão final é realizada após sete passos:

1. A construção da árvore de decisão hierárquica;
2. A comparação entre os critérios decididos;
3. A definição da prioridade relativa de cada critério;
4. A avaliação da consistência das prioridades;
5. Cálculo da matriz de comparação para cada par de critérios;
6. Cálculo da prioridade composta;
7. A decisão final da melhor hipótese.

As subsecções deste capítulo dividem as indicações denotadas acima em três fases, nomeadamente a segmentação hierárquica, a definição de prioridade e, por fim, a decisão final.

3.4.1 Segmentação Hierárquica

O primeiro passo é a estruturação do problema em camadas, normalmente definido sequencialmente através do objetivo, critérios de avaliação e, por fim, as alternativas. Como referido na secção anterior, o objetivo consiste na escolha das ferramentas tecnológicas a utilizar relativamente ao desenvolvimento do *chatbot*. Denotaram-se fatores que têm em interesse os requisitos da Divisão Académica e o nível de experimentação possível, nomeadamente:

1. **Custo de desenvolvimento**: os custos associados com a implementação das ferramentas;
2. **Complexidade de desenvolvimento**: a dificuldade de aprendizagem das funções fornecidas pelas ferramentas;
3. **Escalabilidade da solução**: o nível de expansão do agente, incluindo a integração com serviços exteriores e a capacidade de lidar com um número elevado de tópicos e entidades;
4. **Personalização de técnicas de PLN/ML**: a possibilidade de personalização da sequência de PLN /ML ou se têm métodos pré-definidos;
5. **Gestão de entidades e intenções**: a existência de suporte pré-configurado para a criação e manutenção de entidades e intenções.

Dado que não existe priorização possível, os critérios que são implementados por todas as ferramentas foram omitidos. Exemplos destes são o suporte da linguagem portuguesa e a interpretação da base de dados de diferentes formatos.

No contexto da exploração efetuada na secção 2.5.4, consideram-se quatro alternativas: o uso de Dialogflow, IBM Watson, RASA Stack ou, por fim, o uso de frameworks individuais para PLN e ML (Figura 36).

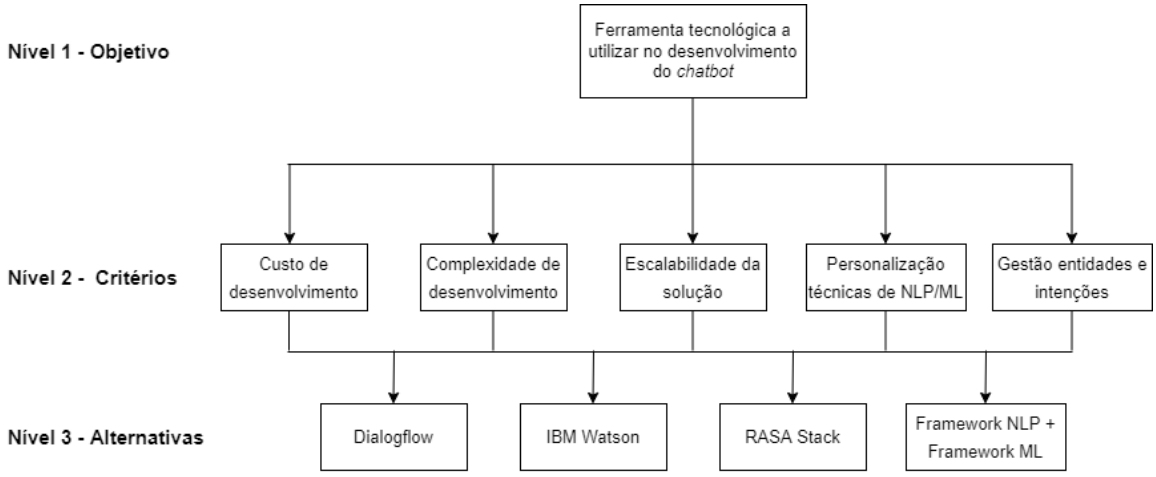

Figura 36. Modelo hierárquico para a escolha da ferramenta tecnológica

3.4.2 Definição de prioridade

O próximo passo é a definição de prioridades entre cada par de critérios. Para isto, será empregue o modelo *fundamental scale of absolute numbers* [77], definido por Saaty, que compara a importância de duas funcionalidades com valores entre 1 e 9 (Figura 37).

Nivel de importância	Definição	Explicação
1	Igual importânica	As duas atividades contribuem igualmente para o objetivo
3	Fraca importância	A experiência e o julgamento favorecem levemente uma atividade em relação à outra
5	Forte importância	A experiência e o julgamento favorecem fortemente uma atividade em relação à outra
7	Muito forte importância	Uma atividade é muito fortemente favorecida em relação a outra
9	Importância absoluta	A evidência favorece uma atividade em relação a outra com o mais alto grau de certeza
2,4,6,8	Valores intermediários	Quando se procura uma condição de compromisso entre duas definições

Figura 37. Escala de Saaty, retirado de [77]

O resultado final da comparação é a matriz denotada na Tabela 11. A análise de prioridades foi realizada através da discussão entre os intervenientes do projeto.

Critério	Custo	Complexidade	Escalabilidade	PLN/ML	Intenção/Entidade
Custo	1	1	0,33	0,20	0,33
Complexidade	1	1	0,5	0,20	0,33
Escalabilidade	3	2	1	0,25	0,5
PLN/ML	5	5	4	1	3
Intenção/Entidade	3	3	2	0,33	1

Tabela 11. Comparação de prioridades dos critérios definidos

Com o intuito de obter o peso relacionado com cada critério, é necessário proceder com a normalização dos valores da matriz anterior. Para isto, utiliza-se a divisão do valor de cada coluna com a sua soma total, como observado na Tabela 12.

Critério	Custo	Complexidade	Escalabilidade	PLN/ML	Intenção/Entidade
Custo	0,08	0,08	0,04	0,10	0,06
Complexidade	0,08	0,08	0,06	0,10	0,06
Escalabilidade	0,23	0,17	0,13	0,13	0,10
PLN/ML	0,38	0,42	0,51	0,51	0,58
Intenção/Entidade	0,23	0,25	0,26	0,16	0,20

Tabela 12. Normalização da matriz de prioridades

Para cada critério, obtém-se os seguintes pesos relativos, dos quais se destacam a capacidade de personalização da fase de *natural language processing* e dos métodos de *machine learning*:

1. Custo de desenvolvimento: 7%
2. Complexidade de desenvolvimento: 8%
3. Escalabilidade da solução: 15%
4. Personalização da *pipeline* de PLN e ML: 49%
5. Gestão de intenções e entidades: 22%

Antes de passar para a fase de atribuição de qualidade a cada alternativa, é necessário confirmar que os julgamentos dados foram consistentes. O rácio de consistência (RC) abrange a divisão do índice de consistência (IC) com o índice de consistência de matrizes aleatórias (IR), sendo que, caso o valor resultante seja abaixo de 0.1, os juízos são considerados válidos.

$$RC = \frac{IC}{IR}$$

O IC é obtido através de um cálculo intermédio da matriz de comparação inicial A, sem os valores normalizados, e o vetor de prioridades x, induzindo o valor de λmax.

$$A * x = \lambda\max * x$$

Utilizando a matriz da Tabela 11 como A e os valores dos pesos relativos, em formato decimal, como o vetor de prioridades x obtém-se o valor $\lambda\,\max = 5.077$.

$$\begin{bmatrix} 1 & 1 & 0{,}33 & 0{,}20 & 0{,}33 \\ 1 & 1 & 0{,}5 & 0{,}20 & 0{,}33 \\ 3 & 2 & 1 & 0{,}25 & 0{,}5 \\ 5 & 5 & 4 & 1 & 3 \\ 3 & 3 & 2 & 0{,}33 & 1 \end{bmatrix} * \begin{bmatrix} 0{,}07 \\ 0{,}08 \\ 0{,}15 \\ 0{,}49 \\ 0{,}22 \end{bmatrix} = \lambda\max * \begin{bmatrix} 0{,}07 \\ 0{,}08 \\ 0{,}15 \\ 0{,}49 \\ 0{,}22 \end{bmatrix}$$

Posteriormente, é possível calcular o valor de IC através do número de critérios existentes n e o valor λmax.

$$IC = \frac{\lambda\,\max - n}{n-1} = \frac{5.077 - 5}{5-1} \cong 0.019$$

Por fim, utilizando o valor $IR = 1.12$, associado a cinco critérios de estudo [78], obtém-se o cálculo final. Visto que o resultado obtido satisfaz a condição $RC < 0.1$, os juízos são validados.

$$RC = \frac{0.019}{1.12} \cong 0.017$$

A próxima etapa do AHP consiste no cálculo das prioridades de cada critério relativo às alternativas expostas. Para isto, utiliza-se o mesmo procedimento definido no início da secção, construindo matrizes de comparação para esclarecer qual a melhor hipótese para cada fator. Para efeitos de simplificação, os valores da matriz normalizada aparecem imediatamente na matriz inicial, entre parênteses, posteriormente admitidos para o cálculo do peso relativo final. A Tabela 13 representa as avaliações efetuadas para o custo de desenvolvimento, sendo que, naturalmente, as opções pagas ficam em desvantagem contra as gratuitas.

Custo de desenvolvimento	Dialogflow	IBM Watson	RASA Stack	*Framework* PLN/ML	Peso Relativo
Dialogflow	1 *(0,06)*	0,33 *(0,05)*	0,17 *(0,07)*	0,17 *(0,07)*	*6.2%*
IBM Watson	3 *(0,19)*	1 *(0,14)*	0,33 *(0,13)*	0,33 *(0,13)*	*14.6%*
RASA Stack	6 *(0,38)*	3 *(0,41)*	1 *(0,4)*	1 *(0,4)*	*39.6%*
Framework PLN/ML	6 *(0,38)*	3 *(0,41)*	1 *(0,4)*	1 *(0,4)*	*39.6%*

Tabela 13. Matriz de comparação para o custo de desenvolvimento

No que toca a complexidade de desenvolvimento, exposto na Tabela 14, IBM Watson e Dialogflow são priorizadas, devido à sua abstração técnica e ao suporte *web*.

Complexidade de desenvolvimento	Dialogflow	IBM Watson	RASA Stack	*Framework* PLN/ML	Peso Relativo
Dialogflow	1 *(0,40)*	1 *(0,40)*	3 *(0,41)*	5 *(0,36)*	*39.3%*
IBM Watson	1 *(0,40)*	1 *(0,40)*	3 *(0,41)*	5 *(0,36)*	*39.3%*
RASA Stack	0,33 *(0,13)*	0,33 *(0,13)*	1 *(0,14)*	3 *(0,21)*	*15.3%*
Framework PLN/ML	0,20 *(0,07)*	0,20 *(0,07)*	0,33 *(0,04)*	1 *(0,07)*	*6.1%*

Tabela 14. Matriz de comparação para a complexidade de desenvolvimento

O mesmo julgamento é empregue na escalabilidade, visto que estas duas *frameworks* abstraem por completo a configuração técnica da solução. Desta forma, permitem uma escalabilidade muito facilitada, com integração a vários serviços exteriores, como analisado na Tabela 15.

Escalabilidade da solução	Dialogflow	IBM Watson	RASA Stack	*Framework* PLN/ML	Peso Relativo
Dialogflow	1 *(0,29)*	0,5 *(0,28)*	3 *(0,36)*	6 *(0,35)*	*32.0%*
IBM Watson	2 *(0,57)*	1 *(0,55)*	4 *(0,48)*	7 *(0,41)*	*50.2%*
RASA Stack	0,33 *(0,09)*	0,25 *(0,14)*	1 *(0,12)*	3 *(0,18)*	*13.3%*
Framework PLN	0,17 *(0,05)*	0,06 *(0,03)*	0,33 *(0,04)*	1 *(0,06)*	*4.5%*

Tabela 15. Matriz de comparação para a escalabilidade da solução

Em contrapartida, as últimas duas alternativas prosperam no critério de personalização do módulo de *natural language processing* e de *machine learning,* deixando quais as técnicas a utilizar ao critério do desenvolvedor (Tabela 16).

À exceção da modificação dos dados de treino, IBM Watson e Dialogflow têm o módulo de inteligência artificial definido internamente e não possibilitam a sua personalização.

Personalização de PLN/ML	Dialogflow	IBM Watson	RASA Stack	*Framework* PLN/ML	Peso Relativo
Dialogflow	1 *(0,10)*	1 *(0,10)*	0,25 *(0,10)*	0,25 *(0,10)*	10.0%
IBM Watson	1 *(0,10)*	1 *(0,10)*	0,25 *(0,10)*	0,25 *(0,10)*	10.0%
RASA Stack	4 *(0,40)*	4 *(0,40)*	1 *(0,4)*	1 *(0,40)*	40.0%
Framework PLN/ML	4 *(0,40)*	4 *(0,40)*	1 *(0,4)*	1 *(0,40)*	40.0%

Tabela 16. Matriz de comparação para a personalização de PLN/ML

A gestão de entidades e intenções é desenvolvida de forma semelhante para as primeiras três alternativas, sendo que, por outro lado, as *frameworks* individuais de PLN e ML não fornecem qualquer suporte.

Gestão Entidade/ Intenção	Dialogflow	IBM Watson	RASA Stack	*Framework* PLN/ML	Peso Relativo
Dialogflow	1 *(0,38)*	1 *(0,38)*	2 *(0,39)*	7 *(0,33)*	37.0%
IBM Watson	1 *(0,38)*	1 *(0,38)*	2 *(0,39)*	7 *(0,33)*	37.0%
RASA Stack	0,5 *(0,19)*	0,5 *(0,19)*	1 *(0,19)*	6 *(0,28)*	22.0%
Framework PLN/ML	0,14 *(0,05)*	0,14 *(0,05)*	0,17 *(0,03)*	1 *(0,06)*	4.0%

Tabela 17. Matriz de comparação para a gestão de entidades e intenções

3.4.3 Decisão final

Com a caracterização de cada alternativa, multiplicam-se os seus pesos relativos com o valor de preferência dos critérios, obtendo o vetor de prioridade final. Cada linha da matriz está associada às cargas obtidas por cada ferramenta, sendo que o peso relativo final maior será a ferramenta escolhida.

$$\begin{bmatrix} 0.06 & 0,39 & 0,32 & 0,10 & 0,37 \\ 0,14 & 0,39 & 0,50 & 0,10 & 0,37 \\ 0,39 & 0,15 & 0,13 & 0,40 & 0,22 \\ 0,39 & 0,06 & 0,04 & 0,40 & 0,04 \end{bmatrix} * \begin{bmatrix} 0,07 \\ 0,08 \\ 0,15 \\ 0,49 \\ 0,22 \end{bmatrix} = \begin{bmatrix} 0.21 \\ 0.25 \\ 0.30 \\ 0.24 \end{bmatrix}$$

O resultado final rotula a utilização de RASA Stack como a mais acertada, com um peso relativo de 30%. Em retrospetiva, é a decisão que faz mais sentido, visto que equilibra uma boa escalabilidade de solução com uma vertente de personalização suficiente para fazer a experimentação de modelos.

4 Análise e desenho do sistema

Esta secção apresenta o sistema através de uma perspetiva mais detalhada, expondo todas as contribuições anteriores à implementação da solução. Tendo em conta o problema apresentado, denota-se o processo de engenharia de requisitos, particularmente os conceitos de negócio principais e a análise dos requisitos funcionais e não funcionais. Para além disso, utilizando o modelo C4+1, enunciam-se os artefactos representantes da arquitetura da aplicação.

4.1 Engenharia de Requisitos

Segundo Easterbrook [79], a engenharia de requisitos é o conjunto de atividades responsável por identificar e comunicar o propósito de um sistema de *software*. Esta disciplina age como um intermédio entre os requisitos empregues pelos *stakeholders* e as funcionalidades da solução, analisando se a perspetiva de negócio está alinhada com a vertente técnica. É um processo fulcral dado que origina documentação inequívoca que possa ser consultada ao longo do desenvolvimento e manutenção da solução.

4.1.1 Stakeholders e Atores

Um *stakeholder* é uma entidade que é diretamente ou indiretamente afetada pelo produto de *software* [79]. Para compreender quais as necessidades que a solução vai abranger, é importante integrar todos os intervenientes no seu processo de desenvolvimento:

- **Colaboradores do ISEP**: responsáveis pela gestão da informação do agente;
- **Estudantes**: os utilizadores finais do *chatbot*, sendo que se incluem os estudantes universitários, de secundário, ou qualquer outro com interesse na Escola;

- **ISEP**: a organização é a maior influenciada do projeto, sendo que terá a adição de um serviço de resposta automática para a sua Divisão Académica.

Por outro lado, um ator representa uma entidade que interage diretamente com o sistema, normalmente definidos através do cargo que representam:

- **Configurador:** Encarregue de adicionar e manter a base de informação que o agente utiliza para moldar o seu comportamento. Isto é efetuado através da criação e edição de tópicos informativos através de pares pergunta-resposta;
- **Emissor:** A entidade que comunica diretamente com o agente conversacional, procurando respostas para as suas perguntas.

As tarefas de configuração estão associadas aos colaboradores da universidade, particularmente da Divisão Académica, enquanto os emissores são os estudantes.

4.1.2 Análise de Requisitos

O processo de elicitação de requisitos foi realizado através de reuniões com os *stakeholders*, complementados com a pesquisa realizada no estado da arte. A exposição dos requisitos será segmentada em duas partes, nomeadamente os requisitos funcionais, através de casos de uso, e não funcionais, através do modelo FURPS+.

4.1.2.1 Requisitos funcionais

Um caso de uso descreve a interação entre uma entidade e o sistema de *software*, sendo que é composto por três módulos, o ator, o sistema e, por fim, o objetivo da comunicação [80]. No caso da solução proposta, foram denotados oito casos, agrupados por ator na Figura 38.

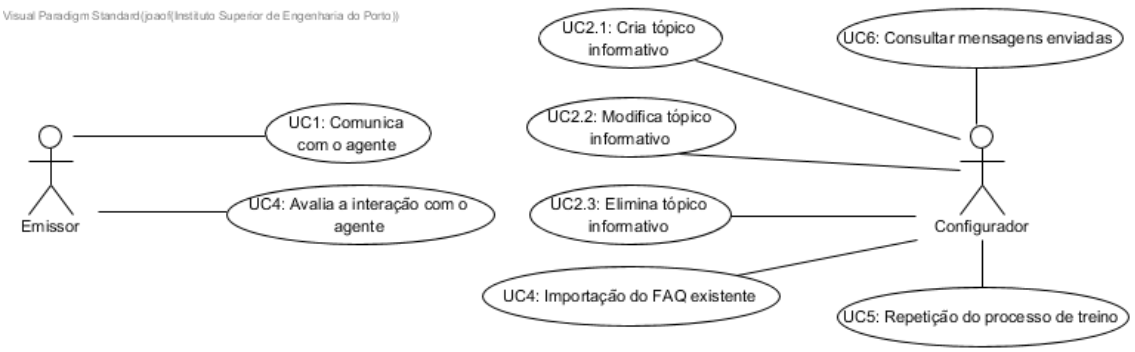

Figura 38. Casos de uso numerados e agrupados por ator

1. **Comunicação com o agente**

O emissor procede ao envio de uma mensagem ao *chatbot*, recebendo uma resposta instantânea. O sistema tem a capacidade de interpretar a mensagem através de linguagem natural, respondendo consoante o contexto do discurso. O emissor tem a possibilidade de continuar a conversa com o agente, mas a sua interação é baseada no esquema pergunta-resposta, tal como denotado na Figura 39. Em caso de falha de compreensão, o agente reencaminhará o utilizador para a Divisão Académica através de uma mensagem base.

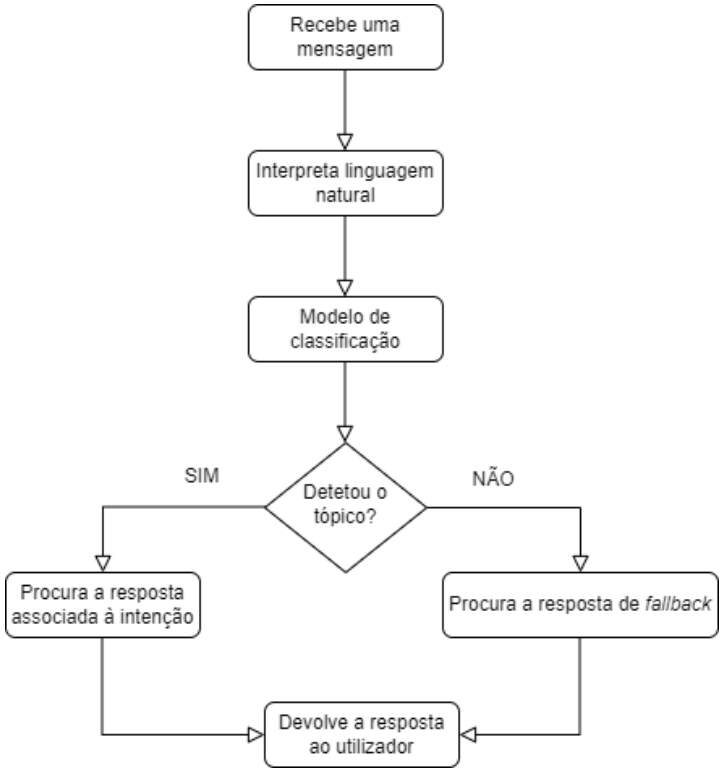

Figura 39. Fluxograma do esquema de comunicação do agente

A comunicação é realizada através da página da Divisão Académica, onde será implantado uma plataforma de conversa. Não existe qualquer tipo de autenticação nem autorização, sendo que não é necessário recolher a identidade do estudante para as respostas.

2. Gestão de tópicos de resposta

O configurador procede à configuração dos tópicos utilizados para a moldagem do comportamento do agente, especificamente as ações de criação (2.1), edição (2.2) e eliminação (2.3). O tópico é um conceito composto pela sua descrição, o identificador textual, uma ou mais respostas associadas e, para objetivos de visibilidade, a categoria a que pertence. Em termos da criação da entidade, as únicas restrições são o identificador textual, que tem de ser único, e a existência de uma categoria. A edição e eliminação será efetuada através do identificador, geridas pelos configuradores através de uma página privada.

3. Importação do FAQ existente

O sistema disponibiliza um serviço que permite aos configuradores importar a página de perguntas frequentes já existentes na Divisão Académica. Este caso de uso é suportado através das funcionalidades denotadas no requisito 2, iteradas por cada tópico introduzido. A informação será introduzida através de um ficheiro CSV.

4. Processo de *feedback*

Após cada interação, o emissor tem a opção de qualificar a resposta do agente, fornecendo um valor entre 0 e 5. O sistema armazena esta avaliação com a questão feita, a intenção interpretada e o valor emitido pelo utilizador.

5. **Treinar o modelo de classificação**

Em caso de adição de nova informação ao conjunto de tópicos, o configurador pode provocar o processo de reaprendizagem. Apesar de não substituir o modelo em execução, a ação cria um novo modelo de classificação que pode ser utilizado posteriormente.

6. **Consultar as mensagens enviadas**

Derivado do requisito não funcional de auditoria de mensagens, demonstrado na secção 4.1.2.2, o configurador tem a possibilidade de verificar as perguntas efetuadas, quais as intenções associadas e o respetivo fator de confiança. É útil para melhorar o comportamento do agente a longo prazo, sendo que permite avaliar a reação do agente relativamente aos pedidos dos estudantes. O emissor não é identificado no processo de conversação.

4.1.2.2 Requisitos não funcionais

Um requisito não funcional consiste em uma condição de funcionamento do sistema, normalmente concebido através de um modelo global. O FURPS+ é um acrónimo que serve de orientação para a anotação destes requisitos, fragmentados em cinco módulos, a funcionalidade, a usabilidade, a confiabilidade, a *performance* e, por fim, o suporte. O '+' simboliza restrições adicionais de desenho e implementação.

A **funcionalidade** refere-se aos requisitos funcionais que não estão diretamente associados com os casos de uso:

1. Armazenamento dos momentos de criação e modificação dos tópicos;
2. Auditoria de todas as respostas associadas a um tópico, de forma a poder adicionar informação ao conjunto de dados;
3. Possibilidade de treinar o agente sobre qualquer conjunto de dados;

Performance, ou desempenho, tem como objetivo medir as condições relativas à gestão de recursos do sistema:

1. O tempo de resposta das mensagens deverá ser o mais baixo possível (abaixo de 500ms).

A **suportabilidade** denota os aspetos relativos à adequabilidade do sistema, nomeadamente os idiomas, escalabilidade e configuração:

1. A linguagem da conversa é o português;
2. O sistema deverá ser suportado em qualquer *browser*;
3. O sistema deverá permitir o treino do agente sem interromper a sua execução.

As **restrições de implementação** consistem em condições aplicadas às escolhas das tecnologias:

1. A escolha de tecnologia inclui a revisão dos métodos de aprendizagem profunda e a decisão de qual a *framework* mais indicada.

O artefacto *House of Quality*, apresentado na secção 3.3, demonstra a relação existente entre os requisitos impostos e as características de engenharia da solução. Tal como referido, a comunicação com o agente e a manutenção dos tópicos informativos são os requisitos fulcrais para o produto mínimo viável, sendo que estabelecem a habilidade básica para a comunicação com os emissores. O processo de avaliação e a habilidade de reconfigurar o agente são

melhorias que apoiam o comportamento do *chatbot* ao longo do tempo, possibilitando melhores condições para o processo de escolha de resposta.

4.1.3 Modelação de Negócio

A modelação de negócio representa uma fase fundamental no processo de planeamento de *software*, prestando uma visão global do sistema para todos os envolvidos. Com o intuito de analisar os conceitos da solução, recorreu-se à construção de um modelo de domínio.

Evans [81] define o modelo de domínio como um artefacto que expõe a organização estruturada do conhecimento de negócio, especificado através de vocabulário adequado, com o foco em expor os conceitos principais e as suas relações. Dado que esta modelação funciona à base do desenho de objetos, é um mediador importante entre a análise de requisitos e a implementação das classes de *software*.

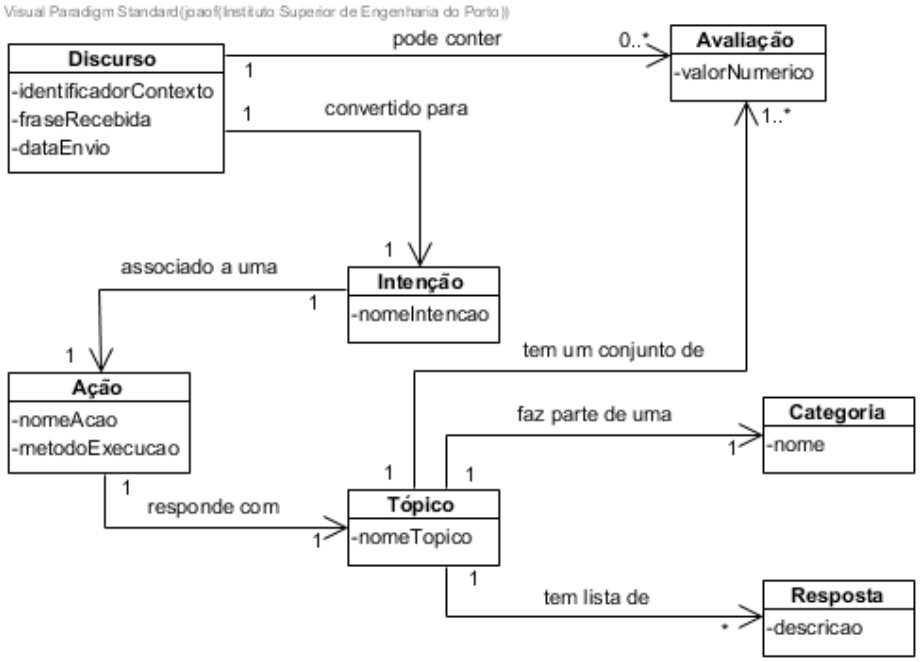

Figura 40. Modelo de domínio da solução

A Figura 40 expõe o modelo de domínio denotado para o ecossistema do *chatbot*:

1. O **discurso** é o conceito que está associado às mensagens enviadas pelo emissor, constituído pelo texto enviado, um identificador único da conversa, necessário para distinguir qual o contexto da sessão, e, por fim, a data de envio. Para além disso, caso se aplique, a avaliação relativa à resposta obtida posteriormente.
2. A **intenção**, referida na secção 2.3.1.1, constitui o identificador dos possíveis fluxos de interpretação do agente;
3. A **ação** representa o comportamento a tomar após o reconhecimento de uma intenção, sendo que é construída com um nome e um método de execução;

4. O **tópico** é o esquema pergunta-resposta que contém informação relativa ao domínio do agente, sendo possível a existência de mais do que uma **resposta** para cada pergunta. Para além de estar associado a uma **categoria**, para agregar intenções através de temas de alto-nível, ainda tem um conjunto de **avaliações**.

O modelo representa o fluxo da Figura 39 através de uma perspetiva orientada a conceitos, podendo ser interpretado de forma sequencial. O discurso é convertido para um conjunto pré-definido de intenções, que, posteriormente, estão associadas ao comportamento do agente, as ações. Estas contêm a lógica que definem como o agente vai construir a resposta, sendo que, no caso do sistema presente, será a recolha do tópico mais adequado e a respetiva resposta.

4.2 Desenho

A presente secção evidencia o processo de desenho percorrido, expondo as decisões que justificam a arquitetura escolhida tal como possíveis alternativas.

4.2.1 Arquitetura do sistema

A arquitetura define-se como a organização elementar de uma aplicação, constituída pelos seus componentes e a respetiva forma de comunicação. O processo de desenho arquitetural é fundamental no planeamento inicial dado que facilita não só o desenvolvimento de novas funcionalidades, como o seu processo de manutenção. A longo prazo, mudanças a nível arquitetural são mais dispendiosas que a nível de implementação [82].

Para a exposição das alternativas arquiteturais estudadas, estruturaram-se artefactos através do modelo C4+1. Este consiste num conjunto sequencial de diagramas relevantes no seu contexto, inicialmente a partir de um ponto de vista geral, definindo o sistema como uma caixa opaca, para níveis mais detalhados, especificamente a delineação dos componentes e, por fim, das classes de *software* [83]. Dado que o último nível se trata de componentes de implementação, apenas é exposto no capítulo 5.

De forma complementar, o modelo C4+1 especifica que, para cada nível, é necessário delinear vários tipos de vistas:

- A **vista lógica** tem como objetivo descrever os elementos de *software* e as suas fronteiras. É usualmente descrito através de diagramas de classes ou componentes;
- A **vista de processo** demonstra a interação entre os processos do sistema, representada através de um diagrama de sequência;
- A **vista física** representa o esquema de implantação de cada componente.

4.2.1.1 Nível 1: O sistema

O diagrama de nível 1 denota o ponto de partida do sistema, demonstrando quais os atores e sistemas exteriores que interagem com a aplicação. Como observado pela Figura 41, o emissor e o configurador comunicam com a solução, sendo que esta é demonstrada através da interface visual do ISEP, o portal.

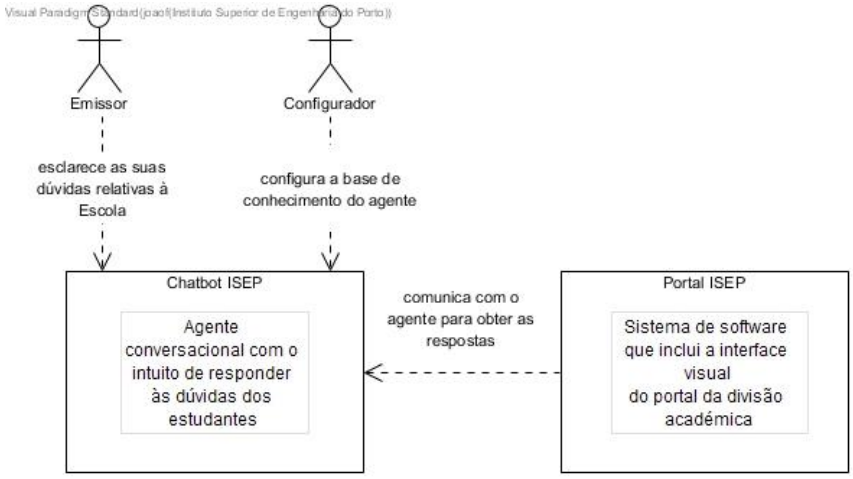

Figura 41. Diagrama lógico de nível 1 para o ecossistema do *chatbot*

4.2.1.2 Nível 2: Contentores

O nível 2 tem como intuito aumentar o âmbito de observação do sistema, expondo os módulos, definidos como uma unidade de execução, e os respetivos protocolos de comunicação. Como exposto na Figura 42, o sistema é composto por dois componentes, nomeadamente o núcleo do agente, Chatbot Core, e o serviço *web* de gestão de tópicos, Topic WebService.

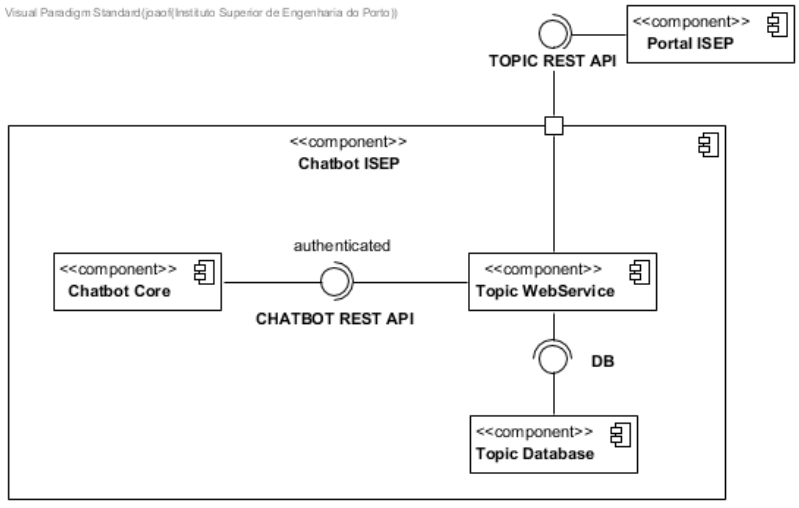

Figura 42. Diagrama lógico de nível 2 para o ecossistema do *chatbot*

Chatbot Core constitui a lógica funcional para o PLN, desenvolvido em RASA, disponibilizando uma interface de comunicação com o exterior através de uma API REST. No entanto, o componente que recebe o pedido do utilizador é o serviço de tópicos, através do portal. Assim que receba uma mensagem, este último contacta o Chatbot Core para identificar qual a intenção interpretada, obtendo um conjunto de tópicos e a respetiva probabilidade de acerto. Caso o nível de confiança seja alto, o serviço devolve a resposta ao utilizador. Este processo é exposto concetualmente na Figura 43, excluindo, para efeitos de simplificação, a devolução de resposta ao utilizador. É importante referir que a comunicação entre os componentes é realizada com recurso a autenticação com *tokens*.

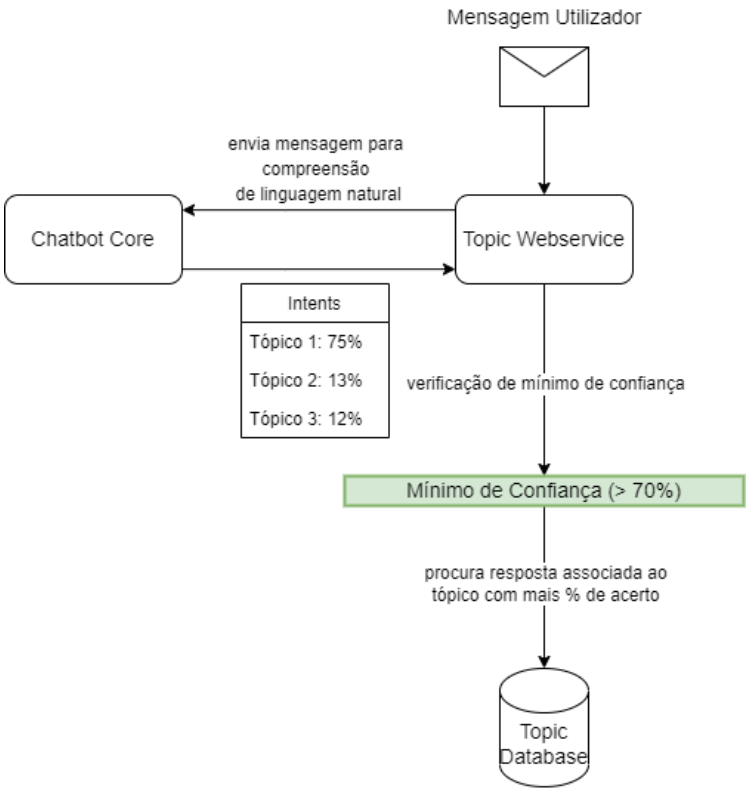

Figura 43. Integração entre os dois componentes delineados para a interação com o utilizador

Este esquema, para além de promover a modularização, ao desacoplar a implementação do agente conversacional com a plataforma de comunicação, ainda permite ter o controlo total sobre o processo de gestão de informação. Para além disso, é especificamente útil para funcionalidades como a auditoria, o rastreio do contexto da conversa e a avaliação de interações.

Alternativa: Base de tópicos interna ao agente conversacional

O primeiro esquema estudado foi a omissão do serviço de tópicos, utilizando as funcionalidades internas do RASA para um conjunto estático de respostas.

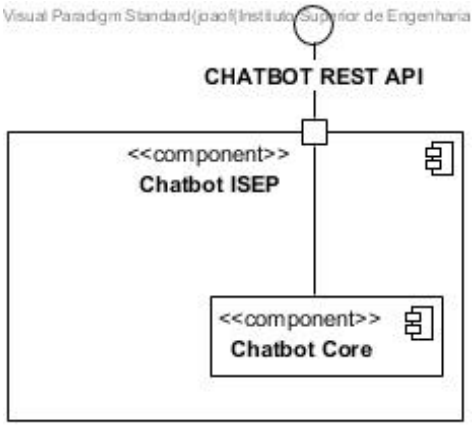

Figura 44. Diagrama lógico de nível 2 para o uso de uma base de tópicos interna ao agente

Esta solução apresenta um nível de desempenho bastante elevado, visto que descarta por completo o pedido *web* sempre que é necessário a geração de resposta. Em contrapartida, não possibilita a implementação da avaliação de interações, restringe o processo de seleção de respostas à lógica do RASA e descarta qualquer processo de auditoria, referidos na secção de Requisitos não funcionais. Não é viável para uma aplicação escalável, nem para a obtenção de um nível de abstração tecnológica para os configuradores da Divisão Académica.

Alternativa: Chatbot Core como ponto de entrada

Outra alternativa analisada foi a utilização do núcleo do *chatbot* como ponto de comunicação direta com o portal, utilizando o serviço de tópicos apenas como armazenamento de informação. Ainda que semelhante à escolhida, esta arquitetura sacrifica dois critérios importantes, particularmente a segurança e a escalabilidade da solução, em troca da diminuição de esforço de desenvolvimento (Figura 45).

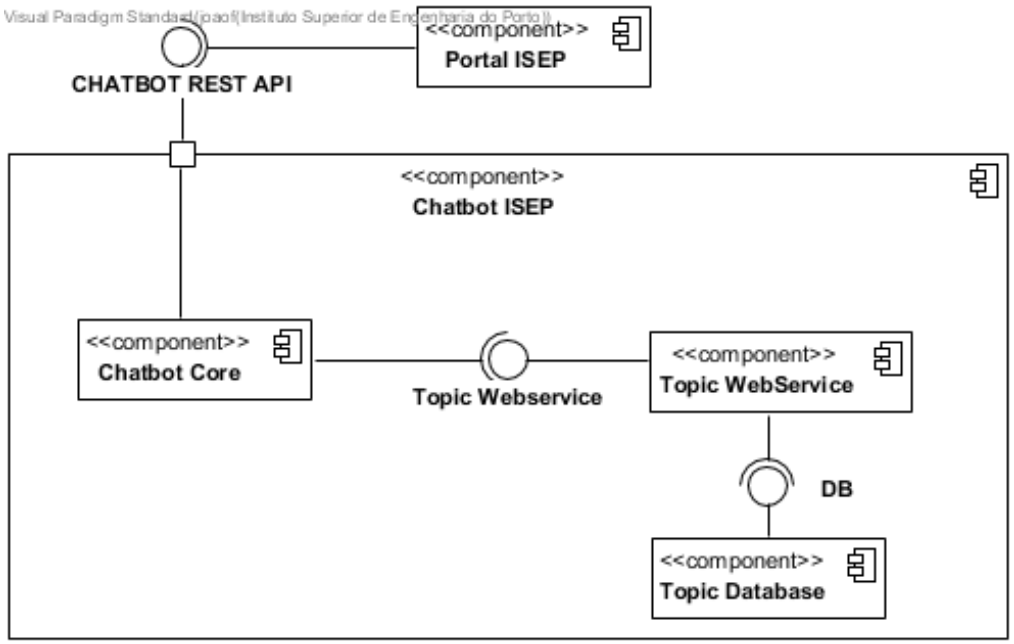

Figura 45. Diagrama lógico de nível 2 para a omissão do serviço de tópicos

A interpretação da mensagem é realizada de imediato no *Chatbot Core,* que comunica com o serviço de tópicos para obter a resposta associada à intenção. A arquitetura, para além de restringir a comunicação do portal à especificação tecnológica utilizada pelo RASA, atribui a gestão do processamento *web* à *framework*. Embora diminua o tempo de desenvolvimento, visto que esta camada é gerida automaticamente, a solução é pouco escalável em termos de funcionalidades, excluindo a construção de mecanismos de autenticação e auditoria.

Em termos de implantação, a solução é dividida de forma igual em dois componentes. Chatbot Core é publicado através de um servidor de aplicações RASA, proveniente da *framework*. O serviço de tópicos é executador com um servidor de aplicação JAVA, comunicando com o núcleo do agente através de HTTPS (Figura 46).

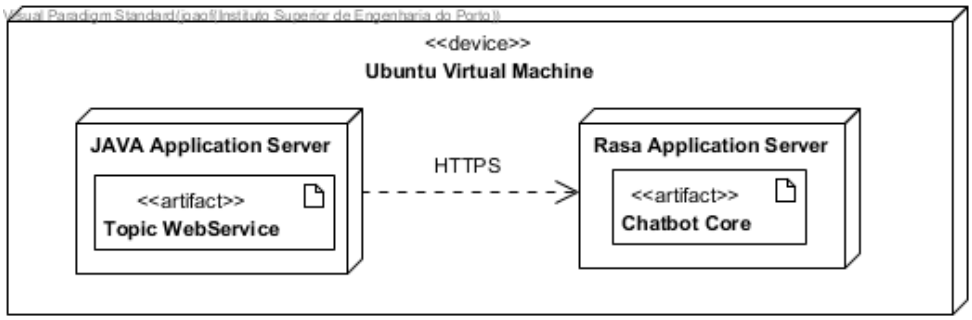

Figura 46. Diagrama físico de nível 2 para a implantação dos contentores

4.2.1.3 Nível 3: Componentes

O nível 3 emprega uma visão mais técnica dos contentores, apresentando os seus componentes internos. No caso da solução atual, expõe-se a arquitetura do Chatbot Core*, com a* Figura 47*,* e do Topic Web Service, com a Figura 49.

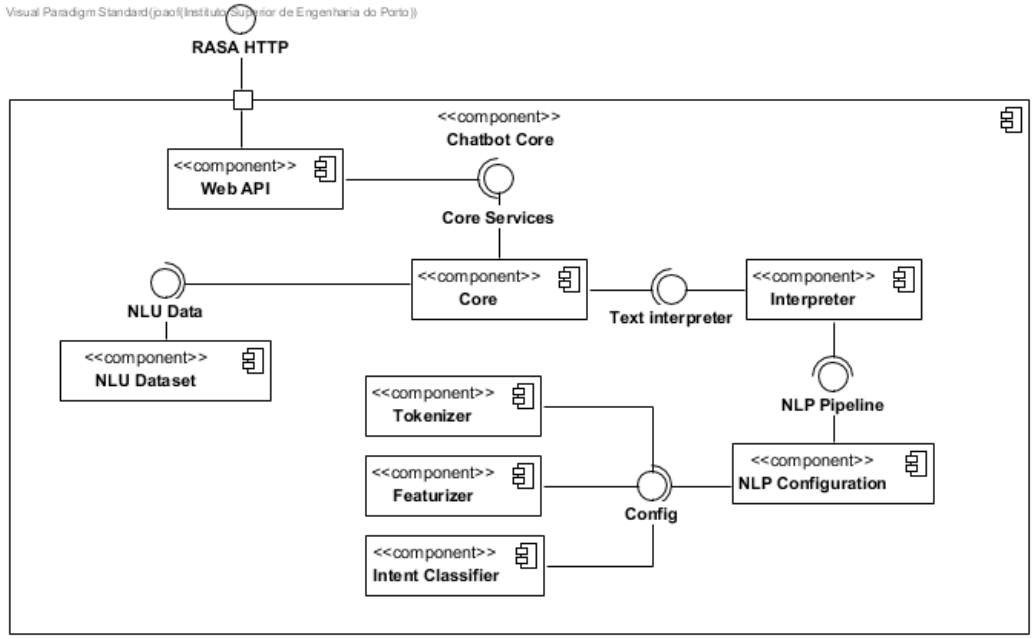

Figura 47. Arquitetura interna do núcleo do *chatbot*

A constituição interna do agente segue o esquema funcional do RASA demonstrado pela Figura 30, denotado através de um nível de granularidade maior. Contudo, sendo que a lógica de seleção de respostas reside no serviço de tópicos, apenas serão utilizados componentes relativos à CLN. *NLU Dataset* refere-se aos dados de treino.

O interpretador de texto funciona através da configuração de métodos de PLN, indicados manualmente pelo desenvolvedor. Cada esquema é divido em três componentes, nomeadamente *tokenizers*, que representam a etapa de pré-processamento de dados, *featurizers,* funções responsáveis por preparar os *tokens* antes da fase de aprendizagem e, por fim, *intent classifiers,* os algoritmos de aprendizagem profunda. A fase de treino é demonstrada através da Figura 48, tendo em conta os componentes descritos.

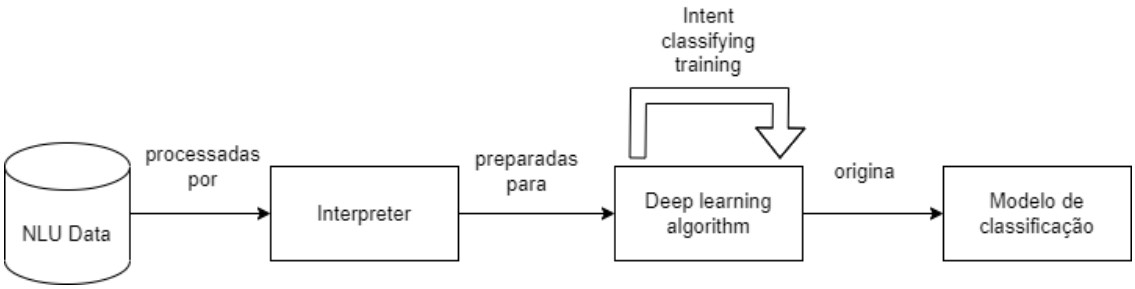

Figura 48. Ilustração da fase de treino do agente implementado em RASA

Para conseguir utilizar as ações do agente através de uma interface *web*, RASA disponibiliza um módulo HTTP, a *Web API*. Tem cinco subconjuntos de pedidos, nomeadamente autenticação, controlo de saúde do servidor, gestor de contexto, modelo e, por último, o domínio de informação. Para a solução proposta, emprega-se o uso de dois pontos de entrada, o de comunicação e de treino, demonstrados através da Tabela 18.

Endpoint	**Descrição**	**Requisito**
/model/parse	Introduz uma mensagem no modelo de classificação, retornando a lista de intenções e probabilidades de acerto.	REQ-1
/model/train	Provoca o processo de treino do agente, utilizando a informação presente na aplicação ou uma configuração enviada no pedido.	REQ-5

Tabela 18. Funções disponibilizadas pela interface *web* de *RASA*

O serviço de tópicos segue uma arquitetura por camadas [84], onde cada uma comunica diretamente com o nível anterior. De forma a simplificar a aplicação, mas manter um nível de escalabilidade positivo, definiram-se quatro camadas, nomeadamente os controladores, os serviços, o nível de domínio e, por fim, o repositório.

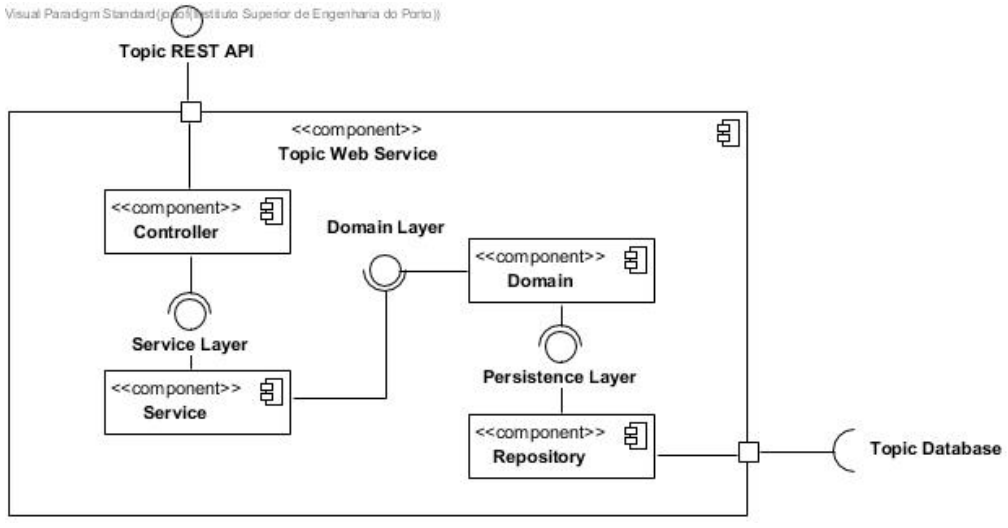

Figura 49. Arquitetura interna do serviço *web* de tópicos

O controlador serve como ponto de entrada da aplicação, mapeando os pedidos do utilizador com as ações a tomar. O serviço é responsável por orientar os objetos de domínio, sendo que estes contêm toda a lógica de negócio. A camada de persistência denota o desacoplamento entre o modelo interno de domínio com a implementação da base de dados. Apesar de omitido, o serviço utiliza objetos especiais para abstrair a camada interna de negócio do exterior, utilizando *data transfer objects* (DTO) e mapeadores. Visto que não existe qualquer restrição tecnológica, o serviço será desenvolvido em JAVA, utilizando a *framework* Spring Boot.

4.2.2 Casos de Uso

Para além da recolha de requisitos funcionais, realizada na secção 4.1.2, efetuou-se a análise detalhada dos casos de uso. A ação inicial entre os atores e os sistemas é demonstrada através da especificação do pedido *web*.

4.2.2.1 REQ-1: Comunicação com o agente

Esta funcionalidade inclui ambos os módulos, sendo que fornece uma visão da comunicação entre si. É importante referir que a interação assume que o treino do agente já foi realizado e o modelo de classificação está definido. Para efeitos de simplificação, o requisito foi dividido entre dois diagramas de sequência, conectados pelo pedido realizado ao núcleo do agente.

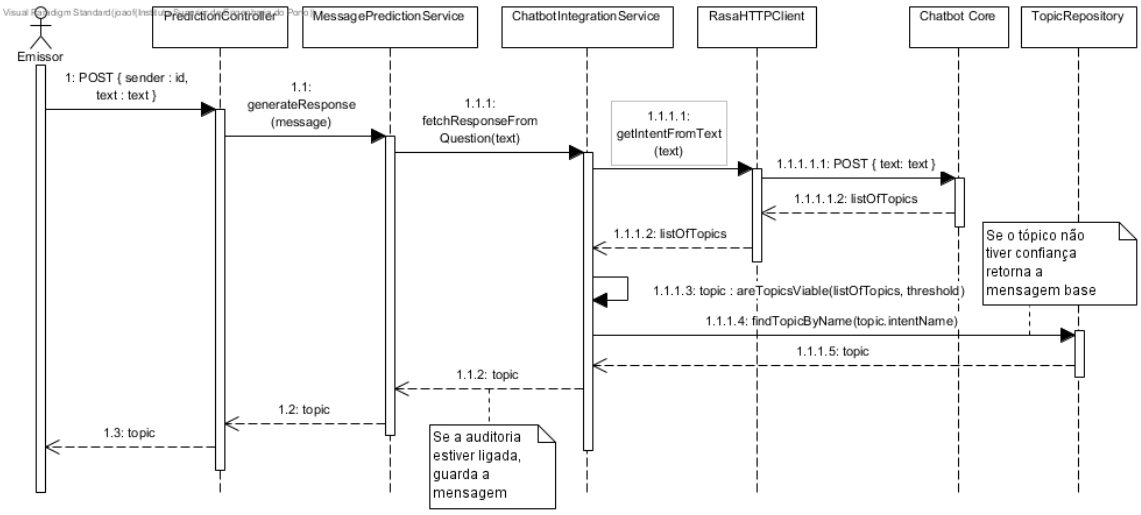

Figura 50. Diagrama de sequência para a comunicação com o agente

O nível do controlador é responsável por receber o pedido do utilizador, orientando a ação de processamento de texto através de serviços. *ChatbotIntegrationService* é a camada de abstração entre o módulo *web* e o agente, sendo que comunicará com componentes externos através de um cliente HTTP (Figura 50 e Figura 51). Ainda que não seja expectável, este mecanismo permite a extensibilidade em relação à tecnologia utilizada para a CLN.

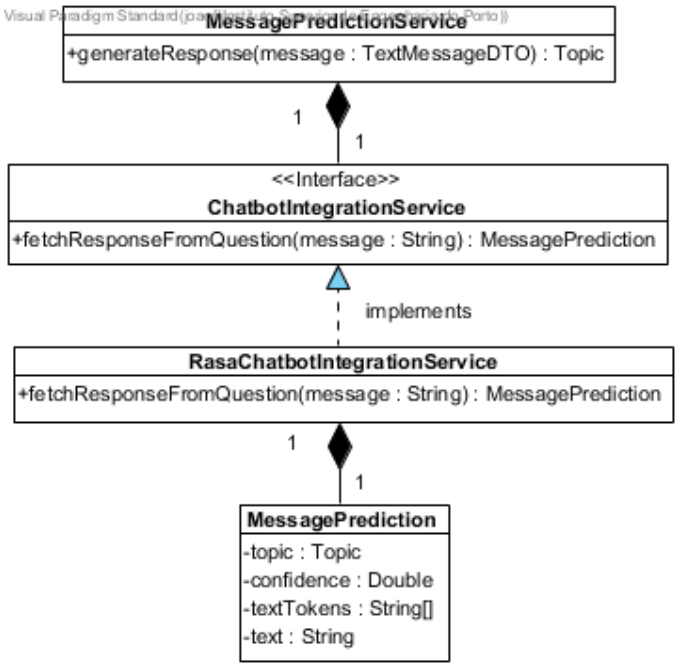

Figura 51. Mecanismo de abstração entre o serviço de tópicos e a implementação de *chatbots*

Para o caso atual, envia-se uma mensagem para o ponto de saída do servidor RASA, referido na Tabela 18, recebendo uma lista de intenções e o respetivo valor de confiança. Caso a intenção favorita tenha probabilidade de acerto superior ao limite definido, procura-se o tópico associado e, posteriormente, retorna-se a resposta ao utilizador. O funcionamento interno do agente é delineado na Figura 52. Tendo em conta a configuração de PLN, o texto recebido pelo emissor é transformado para uma estrutura numérica e, de seguida, conectado a uma intenção.

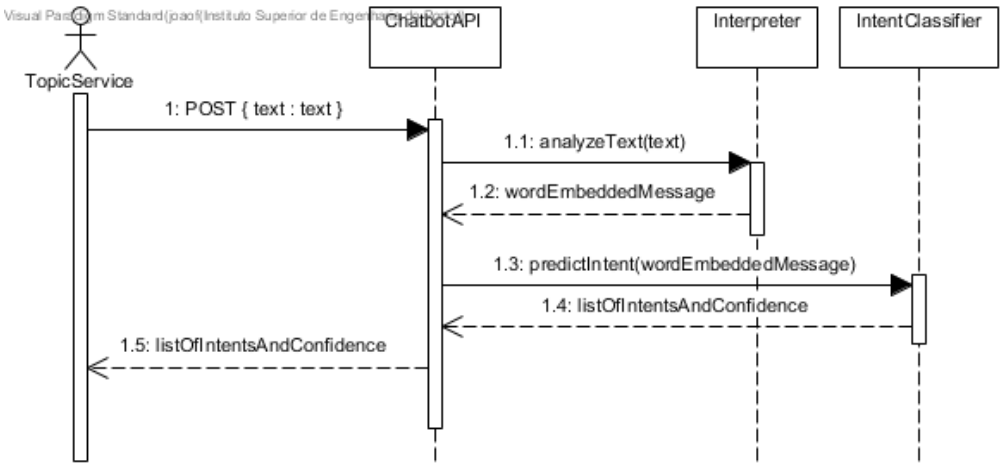

Figura 52. Diagrama de sequência para o PLN

Por fim, para dar resposta ao requisito não funcional de auditoria de mensagens, o servidor armazena todas as questões e as intenções recolhidas. Esta funcionalidade, extraída do fluxo da Figura 50 para melhor visibilidade, é realizada após a recolha da resposta derivada do núcleo do agente (Figura 53).

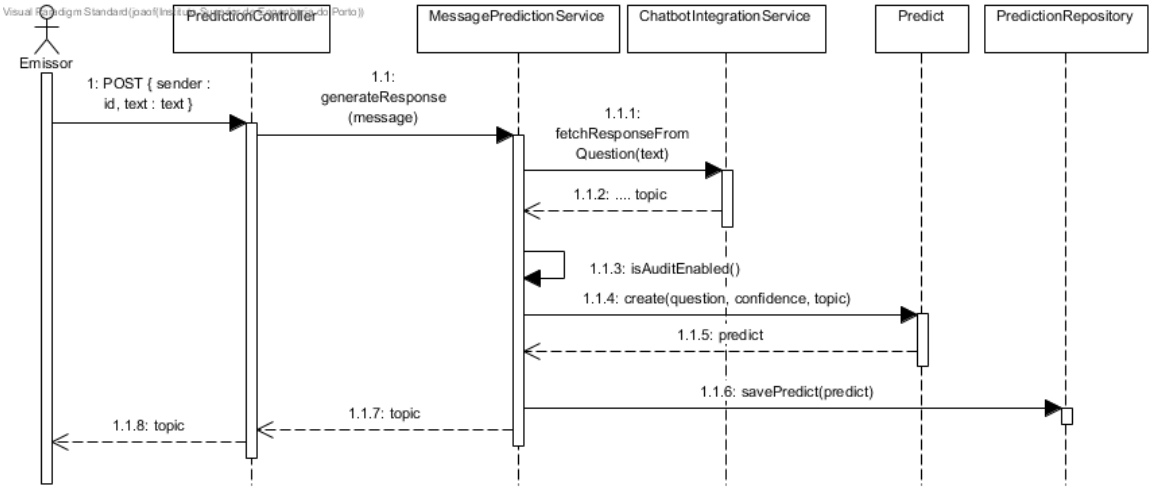

Figura 53. Diagrama de sequência para o processo de auditoria de mensagens

Para cada interação, persiste-se a pergunta realizada, o tópico, a confiança atribuída pelo agente e, caso efetuada, a avaliação do utilizador. São os parâmetros necessários para conseguir analisar se a mensagem pode ser relevante para o novo conjunto de dados.

4.2.2.2 REQ-4: Avaliação da interação

A avaliação da interação é realizada após a receção de uma resposta, sendo que é iniciado após a execução do REQ-1 (Figura 54).

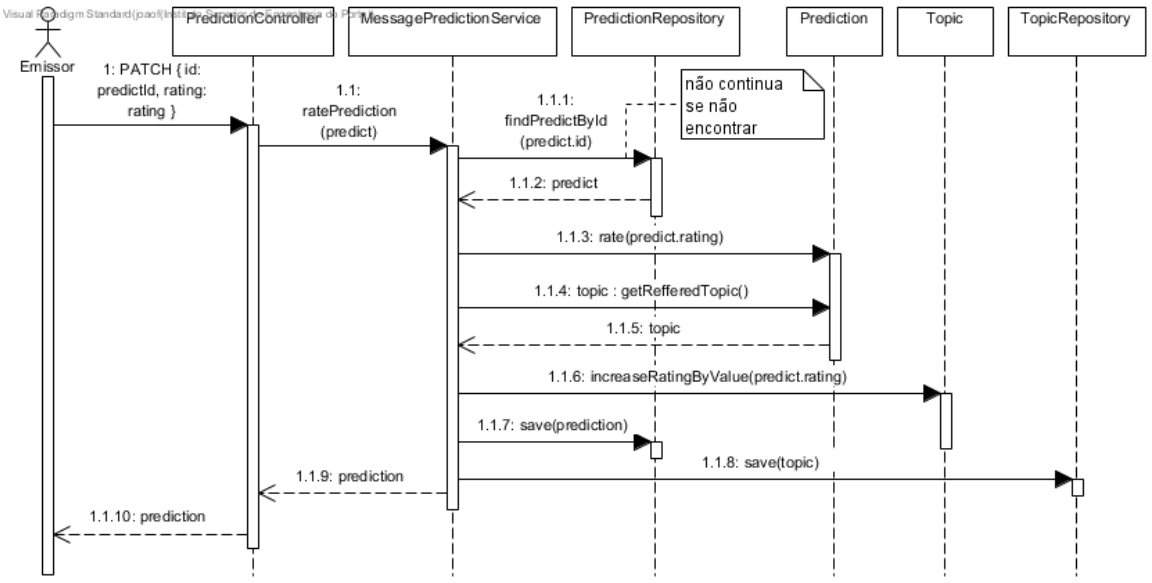

Figura 54. Diagrama de sequência para a avaliação de respostas

A resposta enviada para o utilizador terá um identificador associado para que seja possível associar uma avaliação. A sua persistência é realizada em dois momentos, nomeadamente na previsão, atomicamente, e também no tópico interpretado, de forma a somar o número de avaliações e a média.

4.2.2.3 REQ-2.1: Criação do Tópico e REQ-2.2: Modificação do Tópico

A criação do tópico é realizada através de um formulário, onde o configurador introduz o nome do tópico e a(s) resposta(s) associada(s) (Figura 55).

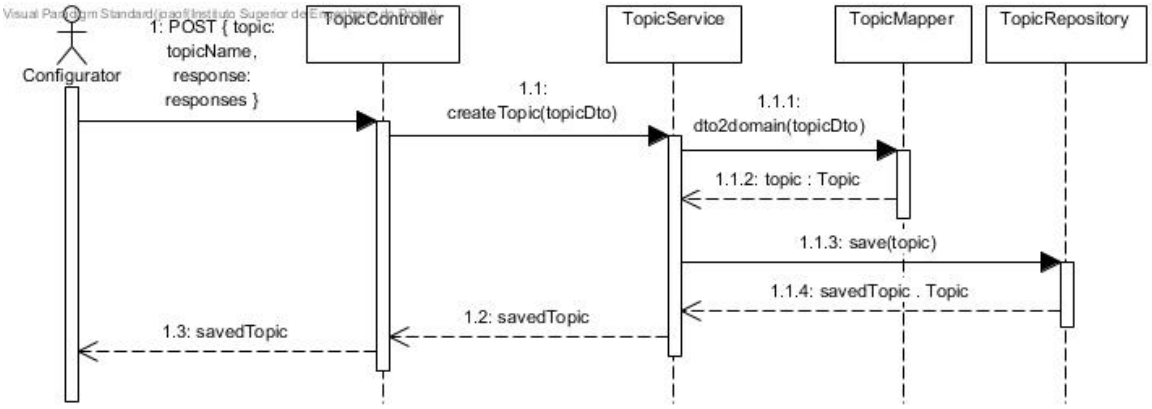

Figura 55. Diagrama de sequência para a criação de um tópico

O caso de uso segue o fluxo denotado pela Figura 49, onde o controlador recebe o tópico inicial através de um DTO e, através da camada de serviço, o converte para um objeto de domínio. A modificação de um tópico é realizada de forma semelhante, dado que a única diferença é o tópico vir com um identificador já definido. Desta forma, o método *save* serve como um ponto de atualização. O tópico é internamente caracterizado na Figura 56.

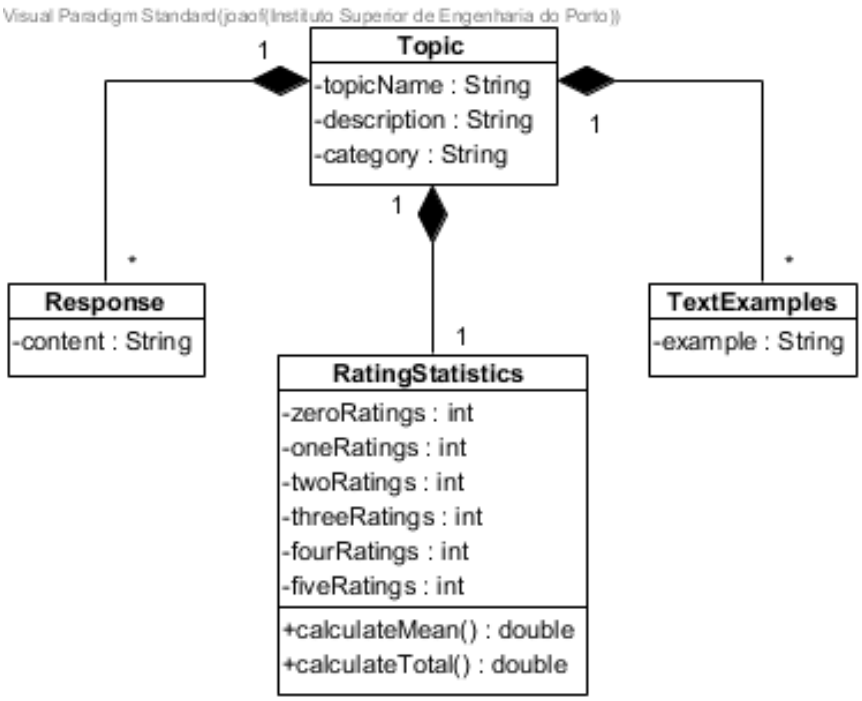

Figura 56. Especificação interna de um tópico

4.2.2.4 REQ-2.3: Eliminação de um tópico

A eliminação do tópico é processada através do seu identificador único (Figura 57).

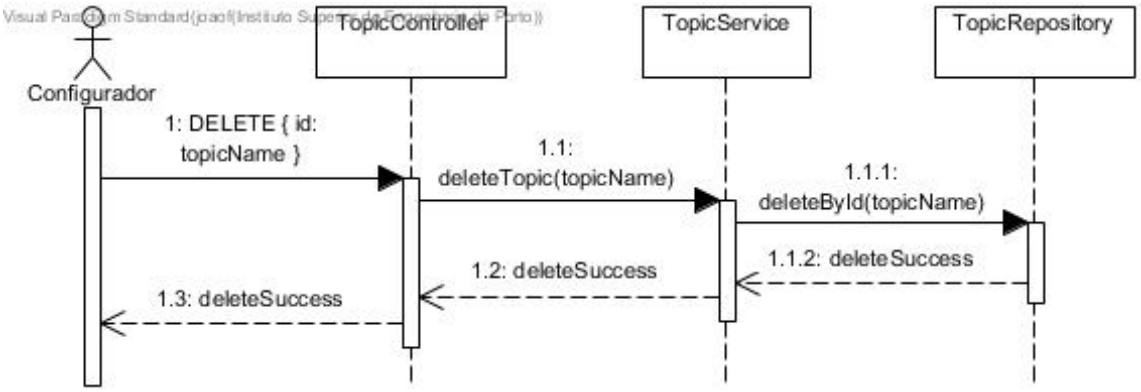

Figura 57. Diagrama de sequência para a eliminação de um tópico

4.2.2.5 REQ-4 Importação do FAQ existente

A informação já existente no portal do ISEP é importada através de um ficheiro CSV, formatado devidamente. Para isto, é necessário o processo de anotação manual para cada pergunta e resposta, ligando-as com uma intenção. O sistema converte as linhas do ficheiro para um tópico, criando-os de forma semelhante à Figura 55.

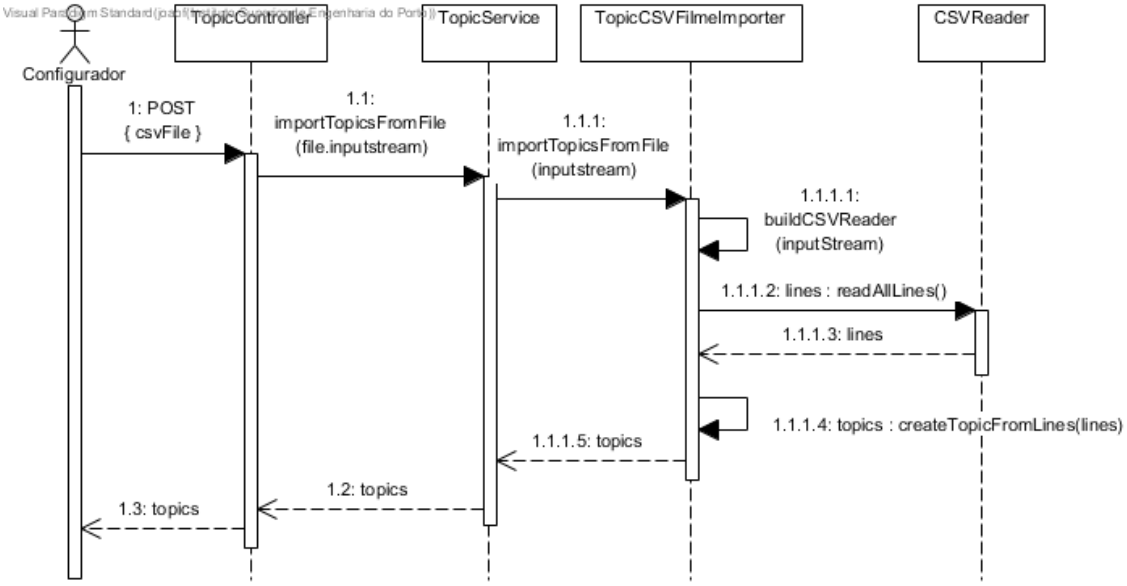

Figura 58. Diagrama de sequência para a importação do FAQ através de um ficheiro CSV

4.2.2.6 REQ-5: Executar a aprendizagem do agente

Um dos requisitos funcionais denotados é a possibilidade de provocar a definição de um modelo de classificação novo. Para isto, utiliza-se uma das funcionalidades disponibilizadas pelo RASA WEB, o *retrain*, que possibilita o treino do agente enviando a sua configuração através de um pedido HTTP (Figura 59).

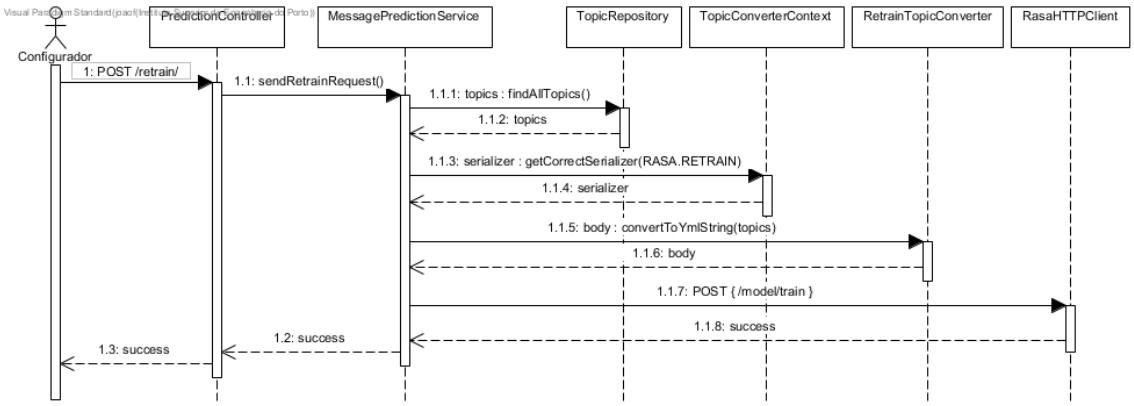

Figura 59. Diagrama de sequência para a provocação do treino do modelo

A base de conhecimento é recolhida através da base de dados, sendo que, posteriormente, os tópicos são convertidos para o formato necessário para a *framework,* YAML. É por esta razão que é necessário armazenar os exemplos de treino de cada tópico. O novo modelo fica disponível no diretório do *chatbot*, porém, só substitui o modelo em execução depois de intervenção manual.

4.2.2.7 REQ-6: Consultar as mensagens enviadas

Para consultar as ações realizadas pelo agente, o sistema tem um ecrã que permite a listagem da informação de cada mensagem enviada.

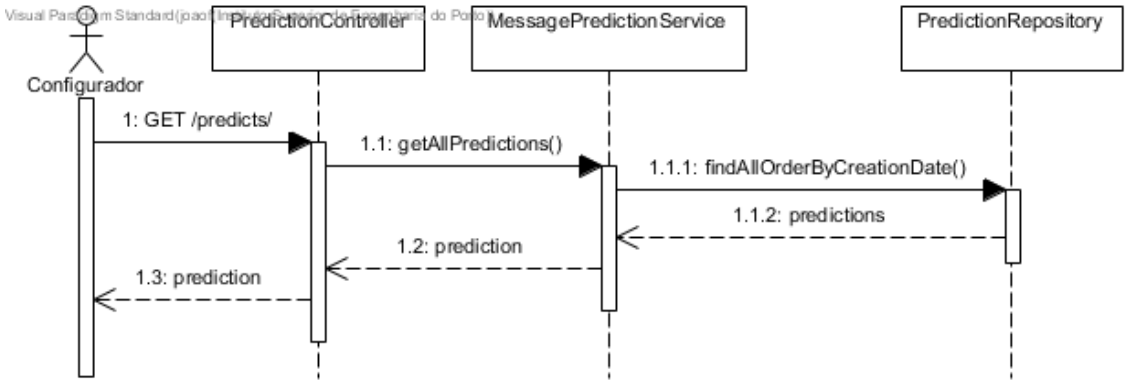

Figura 60. Diagrama de sequência para consultar previsões de mensagens

O bloco de informação guardado para cada previsão está exposto na Figura 51 como *MessagePrediction*.

5 Implementação

Ainda que não represente a sequência de desenvolvimento, este capítulo é divido em três secções, nomeadamente a implementação do serviço de tópicos, a configuração do agente, e, por fim, a exposição do protótipo implantado.

5.1 Gestão de Tópicos

Para construir um sistema intuitivo para os configuradores da Divisão Académica, permitindo a reconfiguração das funcionalidades de compreensão de texto sem parâmetros técnicos, é necessária uma camada de abstração, o serviço de tópicos. As funcionalidades, denotadas na secção 4.1.2.1, foram desenvolvidas com o objetivo de servirem como uma API.

5.1.1 Construção do modelo de negócio

Numa arquitetura em camadas, o nível central é o domínio, onde se representa toda a lógica de negócio. Neste caso, transpôs-se o modelo da Figura 40 para objetos JAVA. O Tópico representa a entidade principal de negócio, que será utilizado para armazenar toda a informação relativa a uma intenção, as respostas e, por fim, as avaliações.

De forma a facilitar a integração entre o nível de negócio e o nível de persistência, utilizou-se uma *framework* ORM (*Object Relational Mapping*), o Hibernate [85], que efetua o mapeamento de objetos com a sua representação na base de dados.

```
1.  @Entity
2.  public class Topic extends Timestampable {
3.
4.      @Id
5.      @GeneratedValue(strategy = GenerationType.AUTO)
6.      private Long id;
```

```
7.
8.      @Column(unique = true)
9.      private String intentName;
10.
11.     private String description;
12.
13.     @OneToOne(fetch = FetchType.EAGER)
14.     private Category category;
15.
16.     @OneToOne(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
17.     private RatingStatistics ratings;
18.
19.     @OneToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
20.     private List<Response> responses;
21.
22.     @OneToMany(fetch = FetchType.EAGER, cascade = CascadeType.ALL)
23.     private Set<TextExample> textExamples;
```

Código 3. Estrutura interna da entidade Tópico

Como observado pelo extrato de Código 3, a correspondência de campos é efetuada através de anotações, sendo que as relações entre objetos são definidas através da sua multiplicidade (`@OneToOne` ou `@OneToMany`). Para cumprir o requisito de auditoria de criação e modificação de objetos, utilizou-se a classe abstrata `Timestampable`, definida no Código 4, estendida pelas entidades.

```
1.  @Column(name = "created_at")
2.  protected OffsetDateTime createdAt;
3.
4.  @Column(name = "updated_at")
5.  protected OffsetDateTime updatedAt;
6.
7.  @PrePersist
8.  public void defaultValues() {
9.      updatedAt = createdAt = OffsetDateTime.now(ZoneOffset.UTC);
10. }
11.
12. @PreUpdate
13. public void updateValues() {
14.     updatedAt = OffsetDateTime.now(ZoneOffset.UTC);
15. }
```

Código 4. Atributos relativos à auditoria de entidades

Utilizando os mecanismos fornecidos pelo Hibernate, é possível definir dois atributos relativos à data de criação e modificação, sendo que são persistidos sempre que alguma atualização ocorra. A nível de repositório, Spring fornece uma classe base para entidades JPA, constituídas por ações de criação, leitura e atualização, sendo apenas necessário implementá-la, indicando qual o objeto e a chave primária (Código 5).

```
1.  public interface TopicRepository extends JpaRepository<Topic, Long>
```

Código 5. Especificação da classe de persistência

5.1.2 Criação, Atualização e Eliminação de Tópicos

A gestão de tópicos constitui o leque principal de funcionalidades do serviço, sendo que são a base de conhecimento do agente. Ainda que exista uma interface, exposta na secção 5.3, este capítulo apenas abordará os pormenores técnicos do servidor.

O ponto de entrada das funcionalidades é a camada de controladores, onde se estabelece o contrato para o pedido *web* e a sua resposta. Cada controlador está associado a uma entidade, neste caso, os tópicos, onde se distribuiu as várias funcionalidades por caminhos diferentes. O serviço é responsável por controlar os objetos de domínio.

```
1.  public TopicDTO createTopic(TopicDTO topic) {
2.      Category category =
    categoryRepository.findByName(topic.category).orElseThrow(() ->
3.              new IllegalArgumentException("Category not found"));
4.      Topic topicModel = TopicMapper.dtoToDomain(topic, category);
5.      Topic savedModel = repo.save(topicModel);
6.      return TopicMapper.domainToDto(savedModel);
7.  }
```

Código 6. Camada de serviço relativa à criação de um tópico

O Código 6 representa a lógica de criação de um tópico, separado em três etapas: a pesquisa da categoria associada, o mapeamento do objeto de visualização para domínio e, por fim, o armazenamento da entidade através da classe repositório. A logica de atualização é semelhante, apenas com uma pesquisa adicional se o tópico a atualizar existe. A eliminação de um tópico é realizada através do identificador único, como observado em Código 7.

```
1.  public void deleteById(Long id) {
2.      repo.deleteById(id);
3.  }
```

Código 7. Camada de serviço relativa à eliminação de um tópico

5.1.3 Importação de Informação

Como delineado na secção 4.1.2.1, a importação da informação já existente é realizada através da exportação de perguntas e respostas para um ficheiro CSV. Existem duas fontes de informação, nomeadamente a página de perguntas frequentes do ISEP, com assuntos como a matrícula, cartão de estudante, inscrições em exames e a conclusão do curso, e um conjunto de *e-mails* reais enviados por estudantes. Por motivos de confidencialidade, a informação pessoal foi omitida da leitura destes dados.

A anotação da base de conhecimento foi realizada de forma manual, sendo que cada tópico é identificado pelo conjunto de exemplos, a resposta associada, o nome da intenção, a descrição e, por fim, a categoria. Um exemplo é denotado na Tabela 19. Para que o sistema os consiga separar, o conjunto de perguntas é dividido através de pontos de interrogação.

Exemplos	Resposta	Intenção	Descrição	Categoria
Quanto tempo até atualizarem as propinas? As propinas não atualizam? Estou com um erro nas propinas? …	"A informação do pagamento das propinas pode demorar entre 24 a 48 horas a atualizar"	PERIODO_PAGAMENTO	Duração para atualização do pagamento de propinas	PROPINAS

Tabela 19. Extrato da intenção de atualização do pagamento das propinas

A implementação desta funcionalidade foi efetuada com recurso a uma biblioteca de compreensão de ficheiros CSV, o openCSV. É possível configurar um leitor, `CSVParser`, parametrizando de acordo com as necessidades do desenvolvedor (Código 8).

```
1. @Service
2. public class TopicCSVFileImporter implements TopicFileImporter {
3.
4.     private static final Character CSV_DELIMITER = ';';
5.     private static final String QUESTION_DELIMITER = "\\?";
6.     private static final CSVParser parser = new
   CSVParserBuilder().withSeparator(CSV_DELIMITER).withIgnoreQuotations(fals
   e)
7.             .build();
8.
9.     @Override
10.    public List<TopicDTO> importTopicsFromFile(InputStream fileInput)
   throws IOException {
11.        CSVReader reader = new CSVReaderBuilder(new
   InputStreamReader(fileInput)).withCSVParser(parser).withSkipLines(1).buil
   d();
12.        List<String[]> listOfLines = reader.readAll();
13.
14.        return
   listOfLines.stream().map(TopicCSVFileImporter::createTopicFromLine).colle
   ct(Collectors.toList());
15.    }
16. }
```

Código 8. Implementação do leitor de ficheiros CSV

O ficheiro é delimitado pelo carater ponto e vírgula, no entanto, as respostas também podem conter essa pontuação. Desta forma, para que os blocos entre aspas sejam ignorados para qualquer divisão, é necessário definir o parâmetro `withIgnoreQuotations` em `CSVParser` como falso.

O cabeçalho é evitado através da instanciação do `CSVReader` com o parâmetro `withSkipLines(1)`, posteriormente processando cada linha atomicamente para a criação do tópico.

5.2 Configuração do Agente

O ponto crucial do sistema é o núcleo do agente conversacional, especificamente a capacidade de entender linguagem natural. Como definido, esta funcionalidade é suportada pela *framework* RASA, sendo que o primeiro passo foi a criação de uma aplicação base. Após a instalação das dependências, RASA disponibiliza um comando para criação de um projeto – `rasa init` – que estrutura os conceitos base em ficheiros.

A base de conhecimento localiza-se no ficheiro `nlu.yml`, local onde se introduz as intenções que o agente irá compreender e respetivos exemplos de treino. Visto que a informação do agente está na base de dados do serviço, é necessário construir um processo de transformação para o formato requerido.

5.2.1 Transformação da base de conhecimento

Os tópicos são persistidos separadamente com o intuito de ter um suporte de informação dinâmico, ao contrário da manutenção de um ficheiro estático. Ainda que apresente vantagens a nível do cliente final, resulta numa maior complexidade de desenvolvimento, particularmente a necessidade de transformar os tópicos persistidos para formatos específicos. Para RASA, o formato é sempre `yml`.

Existem dois tipos de conversões, respetivamente para o ficheiro `nlu.yml` e, abordado na secção 5.2.4, para o pedido de treino. Desta forma, foi esquematizado um sistema que possibilite o desacoplamento do processo de transformação com o pedido final (Figura 61).

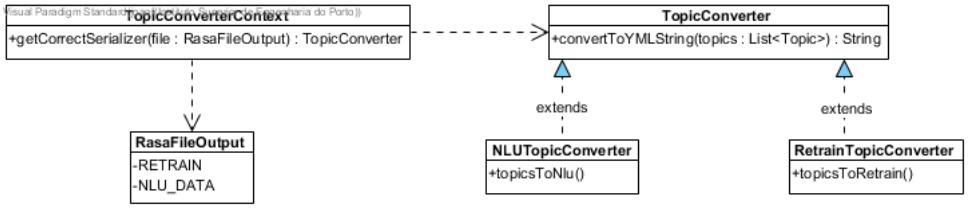

Figura 61. Sistema de transformações para a lista de tópicos existentes

Implementado no Código 9, isto permite ter um único método para qualquer transformação.

```
1.  public String getNluDataFromTopics(RasaFileOutput fileOutput) {
2.      List<Topic> topics = repo.findAllExceptFallback();
3.      return
    TopicConverterContext.getCorrectSerializer(fileOutput).convertToYMLString
    (topics);
4.  }
```

Código 9. Método de transformação de tópicos

A classe `TopicConverter`, exposta no Código 10, é constituída por métodos que permitem a manutenção de uma árvore hierárquica, convertida para uma `String` em `YML`. Além das

funcionalidades de serialização, existe um método abstrato para que as implementações possam personalizar a sua hierarquização.

```
1.  public abstract class TopicConverter {
2.
3.      YAMLTreeBuilder yamlTreeBuilder;
4.
5.      public abstract String convertToYMLString(List<Topic> topic);
6.
7.      public void addPair(String key, Object value) {
8.          this.yamlTreeBuilder.addPairToYAMLTree(key, value);
9.      }
10. }
```

Código 10. Classe abstrata para a serialização de estruturas YAML

Para o caso da transformação do conjunto de dados, o ficheiro `YML` é constituído por um mapa cuja chave é o nome da intenção e os valores os exemplos de treino (Código 11).

```
1.  public String convertToYMLString(List<Topic> topics) {
2.      NaturalLanguageDataDTO languageDataDTO = topicsToNlu(topics);
3.      List<Map<String, Object>> intentsExamplesBlock =
    languageDataDTO.nlu.stream().map(intentExamplesPair -> {
4.          LinkedHashMap<String, Object> map = new LinkedHashMap<>();
5.          map.put("intent", intentExamplesPair.intent);
6.          map.put("examples", intentExamplesPair.examples);
7.          return map;
8.      }).collect(Collectors.toList());
9.      addPair("nlu", intentsExamplesBlock);
10.     return toYAMLString();
11. }
```

Código 11. Criação da estrutura YML para a classe `NLUTopicConverter`

O método referente à linha 2, `topicsToNlu(topics)`, transforma os tópicos para um objeto com cada intenção e respetivos exemplos de texto. Um subconjunto do resultado final é exposto no Código 12.

```
1.  version: '3.1'
2.  nlu:
3.    - intent: PROCEDIMENTO_REALIZACAO_MATRICULA
4.      examples: |
5.        - Que papéis preciso para fazer a matrícula
6.        - O que fazer depois de aceite na universidade
7.        - Quais são as credenciais de acesso ao portal para matricular
8.    - intent: BOLSA_PAGAMENTO_PROPINAS
9.      examples: |
10.       - Sou estudante de bolsa de estudos da DGES, tenho de efetuar o
    pagamento do curso
11.       - É obrigatório pagar as propinas iniciais se tiver bolsa
```

Código 12. Extrato parcial do ficheiro `nlu.yml` final

Outra vertente importante da configuração do conjunto de dados é a seleção de sinónimos, permitida pelo RASA através do mesmo ficheiro. Embora o processo de CLN já use dicionários de texto, existem paralelos a nível de domínio que não são interpretados corretamente. Para o agente, definiram-se os sinónimos denotados no Código 13.

```
- synonym: matrícula
  examples: |
    - inscrição
- synonym: universidade
  examples: |
    - faculdade
    - instituto
    - escola
    - instituição
- synonym: portal
  examples: |
    - site
    - online
- synonym: cadeira
  examples: |
    - unidade curricular
    - disciplina
    - UC
- synonym: CTESP
  examples : |
    - CTeSP
    - cursos técnicos profissionais
    - cursos técnicos superiores profissionais
```

Código 13. Lista de sinónimos presentes no ficheiro de dados

Para que o modelo aprenda a classificar os sinónimos com mais precisão, é importante que os dados de treino retenham exemplos de cada ocorrência. A fase de mapeamento semântica só é realizada após a extração de significado [86].

5.2.2 Anotação da Base de Conhecimento

Como referido, existem duas fontes de informação, o portal da Divisão Académica e extratos de correio eletrónico, fornecidos pelo próprio departamento. Inicialmente, para o desenvolvimento de um produto mínimo viável, utilizou-se apenas a primeira fonte, sendo que se estabeleceu uma base de conhecimento com as dúvidas mais frequentes.

A anotação dos dados foi manual, através da atribuição de uma categoria a cada intenção, facilitando a análise de qualidade tendo em conta cada tema. O processo resultou em 15 intenções, que foram reduzidas em 12 por motivos de simplicidade. A segmentação de tópicos semelhantes, ainda que produzisse mais especificidade, era prejudicial para a capacidade de interpretação do agente. Demasiado pormenor contribui para um processo de compreensão mais complexo devido à confusão de assuntos.

Categoria		Descrição
Inscrição	PROCEDIMENTO_REALIZACAO_MATRICULA	Como fazer a matrícula através do Portal e documentos necessários
	QUANTIDADE_CREDITOS_INSCRICAO	Informação relativa aos créditos e ao seu uso
	ATRASO_RENOVACAO_INSCRICAO	Renovação de inscrição com UCs em atraso
	PROCEDIMENTO_DECLARACAO_MATRICULA	Consultar Declaração de Matrícula no portal e como resolver erros de não abertura (Ano Zero, *Popups*)
	Erros na declaração de matrícula	
	Consultar declaração de Matrícula para Ano Zero	
Propinas	PROCEDIMENTO_PAGAMENTO_PROPINAS	Como fazer o pagamento de propinas no portal e respetivo faseamento
	Faseamento de pagamento de propinas	
	PLANO_INCORRETO_PROPINAS	Plano incorreto de propinas
	BOLSA_PAGAMENTO_PROPINAS	Pagamento de propinas por parte de bolseiros
	PERIODO_ATUALIZACAO_PAGAMENTO	Duração de atualização do pagamento das propinas
Passe SUB23	PROCEDIMENTO_DECLARACAO_SUB23	Consultar Declaração SUB23
Horário	PROCEDIMENTO_DECLARACAO_HORARIO	Consultar Declaração de horário
	PROCEDIMENTO_ESCOLHA_HORARIO	Como funcionam as mudanças de turma
Ano Zero	INSCRICAO_ANO_ZERO	Inscrições para Ano Zero

Tabela 20. Base de conhecimento inicial importada do portal da Divisão Académica

Visto que as intenções da Tabela 20 foram extraídas de uma página *web*, não existe mais do que uma questão por pedido. Para enriquecer a habilidade do agente, foi desenvolvido um formulário partilhado com alunos da Universidade do Porto, exposto no Anexo A, onde se obteve 127 novos exemplos.

O próximo passo foi a anotação dos e-mails enviados pela Divisão Académica, sendo que este ficheiro continha 104 interações. O produto final inclui 15 novas intenções, resultantes de 86 exemplos de texto, sendo que 18 *emails* foram ignorados por serem demasiado específicos. Novamente, depois de testes automáticos, houve uma redução para 14 intenções devido ao fator de semelhança (Tabela 21).

Categoria	Intenção	Descrição
Inscrição	INSCRICAO_MESTRADO	Como funcionam as candidaturas para mestrado
	INSCRICAO_MAIS_23_ANOS	Inscrições para pessoas com mais de 23 anos

Categoria	Intenção	Descrição
	INSCRICAO_CTESP	O que são cursos técnicos superiores profissionais e como inscrever
	INSCRICAO_UCS_ISOLADAS	Inscrição em unidades curriculares fora do plano do curso
	INSCRICAO_REINGRESSO	Regime de reingresso e mudança de curso interno à universidade
Exames	INSCRICAO_EXAME_EPOCA_NORMAL	Procedimentos relativos à inscrição em exames de época normal
	INSCRICAO_EXAME_EPOCA_ANORMAL	Procedimentos relativos à inscrição em exames de época de recurso/especial
	INSCRICAO_EXAME_MELHORIA	Procedimentos relativos à inscrição em melhorias
	PROCEDIMENTO_RETIFICACAO_EXAME	Possibilidade de anular a inscrição em exames
Cartão	INFORMACOES_CARTAO_ISEP	Instruções para o cartão estudante do ISEP
Regime Letivo	PROCEDIMENTO_MUDANCA_REGIMES	Instruções para mudanças de regimes
Equivalências	PROCEDIMENTO_EQUIVALENCIAS	Como pedir equivalências no Portal
Faltas	PROCEDIMENTO_FALTAS	Como justificar faltas no Portal, inclusive exemplo de Covid
	Faltas de Covid	
Estatutos	PROCEDIMENTO_ TRABALHADOR_ESTUDANTE	Como obter estatuto trabalhador estudante

Tabela 21. Adição de tópicos após a extração do correio eletrónico

Por fim, depois do teste realizado com o protótipo, exposto na secção 6.4.2, foram denotados novos tópicos cruciais ao bom funcionamento do agente. Para além da capacidade de diálogo natural, nomeadamente os cumprimentos, adicionaram-se referências relativas ao funcionamento da Divisão Académica. Os exemplos de texto foram extraídos do sistema após a realização do piloto.

Categoria	Intenção	Descrição
Cumprimentos	CUMPRIMENTO_INICIAL	Saudação inicial
	DESPEDIDA	Despedida
Inscrição	INSCRICAO_LICENCIATURA	Candidatar a Licenciatura
	CONCLUSAO_CURSO	Condições excecionais para conclusão do Curso

Categoria	Intenção	Descrição
	TRANSICAO_ANO	ECTs de transição de ano
	MANTER_NOTA_FREQUENCIA	Como manter nota de frequência
Contactos	CONTACTOS_DIVISAO_ACADEMICA	Contactar a Divisão Académica
	REGULAMENTOS_ISEP	Consultar regulamentos do ISEP
	BASE_INFORMACAO	Qual as categorias de conhecimento do agente

Tabela 22. Tópicos criados após a sessão de *feedback* com a Divisão Académica

A base de conhecimento final do agente inclui 38 tópicos, inteiramente presente no Anexo B. Não se configurou qualquer reconhecimento de entidades, visto que não foi necessário para qualquer intenção.

5.2.3 Compreensão de Linguagem Natural

Assim que os dados de treino estejam preparados, é possível proceder com a criação do modelo de classificação. Para RASA, o processo de CLN segmenta-se em três componentes, os *tokenizers*, *featurizers* e *intent classifiers*.

Cada implementação tem os seus parâmetros, sendo que a sua especificação é indicada através do ficheiro `config.yml`. O fluxo de processamento funciona sequencialmente de acordo com a ordem da configuração, sendo que cada componente recebe valores de entrada, processa-os, e retorna valores de saída. A configuração do protótipo final é demonstrada no bloco de Código 14.

```
1.  pipeline:
2.    - name: WhitespaceTokenizer
3.    - name: LanguageModelFeaturizer
4.      model_name: "bert"
5.      model_weights: "rasa/LaBSE"
6.    - name: LexicalSyntacticFeaturizer
7.    - name: CountVectorsFeaturizer
8.      analyzer: "char_wb"
9.      min_ngram: 1
10.     max_ngram: 4
11.   - name: DIETClassifier
12.     constrain_similarities: True
13.     epochs: 200
```

Código 14. Extrato parcial do ficheiro `config.yml` final

Cada componente tem diferentes funções, delineadas na Tabela 23.

Componente	Funcionalidade	
Tokenizer	`WhitespaceTokenizer`	Segmenta a frase por espaços
Featurizer	`LanguageModelFeaturizer`	Interpreta o significado do *token* consoante um modelo de linguagem pré-treinado.

Componente	Funcionalidade	
	`LexicalSyntacticFeaturizer`	Adiciona a função frásica do *token*
	`CountVectorsFeaturizer`	Adiciona a frequência do *token* nos exemplos de treino
Classifier	`DIET Classifier`	Realiza a previsão de intenção através do uso de transformadores e redes neuronais (secção 2.4.4.1).

Tabela 23. Responsabilidade de cada componente na configuração escolhida

A construção de um modelo de classificação é possível através do comando `rasa train nlu`, que cria um ficheiro `tar` que pode ser incluído na execução do agente. O teste de várias configurações foi efetuado e analisado na secção 6.4.1.

5.2.3.1 Processamento de texto

Internamente, o sistema realiza a compreensão de texto em três fases, nomeadamente a separação da frase em *tokens*, a criação de atributos para cada um e, por fim, a previsão da intenção obtida. Em caso de treino, esta última é utilizada para comparar com os exemplos de validação, originando um valor de similaridade que é utilizado para ajustar os pesos do classificador. Em contrapartida, em caso de classificação, a intenção obtida é devolvida ao utilizador. A presente secção apresenta um exemplo do processamento efetuado pelo agente, tendo em consideração a configuração ilustrada no Código 14.

Primeiramente, consoante o separador espaço, o módulo de PLN transforma a frase recebida em *tokens*, como demonstrado no bloco de Código 15. O atributo `text_tokens` enumera as palavras processadas, especificamente o seu índice de início e fim.

```
{
    "text": "quero pagar as propinas",
    "text_tokens": [
        [0,5],
        [6,11],
        [12,14],
        [15,23]
    ]
}
```

Código 15. Resultado do *tokenizer* com o separador espaço

De seguida, denota-se a criação de *token embeddings, as* estruturas que armazenam cada palavra e as suas características. Estas são originadas através dos *featurizers* definidos. O `CountVectorsFeaturizer` simboliza o algoritmo *Bag Of Words*, que converte cada elemento para um vetor numérico com a frequência de ocorrência nos exemplos de treino. Os parâmetros *n-gram* simbolizam o tamanho mínimo (1) e máximo (4) da sequência de palavras que será comparada.

Por outro lado, `LexicalSyntacticFeaturizer` atribui a cada elemento uma função sintática, também através de um vetor numérico. É, para RASA, o equivalente de *POS-Tagging*.

Os modelos de linguagem funcionam de forma semelhante, compostos por um codificador pré-treinado (BERT) que, consoante a frase recebida, origina um significado semântico. Ainda que a *framework* não disponibilize os resultados do processo, este é representado concetualmente na Figura 62.

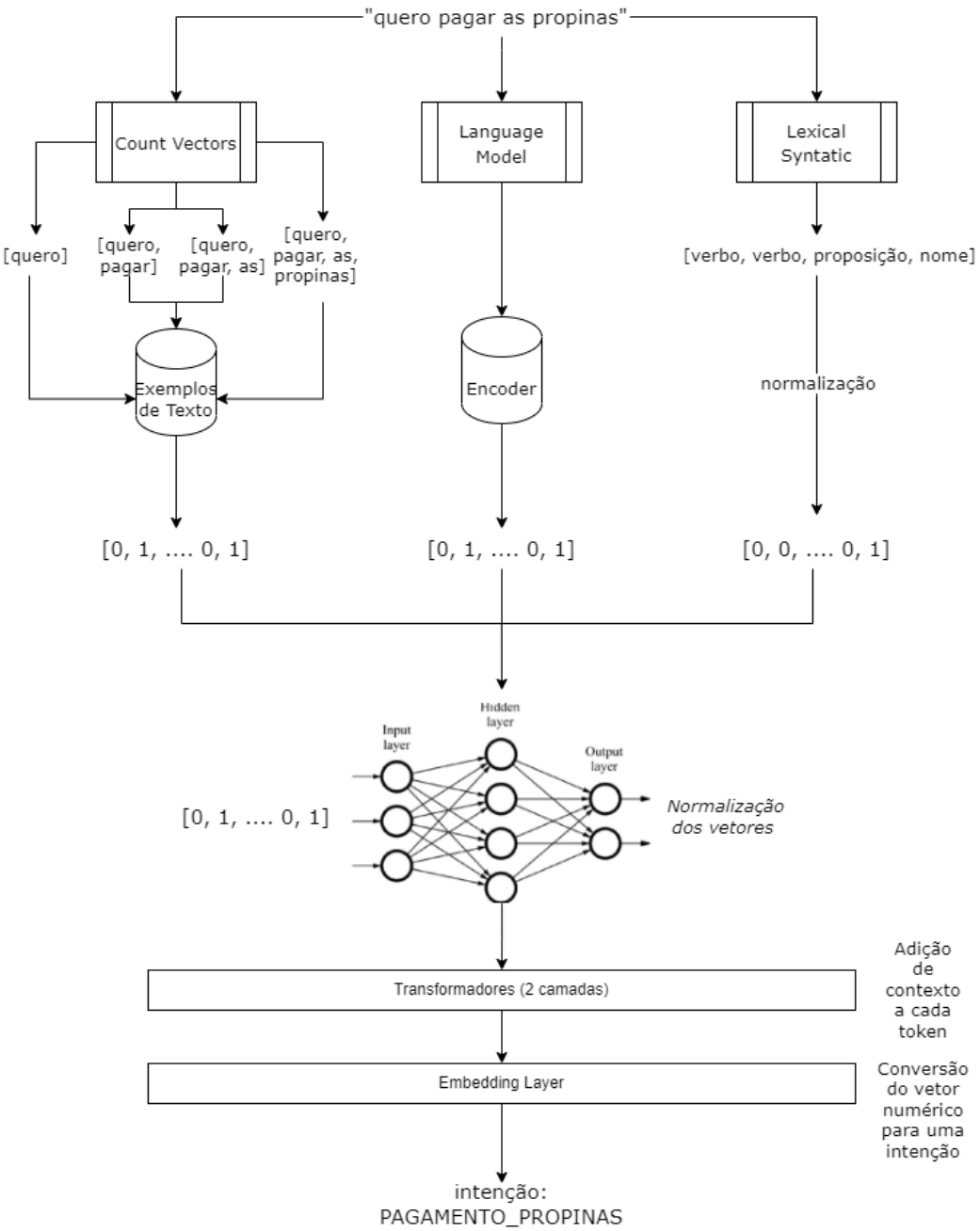

Figura 62. Criação das características de cada token

Após receber os vetores finais da fase de PLN, a camada de aprendizagem profunda é responsável por computar a intenção obtida através de funções numéricas. O tópico é devolvido após passar pela camada de *embedding*, que converte o vetor espacial final para uma das intenções existentes.

5.2.3.2 Envio de Mensagem

Assim que a fase de compreensão de texto esteja terminada, a intenção está pronta para ser emitida ao estudante. Como estipulado na fase de desenho, presente na secção 4.2.1.2, o ponto intermédio entre o portal e o agente é o serviço de tópicos. Este último recebe o texto do pedido e reencaminha-o para o *chatbot*, obtendo o conteúdo da análise do agente.

```
1.  {
2.      "text": "quero pagar as propinas",
3.      "intent": {
4.          "name": "PROCEDIMENTO_PAGAMENTO_PROPINAS",
5.          "confidence": 0.9905802607536316
6.      },
7.      "entities": [],
8.      "text_tokens": [
9.          [0,5],
10.         [6,11],
11.         [12,14],
12.         [15,23]
13.     ],
14.     "intent_ranking": [
15.         {
16.             "name": "PROCEDIMENTO_PAGAMENTO_PROPINAS",
17.             "confidence": 0.9905802607536316
18.         },
19.         {
20.             "name": "PLANO_INCORRETO_PROPINAS",
21.             "confidence": 0.0026339690666645765
22.         },
23.         ...
24.     ]
25. }
```

Código 16. Conteúdo JSON da resposta originada pelo agente

O tópico final é obtido através do bloco `intent`, em conjunção com a percentagem de confiança associada. Caso exista na base de dados, a informação permite fazer a conexão da intenção com uma resposta e confirmar se a confiança está acima do mínimo delineado.

Em termos técnicos, isto é realizado através da leitura da resposta JSON, convertendo-a para um objeto JAVA. Caso a interpretação não seja de confiança, retorna-se a resposta base.

```
1.  @Service
2.  public class RasaChatbotIntegrationService implements
    ChatbotIntegrationService {
3.
4.      @Value("${rasa.chatbot.url}")
5.      private String SERVER_URL;
6.      @Value("${rasa.chatbot.token}")
7.      private String AUTH_TOKEN;
8.
9.      private static final JsonGenericMapper<RasaPredictionResponse>
    JSON_RESPONSE_MAPPER = new
    JsonGenericMapper<>(RasaPredictionResponse.class);
10.     private static final Double CONFIDENCE_THRESHOLD = 0.7;
11.     private static final String PARSE_MESSAGE_ENDPOINT = "/model/parse";
12.
```

```
13.     @Override
14.     public MessagePrediction generateResponseFromQuestion(String
    question) {
15.         TextBlockDTO rasaBody = new TextBlockDTO(question);
16.         try {
17.             String returnedMessage = new
    RasaHttpClient().getIntentPrediction(rasaBody);
18.             RasaPredictionResponse response =
    JSON_RESPONSE_MAPPER.readObjectFromJsonString(returnedMessage);
19.
20.             if(response.intent.confidence < getConfidenceThreshold()) {
21.                 Topic fallbackTopic =
    topicRepo.findByIntentName(FALLBACK_INTENT).orElseThrow(() ->
22.                         new ChatbotResponseNotUsableException("Fallback
    topic not found"));
23.                 return new MessagePrediction(fallbackTopic, question, 0d,
    null);
24.             }
25.
26.             Topic topic =
    topicRepo.findByIntentName(response.intent.name).orElseThrow(() ->
27.                     new ChatbotResponseNotUsableException("Intent not
    found"));
28.             return new MessagePrediction(topic, question,
    response.intent.confidence, null);
29.         } catch (Exception e) {
30.             throw new ChatbotResponseNotUsableException(e.getMessage());
31.         }
32.     }
```

Código 17. Lógica para a leitura da interpretação do agente

Como se pode analisar pela linha 4 e 6 do extrato Código 17, tanto o local do servidor como o *token* de segurança são injetados através das propriedades da aplicação. Este paradigma é especificamente útil devido à centralização de dados, sendo que, caso alterados, apenas será refletido no ficheiro de configuração.

O método que envia o pedido HTTP é intuitivo, utilizando a classe `RestTemplate`, como observado em Código 18.

```
1.  private class RasaHttpClient {
2.
3.          public String getIntentPrediction(TextBlockDTO body) {
4.              RestTemplate rest = new RestTemplate();
5.              return rest.postForEntity(SERVER_URL + PARSE_MESSAGE_ENDPOINT
    + "?token=" + AUTH_TOKEN, body, String.class).getBody();
6.          }
7.  }
```

Código 18. Construção do pedido HTTP para o agente

Para satisfazer o requisito não funcional de auditoria de mensagens, construiu-se um sistema que armazena todas as interações e a sua previsão de intenção. Este processo é espoletado assim que o método referido acima termine, consoante uma propriedade `IS_AUDIT_ENABLED` (Código 19).

```
1.  public MessagePredictionDTO generateResponseFromMessage(String text) {
2.      MessagePrediction prediction =
    chatbotService.generateResponseFromQuestion(text);
3.      if(IS_AUDIT_ENABLED) {
4.          MessagePrediction predictionFinal = repo.save(prediction);
5.          return PredictionMapper.domainToDto(predictionFinal);
6.      }
7.      return PredictionMapper.domainToDto(prediction); // no id
8.  }
```

Código 19. Mecanismo de auditoria de interações

5.2.4 Espoletar o processo de treino

Para treinar um novo modelo de classificação, o serviço utiliza o ponto de saída `/model/train`, disponibilizado por RASA. A especificação do pedido é bastante complexa, sendo que implica a informação completa referente às propriedades do agente, particularmente a configuração de CLN, as intenções e os exemplos de treino. Essencialmente, é uma junção do mapeamento demonstrado na secção 5.2.1 com o esquema de aprendizagem.

O primeiro passo é a recolha dos tópicos e respetivos exemplos de texto existentes na base de dados, excluindo as categorias excecionais. De seguida, através da indicação do formato de treino, utiliza-se o sistema de abstração de transformações para compor o conteúdo necessário para o envio ao agente.

```
1.  public void sendRetrainRequest() {
2.      List<Topic> topics =
    topicRepo.findAllByCategoryNameNotIn(Arrays.asList(CategoryEnum.FALLBACK.
    name(), CategoryEnum.UNKNOWN.name()));
3.      String bodyContent =
    TopicConverterContext.getCorrectYamlSerializer(RasaFileOutput.RETRAIN).co
    nvertToYMLString(topics);
4.      chatbotService.retrainModel(bodyContent);
5.  }
```

Código 20. Camada de serviço relativa à provocação de uma nova aprendizagem

O método `chatbotService.retrainModel(bodyContent)` é responsável por enviar o pedido HTTP, semelhante ao extrato demonstrado no Código 20.

5.3 Protótipo

Por limitações de tempo, a integração do agente com o portal da Divisão Académica não foi conseguida. Contudo, construiu-se um protótipo para demonstrar a capacidade do sistema desenvolvido, expondo todas as funcionalidades desenhadas através de uma interface visual implementada em React [87]. O objetivo deste protótipo foi estruturar uma interface simples que conseguisse integrar o ecossistema produzido.

5.3.1 Implantação

A implantação do ecossistema foi seguida tal como representada na Figura 46, utilizando uma máquina virtual hospedada pelo Azure [88]. A execução do núcleo do agente é suportada com recurso ao protocolo de segurança da *framework* RASA, com um *token* secreto, um argumento que permite a comunicação HTTP e, por fim, o modelo de classificação a utilizar. O serviço *web* é uma aplicação JAVA que é executada através do JRE (*Java Runtime Environment*) nativo da máquina virtual Ubuntu, enquanto a interface visual é hospedada através do ambiente de execução *Node* [89].

A Tabela 24 demonstra as configurações de implantação e o protocolo de comunicação.

Componente	Comando de Execução	Portas Utilizadas
Núcleo do Agente	`rasa run --enable-api -token ${token} -m models/${model}`	5005 (HTTP)
Serviço de Tópicos	`java -jar ${jar}`	8080 (HTTP)
Protótipo	`npm run build`	3000 (HTTP)

Tabela 24. Configuração de redes na máquina virtual

Isto significa que a porta 3000 tem de estar aberta ao público, enquanto as restantes permanecem privadas para processamento interno.

5.3.2 Interface final

O protótipo subdivide-se em três grandes secções, nomeadamente a página de conversa, o painel de gestão de tópicos e, por fim, a lista de mensagens observadas.

5.3.2.1 Página de Arranque

A página principal é simples, constituída por um bloco de conversa onde uma linha representa o agente ou, caso introduzida alguma pergunta, o utilizador. Para efeitos de simplificação, cada mensagem tem uma cor (Figura 63).

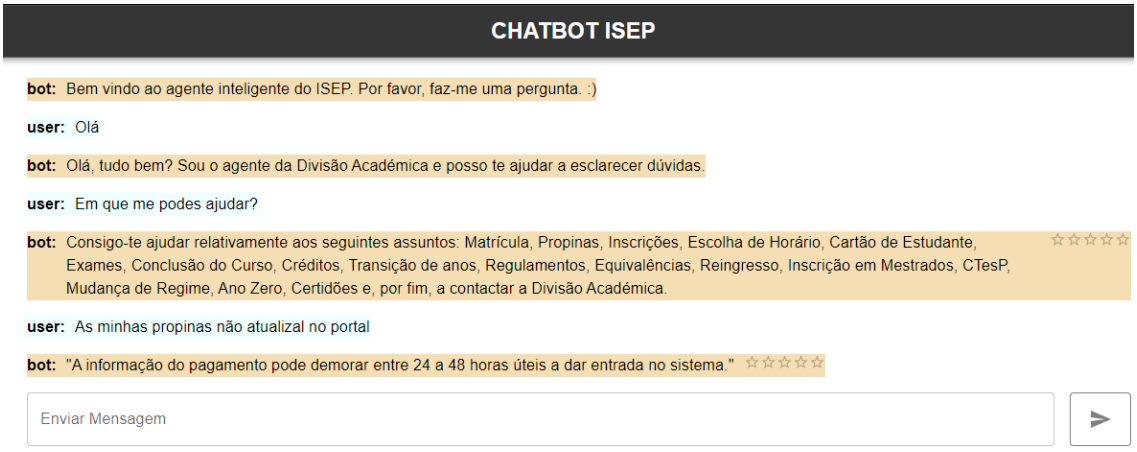

Figura 63. Página inicial do protótipo

Caso um tópico seja diretamente respondido, o sistema disponibiliza a funcionalidade de avaliar a interação numa escala de 0 a 5, como ilustrado na Figura 64.

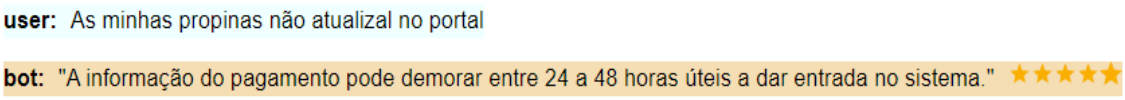

Figura 64. Possibilidade de avaliação de uma interação interpretada pelo agente

Para os tópicos que não tenham um propósito direto de resposta, como cumprimentos iniciais, despedidas e pedidos de esclarecimento, não existe a possibilidade de avaliação.

5.3.2.2 Painel de Tópicos

Este painel tem como objetivo permitir aos configuradores adicionar e modificar tópicos da base de conhecimento do agente e espoletar um novo processo de treino. Exposta na Figura 65, a página consiste numa tabela que dispõe todos os tópicos existentes e respetiva informação relevante.

Figura 65. Extrato da tabela de tópicos

É possível editar a resposta de um tópico ao expandi-lo, pressionando a sua linha duas vezes (Figura 66).

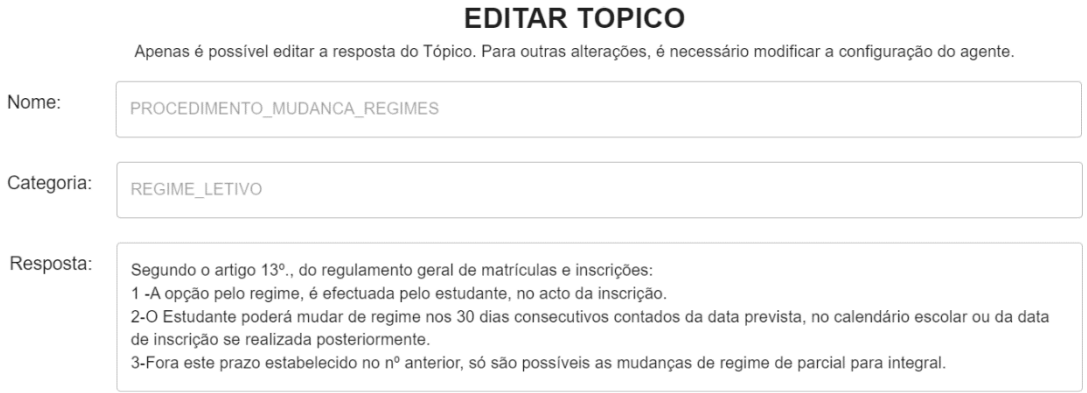

Figura 66. Janela para editar a resposta de um tópico

A mesma janela existe para a criação de um tópico, num botão localizada abaixo da tabela. Para efeitos de simplificação, este não foi exposto. O mesmo se aplicou ao botão de treino, sendo que não expõe nada exceto uma notificação de sucesso.

5.3.2.3 Auditoria de Mensagens

Para processos de análise de qualidade, é possível visualizar a lista de mensagens trocadas com o agente, qual a avaliação dada e, ainda, a intenção identificada. Como ilustrado na Figura 67, é uma lista que contém toda esta informação mais a data de auditoria.

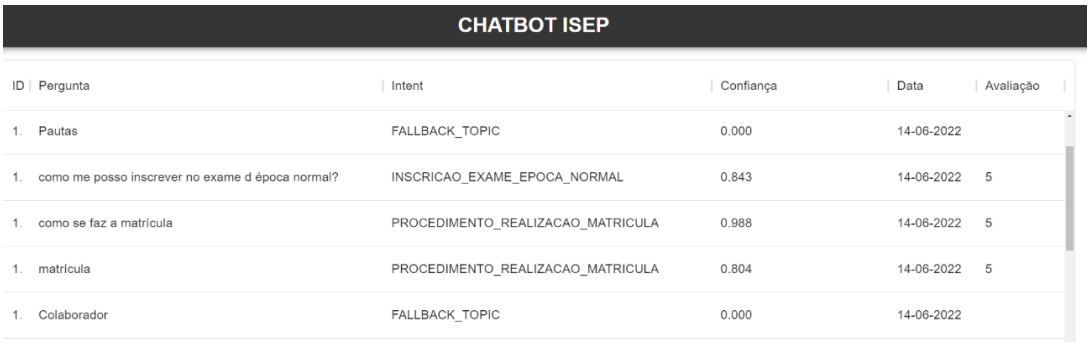

Figura 67. Tabela de auditoria de mensagens

O protótipo permitiu fazer um teste com os alunos da Escola, utilizando os seus comentários para avaliar a qualidade do sistema, como exposto na secção 6.4.2. É uma visualização do seu potencial que permite ponderar sobre a integração com o portal do ISEP.

6 Experimentação e Avaliação da Solução

O projeto desenvolvido tem como objetivo a construção de um agente conversacional capaz de reagir corretamente a pedidos dos estudantes e, consequentemente, auxiliar a Divisão Académica no esclarecimento de dúvidas. Para isto, é necessário realizar vários tipos de testes para auferir a capacidade de reação do sistema relativa à intenção do utilizador. A presente seção descreve as hipóteses que necessitam de revisão para comprovar o sucesso da solução e as respetivas metodologias de prova.

6.1 Hipótese

O objetivo principal do sistema proposto é agilizar o processo de atendimento ao estudante por parte da Divisão Académica. Pretende não só diminuir a carga de trabalho deste departamento, como também melhorar a experiência no que toca ao apoio aos membros da Escola. Desta forma, a hipótese alternativa é definida como a utilização frequente do agente com avaliações positivas, enquanto a hipótese nula consiste na depreciação do agente e, por consequente, os estudantes continuarem a utilizar os canais habituais da Divisão Académica (Tabela 25).

Hipótese Nula	O agente conversacional é ineficaz no esclarecimento de dúvidas e os estudantes remetem-se aos canais habituais da Divisão Académica
Hipótese Alternativa	A implementação do agente conversacional é eficaz no esclarecimento de dúvidas e contribui para o melhoramento do *helpdesk* do ISEP

Tabela 25. Definição da hipótese nula e alternativa

Ainda que a diminuição de carga de trabalho não consiga ser testada, é possível analisar a qualidade da interação com o agente conversacional, tanto diretamente como indiretamente.

Assim, para a avaliação da hipótese delineada, é importante qualificar a capacidade do componente de compreensão do discurso do emissor. O seu desempenho será avaliado através da análise de métricas aplicadas ao modelo de classificação de intenções. É expectável que a experiência do utilizador varie consoante a preparação do sistema, sendo que isto implica não só a vertente de aprendizagem profunda, como de compreensão de texto. Para isto, estabelece-se uma análise comparativa do modelo variando não só o algoritmo de aprendizagem e os seus parâmetros, como também a etapa de pré-processamento de dados. Estuda-se assim o impacto relativo à sua capacidade de previsão.

Por outro lado, uma métrica tão fiável como a capacidade do sistema é a avaliação direta do utilizador. Isto foi um dos requisitos funcionais, nomeadamente a construção de um mecanismo de avaliação numérica para que os estudantes conseguissem qualificar o nível de ajuda que obtiveram. Para isto, realizaram-se testes reais, demonstrados na secção 6.4.

6.2 Métricas de Avaliação

As métricas de avaliação diversificam-se consoante o tipo de avaliação que está a ser elaborada. Como referido anteriormente, existe a avaliação indireta, ao calcular a capacidade de previsão dos modelos de classificação, e a avaliação direta, através do *feedback* dos utilizadores.

No que toca à qualificação de modelos de previsão, as métricas mais relevantes são o cálculo da eficácia e eficiência. Braun [25], IBM Lab [63] e Thorat [59] seguem esta suposição, avaliando diferentes *frameworks,* ou algoritmos de aprendizagem, tendo em conta o seu desempenho na classificação de intenções. De forma igual, a avaliação da solução proposta é realizada através do mesmo critério, utilizando métricas relativas ao sucesso de classificação.

Por outro lado, temos o nível de satisfação dos estudantes, que, posteriormente, é processado em conjunto.

6.2.1 Capacidade de previsão

Os métodos de classificação são avaliados conforme a sua aptidão de previsão, extraindo o seu desempenho com métricas derivadas de uma matriz de confusão. De forma a conseguir obter estes critérios, é primeiro necessário considerar o tipo de resultados que uma previsão possa ter, sendo que se utiliza uma classificação binária entre os valores reais e os previstos:

1. *True positive* (TP) simboliza uma instância em que o valor positivo é previsto corretamente;
2. *True negative* (TN) simboliza uma instância em que o valor negativo é previsto corretamente;
3. *False positive* (FP) simboliza uma instância em que o valor positivo não é previsto corretamente;

4. *False negative (FN)* simboliza uma instância em que o valor negativo não é previsto corretamente.

Estes quatro fatores são empregues para a construção da matriz de confusão, um artefacto que ilustra a quantificação da qualidade de previsão (Figura 68).

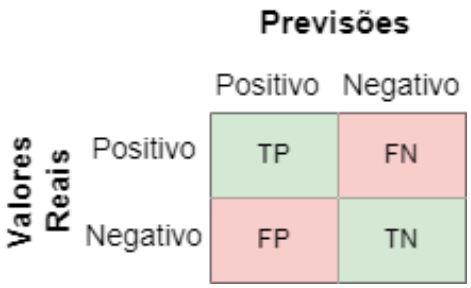

Figura 68. Matriz de confusão relativa à classificação de respostas de acordo com um tópico

Accuracy a é uma medida que representa a taxa de acerto geral.

$$a = \frac{TP + TN}{TP + FN + FP + TN}$$

A precisão p é a percentagem de respostas que o classificador sinaliza que realmente respondem ao tópico. Uma precisão de 93% significa que 7% das respostas não estão relacionadas com o tópico$(TP + FP)$.

$$p = \frac{TP}{TP + FP}$$

R*ecall* r é a percentagem de respostas ao tópico que são detetadas pelo classificador. Um *recall* de 90% significa 10% de respostas ao tópico não são usadas pelo modelo.

$$r = \frac{TP}{TP + FN}$$

O *fallout* f denota o rácio entre respostas ao tópico não-relevantes relativamente a todas as respostas não-relevantes disponíveis.

$$f = \frac{FP}{FP + TN}$$

Por fim, *F1-score* $f1$ representa a média harmónica entre a precisão e o *recall*.

$$f1 = \frac{2TP}{2TP + FP + FN}$$

No contexto de agentes conversacionais, estas medidas são utilizadas para avaliar a qualidade das respostas. A matriz de confusão final representa a sua aglomeração em simultâneo, constituída por uma dimensão de $n * n$, onde n simboliza o número de tópicos a analisar (Figura 69).

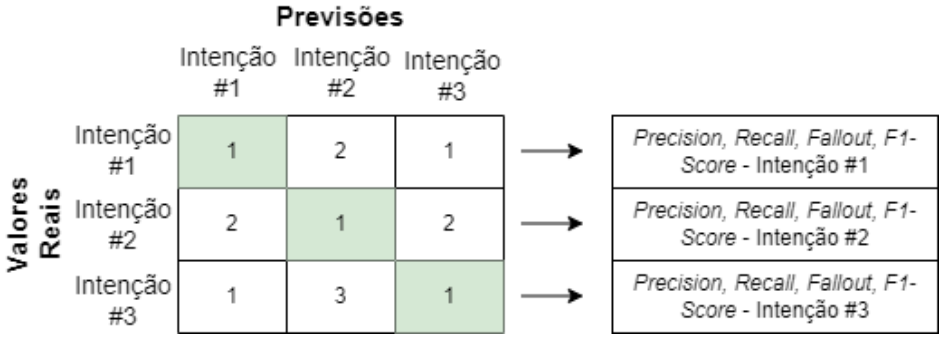

Figura 69. Exemplo de matriz de confusão para comparação de intenções

Por conseguinte, a precisão representa a frequência de classificações corretas para uma intenção, tendo em conta apenas as previsões dessa classe. Em contrapartida, o *recall* expõe a percentagem de exemplos de um tópico que efetivamente reverteram para essa previsão [90]. Neste caso, visto que são relacionadas à presciência de cada intenção, as métricas estão presentes individualmente em cada linha. O artefacto representa o desempenho na interpretação de um tópico, demonstrando ainda uma comparação com os restantes. É relevante para a avaliação do *chatbot* visto que, para além de permitir a extração de métricas globais, permite avaliar atomicamente cada intenção.

6.2.2 Avaliação do utilizador

Para além da capacidade de aprendizagem, o *chatbot* também é avaliado tendo em conta a experiência do utilizador. Por conseguinte, e tendo em conta o requisito funcional denotado na secção 4.2.2.2, o estudante tem a opção de classificar a resposta dada pelo agente numa escala de valor crescente entre 0 a 5.

Este valor é importante para confirmar se, ainda que o discurso do utilizador seja enquadrado corretamente pelo *chatbot*, a resposta é compreensível e esclareça a sua dúvida. Utilizando medidas como a média e o desvio padrão, é possível auferir, respetivamente, a qualidade das respostas e possíveis inconsistências. A avaliação é associada atomicamente a uma intenção.

6.3 Metodologia de Avaliação

As medidas relativas à capacidade de previsão são calculadas com recurso à ferramenta utilizada, RASA, que permite a geração da matriz de confusão sempre que um modelo é treinado. Com este artefacto, o cálculo das métricas conclui-se com a aplicação das fórmulas estipuladas na secção anterior.

Adicionalmente, RASA permite conferir se a base de conhecimento é, ou não, suficiente para alcançar o máximo *F1-Score*, ao verificar se o modelo tem um melhor desempenho consoante diferentes porções de dados de treino.

O processo de teste funciona de forma sequencial, separando, como mencionado, dados para treino e para validação:

1. Divisão do universo de dados, presente em `nlu.yml`, com uma porção para treino e outra para validação;
2. Treino dos modelos consoante a nova porção de dados;
3. Avaliação de cada modelo, recolhendo o seu *F1-score,* as matrizes de confusão e os erros de previsão.

Atomicamente, o procedimento de testes é demonstrado pela Figura 70. A segmentação base, tal como indicado na figura, é 80% para dados de treino e 20% para validação, no entanto, pode ser manualmente configurada.

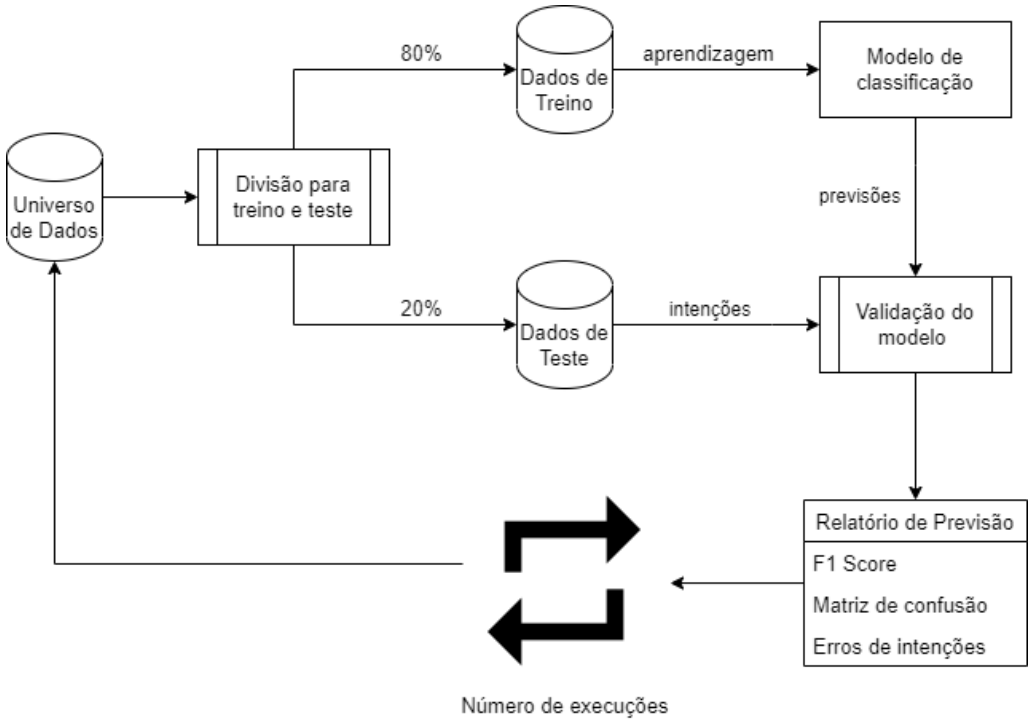

Figura 70. Processo iterative de testes em RASA

Visto que o treino de um modelo de classificação não é determinístico, é possível repetir o processo de avaliação. A matriz de confusão e os erros de previsão são ilustrados individualmente, enquanto a análise da métrica *F1-Score* é processada como uma média de todas as iterações. Caso esta última continue a aumentar com a presença de mais dados, o modelo pode beneficiar da adição de exemplos de texto. Em contrapartida, caso o *F1-Score* se mantenha a um nível estável, denota-se que o modelo está em sintonia com a quantidade de informação. O universo de dados está presente no Anexo B, sendo que se delinearam as seguintes porções de dados para treino: 80%, 70% e 50%.

No que toca à avaliação direta do utilizador, o protótipo foi exposto publicamente para a realização de testes precoces à sua implantação no ISEP. A aplicação foi exibida para os atuais estudantes do Mestrado de Engenharia Informática, indicando que deveriam fazer a avaliação

de cada resposta. Tirando partido do modelo de dados definido, é possível retirar o número de avaliações o valor médio por tópico e o desvio padrão, como demonstrado pela Figura 71.

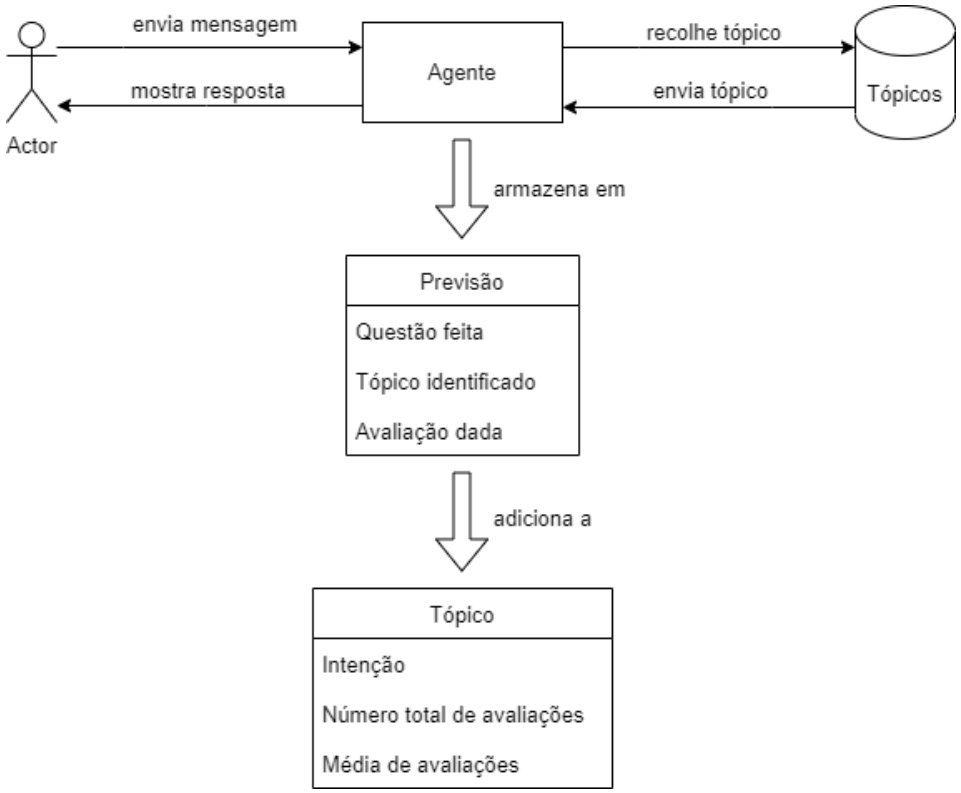

Figura 71. Utilidade e resultados do armazenamento de interações do sistema

O protótipo esteve em execução durante 12 dias, entre 12 a 24 de Junho, sendo que se obteve um total de 250 interações.

6.4 Análise de Resultados

A presente secção apresenta os resultados extraídos do sistema, segmentados através dos testes ao modelo de aprendizagem e das avaliações ao protótipo.

6.4.1 Comparação de modelos de classificação

A comparação do modelo de classificação foi executada consoante o processo denotado na Figura 70. O teste pretende descobrir qual a combinação de aprendizagem que produz melhores resultados, tendo em consideração a mesma base de informação. Devido à modularidade de RASA, é possível criar e testar configurações em simultâneo, definindo os componentes desejados como apresentados na secção 5.2.3.

A base de informação empregue é pequena, composta por 38 intenções e apenas 456 exemplos de texto. É expectável que modelos que utilizem repositórios de linguagem pré-definidos

tenham melhores resultados, contudo, testou-se qualquer vertente. Atendendo que a *framework* se baseia no seu classificador de intenções, o DIET, explicado na secção 2.4.4.1, existe pouca variedade para este componente. Embora se tenham experimentado métodos de regressão linear e *support-vector-machine*, os seus resultados foram fracos e, por consequente, excluídos da análise.

A primeira fase de verificação foi concretizada com o classificador DIET base, testando apenas *tokenizers* e *featurizers*. Para efeitos de visibilidade, a tabela não expõe os seus parâmetros, sendo que estes são sempre iguais:

- Iterações de treino (*Epochs*): 200
- Número de camadas *feed-forward:* 1
- Número de Transformadores: 2

O teste foi efetuado com 5 diferentes configurações, descritas na Tabela 26.

ID	**Configuração**	***Tokenizer***	***Featurizer***
1	`config_bert_base.yml`	Whitespace	Language Model • BERT
2	`config_bert_sparse_features.yml`	Whitespace	Language Model • BERT Lexical Syntatic Count Vectors • Minímo *n-gram*: 1 • Máximo *n-gram*: 4
3	`config_roberta.yml`	Whitespace	Language Model • RoBERTa Lexical Syntatic Count Vectors • Minímo *n-gram*: 1 • Máximo *n-gram*: 4
4	`config_spacy_portuguese.yml`	SpaCy	SpaCyNLP – Dicionário Português Médio Lexical Syntatic Count Vectors • Minímo *n-gram*: 1 • Máximo *n-gram*: 4
5	`config_domain.yml`	Whitespace	Lexical Syntatic Count Vectors • Minímo *n-gram*: 1 • Máximo *n-gram*: 4

Tabela 26 Esquema de aprendizagem para as várias configurações de teste

As configurações 1, 2 e 3 representam a implantação dos modelos de linguagem da Google, o BERT [45] e o RoBERTa [46], que suportam qualquer idioma. Para confirmar se os algoritmos de análise sintática e de contagem de frequência de palavras auxiliam a capacidade de previsão, o primeiro não contém a sua adição.

A configuração 4 suporta a *pipeline* de PLN da biblioteca SpaCy, que tem um fluxo completo já configurado. Para além de um dicionário de dados inicial, baseado em notícias portuguesas [91], é composto por *tokenizers* e um generalizador morfológico, um *lemmatizer*.

Por fim, para verificar se a aprendizagem sem uma base de conhecimento pré-definida é efetiva, denota-se a configuração 5. Esta constrói os rótulos de aprendizagem exclusivamente através dos exemplos fornecidos. Em termos técnicos, o teste é executado com os argumentos demonstrados no Código 21.

```
1. rasa test nlu --nlu data/nlu.yml --config config_bert_base.yml
   config_bert_sparse_features.yml config_domain.yml config_roberta.yml
   config_spacy_portuguese.yml --runs 3 --percentages 0 10 30
```

Código 21. Comando para execução dos testes

Os modelos a analisar são parametrizados através de `--config`, em conjunto com o número de iterações de teste e a percentagem de exclusão de dados. Os resultados consistem em três documentos, nomeadamente um relatório da média de *F1-Score* para cada configuração, a matriz de confusão e, ainda, a listagem dos erros de classificação.

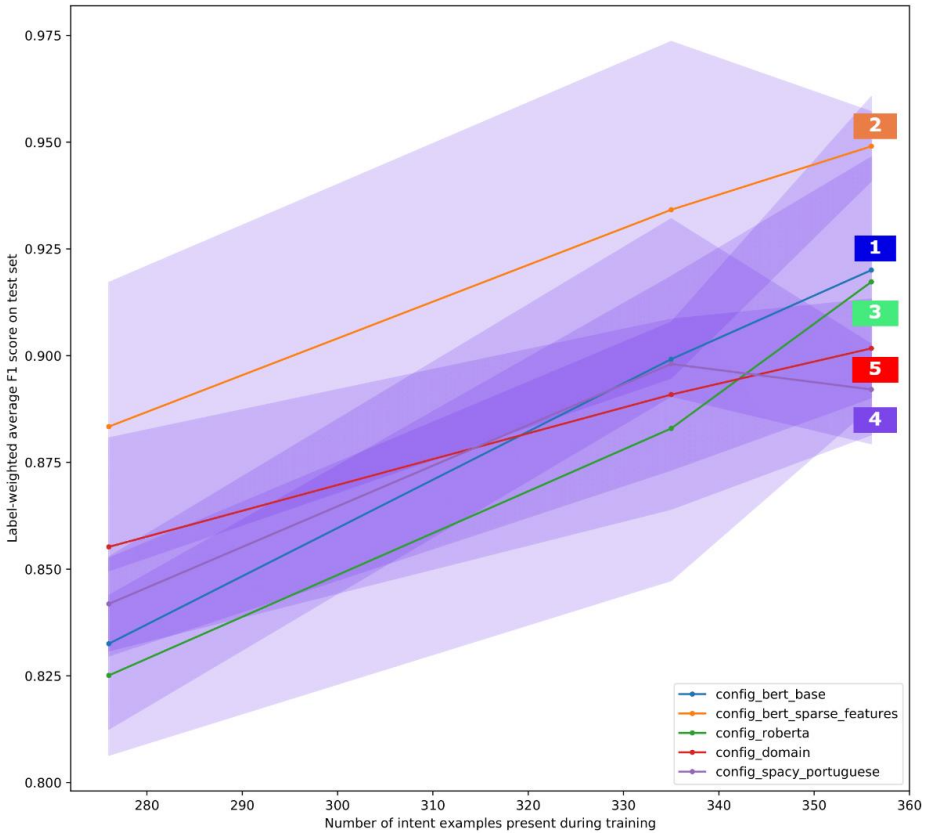

Figura 72. Média de *F1-Score* para as configurações. Os pontos representam a média de cada iteração, enquanto os blocos roxos representam o limite máximo e mínimo das execuções.

A Figura 72 ilustra os resultados obtidos, onde se observa um favoritismo acentuado de BERT. Não só obteve o maior *F1-Score* em todas as iterações, como também evoluiu com a existência

de mais dados de treino. Adicionalmente, entende-se que a utilização de funções sintáticas e de contagem de palavras ajudaram na interpretação de texto, sendo que a configuração 1 classificou-se abaixo da 2.

As restantes configurações obtiveram uma qualificação similar, posicionadas entre os valores de 0.80 a 0.91. O facto de que, à exceção do modelo SpaCy, todas as configurações evoluíram ao longo dos testes, prova que o agente irá beneficiar de um número maior de exemplos de treino. As métricas do melhor teste de cada modelo são visualizadas através da Tabela 27, enquanto as matrizes de confusão estão presentes Anexo D.

ID	Configuração	*Accuracy*	Precisão	*Recall*	*F1-Score*
1	`config_bert_base.yml`	0.891	0.936	0.918	0.901
2	`config_bert_sparse_features.yml`	0.949	0.951	0.956	0.951
3	`config_roberta.yml`	0.929	0.946	0.929	0.924
4	`config_spacy_portuguese.yml`	0.898	0.928	0.898	0.894
5	`config_domain.yml`	0.888	0.917	0.888	0.889

Tabela 27. Exposição das métricas retiradas da matriz de confusão da melhor execução de cada configuração

Embora todos os modelos tenham apresentado resultados positivos, BERT (2) alcança as melhores métricas, com o segundo lugar atribuído a RoBERTa (3).

A próxima fase de verificação incidiu sobre a parametrização do modelo de classificação DIET, utilizando as melhores combinações de preparação de dados delineadas na etapa anterior. O objetivo é aumentar a capacidade de previsão dos modelos, tentando obter resultados superiores em relação aos parâmetros base. As configurações observadas na Tabela 28 foram submetidas tanto para BERT como para RoBERTa.

ID	Configuração	Parametrização
1	`config_bert_diet0`	• Iterações de treino (*Epochs*): 200 • Número de camadas *feed-forward*: *1* • Número de Transformadores: 2
2	`config_roberta_diet0`	
3	`config_bert_diet1`	• Iterações de treino (*Epochs*): 300 • Número de camadas *feed-forward:* 2 • Número de Transformadores: 2
4	`config_roberta_diet1`	
5	`config_bert_diet2`	• Iterações de treino (*Epochs*): 300 • Número de camadas *feed-forward:* 3 • Número de Transformadores: 3
6	`config_roberta_diet2`	

Tabela 28. Parametrização do vários classificadores DIET

A variação recaiu no aumento de transformadores, no número de camadas *feed-forward* e no número de iterações de treino. O relatório obtido é apresentado na Figura 73.

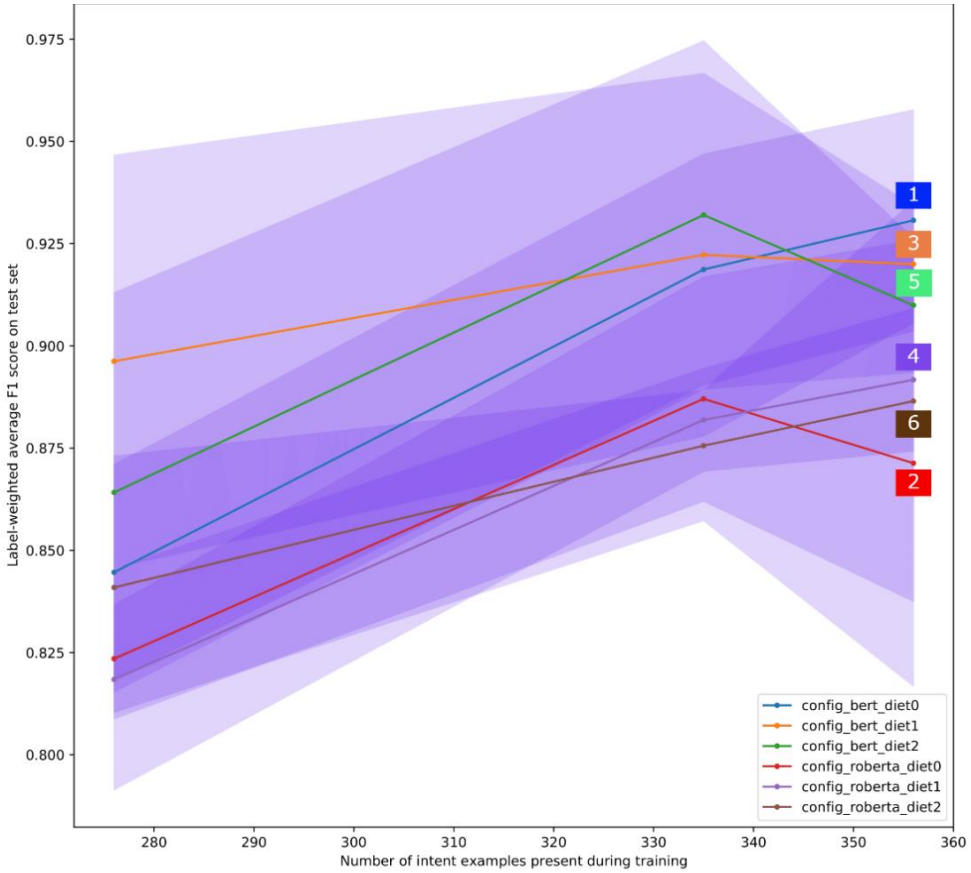

Figura 73. Média de *F1-Score* para os vários classificadores DIET qualificados

Novamente, averigua-se a superioridade do modelo de linguagem BERT (1, 3, 5), que expôs resultados superiores em todas as iterações. Ainda assim, a parametrização de DIET não originou grandes variações na capacidade dos modelos, uma vez que os dois modelos continuaram com *F1-Score* similares à primeira fase.

No que toca ao *F1-Score* máximo, o aumento do número de camadas *feed-forward* e de transformadores conseguiu a melhor qualificação. Contudo, é importante referir que a única configuração que continuou a evoluir em cada iteração foi a 1, com os atributos base. Os exemplos da iteração com melhor desempenho estão na Tabela 29, enquanto as matrizes de confusão estão localizadas no Anexo E.

ID	Configuração	*Accuracy*	Precisão	*Recall*	*F1-Score*
1	`config_bert_diet0.yml`	0.952	0.972	0.969	0.965
2	`config_roberta_diet0.yml`	0.929	0.951	0.939	0.940
3	`config_bert_diet1.yml`	0.939	0.943	0.939	0.938
4	`config_roberta_diet1.yml`	0.919	0.944	0.911	0.914
5	`config_bert_diet2.yml`	0.959	0.960	0.959	0.967
6	`config_roberta_diet2.yml`	0.914	0.937	0.919	0.914

Tabela 29 Exposição das métricas retiradas da matriz de confusão da melhor execução de cada configuração

Tendo em consideração a análise efetuada, a configuração escolhida para o protótipo foi o `config_bert_diet0.yml`. Não só apresenta potencial caso haja expansão de informação, como já apresenta as melhores métricas.

Outro artefacto consequente do teste é a representação da distribuição da confiança de previsões. Existem dois histogramas, apresentando, respetivamente, as previsões corretas e erradas, tendo em consideração a confiança (eixo vertical) e o número de amostras (eixo horizontal). Tal como analisado pela Figura 74, o modelo realizou a maioria dos prognósticos corretos com confiança superior a 70%, obtendo, em contrapartida, três errados.

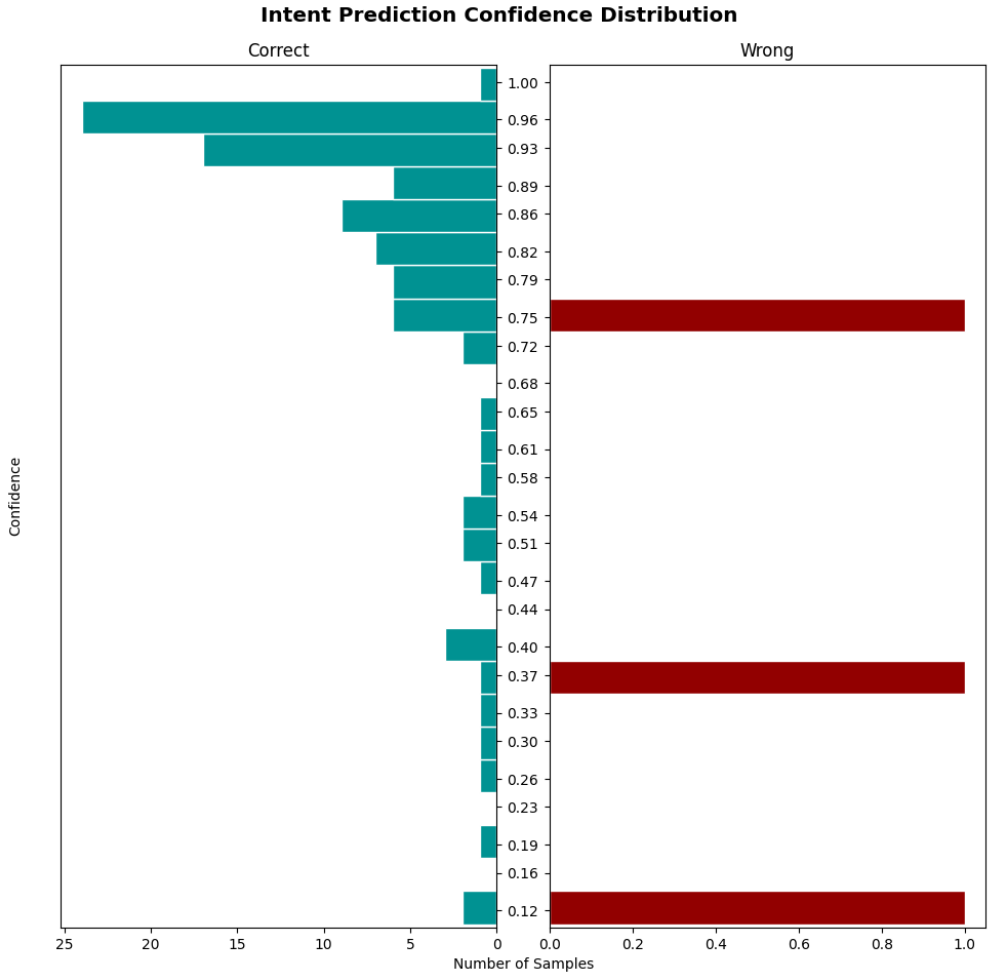

Figura 74. Histograma relativo à capacidade de previsão do modelo escolhido

A previsão relatada pelo agente não implica que esteja correta, daí estar associada a um valor de confiança. Consequentemente, o gráfico é importante para entender qual o limite a utilizar quando se escolhe, ou não, se a resposta associada é enviada ao utilizador. Embora exista um erro acima dos 70%, a maioria das previsões de sucesso foram superiores a este valor, sendo que foi o *threshold* escolhido para a publicação do protótipo.

Em suma, as métricas relativas aos modelos de classificação foram positivas, especificamente a utilização do modelo de linguagem BERT.

6.4.2 Nível de satisfação do utilizador

O armazenamento das interações dos utilizadores permite não só extrair a medida de satisfação dos estudantes, como também verificar quais os tópicos mais frequentes. Ainda que o procedimento de publicação do protótipo não tenha sido o melhor, uma vez que muitos dos estudantes não classificaram as respostas dadas, ainda foi possível obter um número relevante de avaliações. Adicionalmente, o teste não é 100% adequado dado que os utilizadores não tinham propriamente dúvidas, mas sim o intuito de averiguar a utilidade do agente.

O teste foi dividido em duas partes, particularmente a publicação do protótipo para os membros da Divisão Académica e, posteriormente, para os membros do mestrado de Engenharia Informática.

A primeira fase resultou em cerca de 100 interações, onde se verificou uma grande quantidade de assuntos não encontrados. Isto foi resultado de má preparação não só do agente, como da apresentação do protótipo para a Divisão Académica (Figura 75).

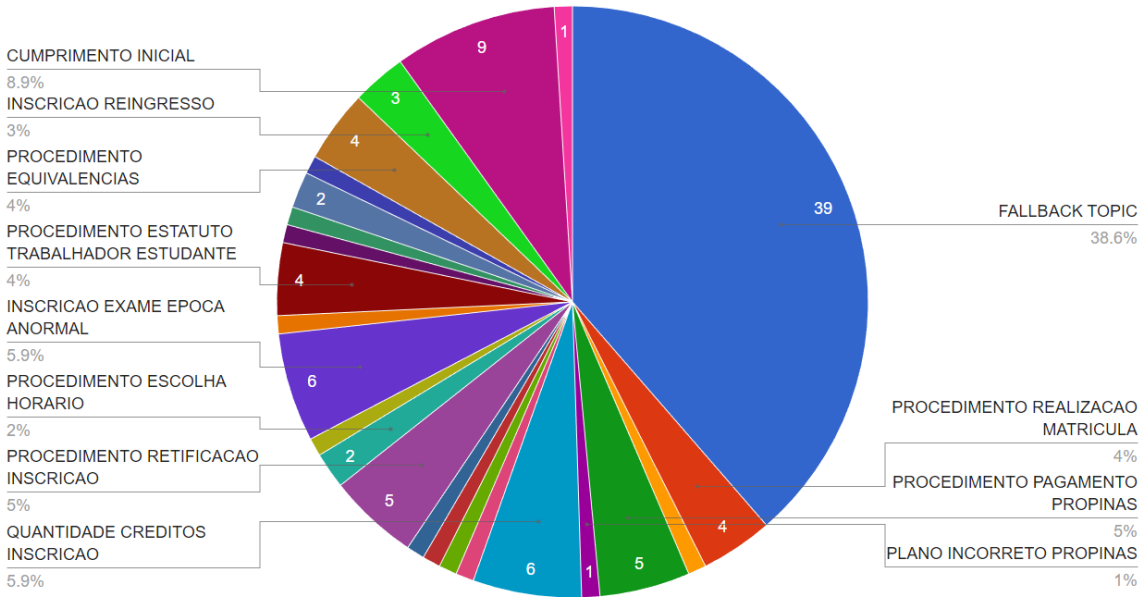

Figura 75. Gráfico da ocorrência de tópicos

No entanto, os restantes tópicos apresentaram um número de ocorrências equilibradas, onde assuntos como estatutos, exames, inscrições e propinas foram os mais frequentes, dado que são as dúvidas mais populares.

O ponto positivo do teste foi, efetivamente, as avaliações dadas pelos configuradores. Como interpretado pela Figura 76, a maioria teve uma média positiva, com um desvio padrão reduzido. No entanto, tópicos como a quantidade de créditos e a anulação do exame apresentam avaliações medíocres, que incentivou a sua mudança. A tabela de ocorrências, médias e desvios está presente no Anexo C.

Figura 76. Avaliações do utilizador segmentadas por intenção (média e desvio padrão)

Para resolver o número de intenções não encontradas, adicionou-se um tópico relativo ao conhecimento do agente. Para além de conseguir explicar perguntas como *"o que sabes? Em que me podes ajudar?"* que foram observadas, possibilita uma experiência mais agradável com o utilizador. Esta atualização, em conjunto com o melhoramento de respostas sugerido pela Divisão Académica, foi implementada antes da segunda fase. Esta exposição originou mais 120 interações, ilustradas na Figura 77.

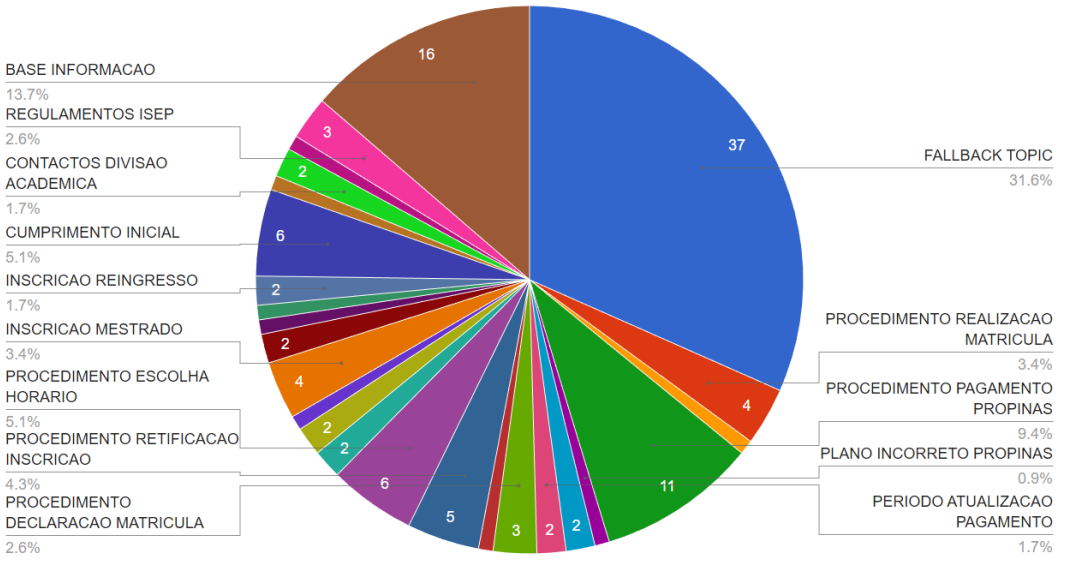

Figura 77. Gráfico da ocorrência de tópicos pertencente à segunda fase

Tal como inicialmente, o tópico base foi o mais predominante, com 31% das interações. No entanto, observou-se uma redução acentuada em comparação aos 38% anteriores, relacionados à adição da intenção da base de informação. Adicionalmente, o efeito deste tópico repara-se especialmente no equilíbrio dos restantes, à exceção do pagamento das propinas.

Por fim, as sugestões da Divisão Académica também foram positivas, uma vez que as respostas obtiveram uma média de avaliações bastante positiva, como exposto na Figura 78. É importante referir que houve tópicos sem qualquer avaliação, como o caso do Ano Zero. Novamente, a tabela de frequências completa está localizada no Anexo C.

Figura 78. Avaliações do utilizador segmentadas por intenção pertencentes à segunda fase (média e desvio padrão)

Em retrospetiva, a publicação do protótipo foi otimista, uma vez que se obteve bons comentários não só por parte dos *stakeholders*, a Divisão Académica, como também avaliações diretas dos estudantes. Contudo, ainda que isto melhore com a longevidade do agente, acentua-se o tamanho reduzido da sua base de conhecimento, visto que várias perguntas não foram compreendidas corretamente.

A média de avaliações global foi de 4.20 com um desvio padrão de 1.23.

7 Conclusão

O projeto desenvolvido teve como objetivo a construção de um agente conversacional totalmente configurável para auxílio da Divisão Académica. O seu planeamento residiu na produção de um sistema de compreensão de linguagem natural, com a capacidade de ser modificado sem conhecimento técnicos. O presente documento expôs o processo de desenvolvimento desde a etapa de investigação, até ao desenho e implementação do agente.

A recolha do estado da arte de sistemas conversacionais ilustrou a sua relevância tanto para organizações como para os seus clientes finais. O estudo permitiu afirmar que o atendimento presencial é cada vez menos eleito em relação ao virtual, sendo que a implementação de um sistema de resposta automática seria uma mais-valia para o ISEP. Acentuam-se benefícios tanto para o estudante, como a centralização do processo de esclarecimento de dúvidas, como para a Divisão Académica e o seu aumento de produtividade.

Para além da análise dos métodos mais relevantes para a compreensão de linguagem natural, elaborou-se uma comparação entre as possíveis ferramentas a utilizar. Consoante a sua escalabilidade, complexidade, custo e as capacidades de aprendizagem, RASA foi a solução mais indicada. É a *framework* que permitiu o maior nível de personalização tanto para o modelo de classificação de intenções, como na preparação dos dados de treino.

A secção de engenharia de requisitos, em conjunção com a análise de valor estruturada, permitiu priorizar os requisitos e construir um conjunto de funcionalidades que compõe o produto mínimo viável, particularmente o módulo de interpretação de linguagem natural e o serviço de gestão de tópicos informativos. Evidencia-se a composição interna da arquitetura sugerida, constituída pelo núcleo do agente conversacional e um serviço *web* para o armazenamento das perguntas e respostas mais frequentes.

Em termos de implementação e experimentação, construiu-se um ecossistema totalmente funcional cujos objetivos são analisados nas seguintes secções.

7.1 Objetivos Concretizados

Face ao planeamento realizado no início do projeto, a aplicação final tem um balanço positivo, sendo que a maioria das metas foram alcançadas:

- **Compreensão da arquitetura convencional de agentes conversacionais**: a análise e avaliação das propriedades referentes a um agente inteligente, nomeadamente as suas características, estratégias de interpretação e geração de texto e, ainda que não utilizado, gestão de contexto. No presente caso, definiu-se um agente interpessoal, informativo, com a habilidade de entender linguagem natural;
- **Criação de um modelo de compreensão de linguagem natural**: investigação sobre os vários métodos de compreensão de texto, nomeadamente técnicas de processamento de linguagem natural e aprendizagem profunda. Com recurso à ferramenta RASA, escolhida após uma comparação com as plataformas mais populares, desenvolveu-se um modelo inteligente capaz de interpretar linguagem portuguesa. Utilizou-se estratégias como *Bag of Words, Pos-Tagging* para a etapa de pré-processamento de dados, modelos de linguagem para a criação de funcionalidades de aprendizagem e, por fim, o classificador DIET para compor as previsões. Os seus resultados foram positivos, com métricas como a precisão, *recall, accuracy*, e *F1-score* acima dos 90%;
- **Criação de mecanismos de configuração do agente:** implementação de um serviço de tópicos acoplado ao agente conversacional, permitindo aos configuradores manipular a sua base de conhecimento sem reformas técnicas. O ecossistema resultante é intuitivo e *user-friendly*;
- **Conversão da informação existente para o agente:** análise e anotação manual do FAQ existente no portal da Divisão Académica, introduzindo esta informação no agente conversacional. O resultado final consistiu em 38 tópicos, constituídos por cerca de 450 exemplos de texto;
- **Publicação do agente aos estudantes:** exposição de um protótipo de teste não só aos membros da Divisão Académica, como aos estudantes do mestrado de Engenharia Informática. Ainda que os resultados apresentem bastantes ocorrências de falha de resposta (30%), as restantes foram bem avaliadas (média: 4.20, desvio-padrão: 1.23);
- **Integração do agente no portal da Escola**: por questões de tempo, o único requisito não cumprido foi, efetivamente, a integração do sistema desenvolvido com o portal da Escola. No entanto, a criação do protótipo ilustra o quão fácil é fazer a conexão do agente a uma interface visual e prepará-lo para interações reais.

Embora os testes tenham sido assertórios, é importante denotar que o comportamento do agente melhora exponencialmente consoante o seu uso. Com a exposição a um maior número de pessoas, o conteúdo tende a desviar do treinado, sendo que terá mais capacidade de interpretação. É de extrema relevância implantar o sistema num ponto público do ISEP, com o intuito de obter o máximo de uso possível.

7.2 Limitações e Ameaças

Embora o sistema se apresente numa versão estável e preparada para a exibição aos estudantes, existem limitações a ter em conta:

- **Restrição de linguagem:** o modelo de classificação apenas é capaz de interpretar a linguagem portuguesa, sendo que estudantes de outras nacionalidades não serão capazes de o usar;
- **Base de conhecimento inicial:** uma limitação ligada ao desenvolvimento do projeto foi a quantidade de exemplos de dúvidas presentes, sendo que os modelos de classificação precisam de largos de corpos de texto para serem eficazes. Ainda assim, devido não só a formulários realizados aos estudantes, como a colaboração da Divisão Académica, conseguiu-se estabelecer um comportamento positivo.

Estas limitações também podem ser vistas como oportunidades, sendo que a restrição de linguagem pode ser facilmente resolvida com um mecanismo intermédio de tradução. De forma complementar, o tamanho da base de conhecimento também aumenta consoante a utilização frequente do sistema.

Por fim, também se denota ameaças que podem deteriorar o sucesso do sistema, particularmente a resistência ao uso do agente ou a má experiência de utilização. Ambos representam a deprecação da aplicação, uma vez que sem exposição a casos reais, a sua evolução é estagnada. Tanto uma como outra retiram o propósito do projeto, a poupança de trabalho à Divisão Académica, uma vez que as vias de esclarecimento utilizadas pelos membros da Escola continuarão a ser as mesmas. Contudo, os testes precoces à sua implantação sugerem o contrário, dado que se obteve avaliações diretas e indiretas positivas.

7.3 Apreciação Pessoal

Várias dificuldades surgiram durante o desenvolvimento do projeto, maioritariamente pela minha experiência reduzida em inteligência artificial. Ainda que a aplicação não seja totalmente baseada em aprendizagem automática, é o seu ponto crucial. Contudo, a área de compreensão de linguagem natural revelou-se bastante interessante, sendo que, aliado ao facto de que o *chatbot* poderia ajudar estudantes como eu, serviu de motivação para continuar.

Em termos técnicos, a dissertação exponenciou acentuadamente o meu conhecimento em termos de inteligência artificial, particularmente técnicas de processamento de linguagem natural e algoritmia de aprendizagem profunda. Apesar de ser abordado em alto nível na licenciatura, esta experiência permitiu aprofundar a área a um nível mais detalhado. Para além disso, introduziu-me a tecnologias e metodologias relevantes no mercado de trabalho atual.

Por fim, a aplicação final encontra-se num ponto positivo, estável, com bastante espaço para expansibilidade, graças aos padrões de engenharia de *software* seguidos. Sinto bastante

orgulho no ecossistema criado e, ainda que não tenha sido integrado no ISEP, espero que seja uma opção futura para ajudar os seus membros.

Referências

[1] A. Huddar, C. Bysani, C. Suchak, U. D. Kolekar, and K. Upadhyaya, “Dexter the College FAQ Chatbot,” in *2020 International Conference on Convergence to Digital World - Quo Vadis (ICCDW)*, 2020, pp. 1–5, doi: 10.1109/ICCDW45521.2020.9318648.

[2] A. Rehan, “10 Best Chatbot Development Frameworks to Build Powerful Bots,” Nov. 24, 2020. https://geekflare.com/chatbot-development-frameworks/ (accessed Dec. 29, 2021).

[3] S. Han and M. K. Lee, “FAQ chatbot and inclusive learning in massive open online courses,” *Comput. Educ.*, vol. 179, p. 104395, 2022, doi: https://doi.org/10.1016/j.compedu.2021.104395.

[4] T. Wambsganss, R. Winkler, M. Söllner, and J. M. Leimeister, “A Conversational Agent to Improve Response Quality in Course Evaluations,” in *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–9, doi: 10.1145/3334480.3382805.

[5] A. Shevat, *Designing bots: Creating conversational experiences*. O’Reilly, 2017.

[6] J. Mayo, *Programming the Microsoft BOT framework: A multiplatform approach to Building Chatbots*. Microsoft, 2018.

[7] J. Weizenbaum, “ELIZA—a Computer Program for the Study of Natural Language Communication between Man and Machine,” *Commun. ACM*, vol. 9, no. 1, pp. 36–45, Jan. 1966, doi: 10.1145/365153.365168.

[8] R. S. Wallace, “The Anatomy of A.L.I.C.E.,” in *Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer*, R. Epstein, G. Roberts, and G. Beber, Eds. Dordrecht: Springer Netherlands, 2009, pp. 181–210.

[9] F. Promoteur and L. Facult, “University of Liège - Faculty of Applied Sciences Master thesis Design and implementation of a chatbot in the context of customer support Graduation Studies conducted for obtaining the Master ’ s degree in,” p. 77, 2018.

[10] J. da S. Oliveira, D. B. Espíndola, R. Barwaldt, L. M. Ribeiro, and M. Pias, “IBM Watson Application as FAQ Assistant about Moodle,” in *2019 IEEE Frontiers in Education Conference (FIE)*, 2019, pp. 1–8, doi: 10.1109/FIE43999.2019.9028667.

[11] “Gartner,” “Gartner Customer 360 Summit 2011: CRM Strategies and Technologies to Understand, Grow and Manage Customer Experiences,” p. 2, 2011, [Online]. Available: https://www.gartner.com/imagesrv/summits/docs/na/customer-360/C360_2011_brochure_FINAL.pdf.

[12] “80% of Businesses Want Chatbots by 2020.” https://www.businessinsider.com/80-of-businesses-want-chatbots-by-2020-2016-12 (accessed Jan. 03, 2022).

[13] P. Gentsch, “Best Practices,” in *Künstliche Intelligenz für Sales, Marketing und Service: Mit AI und Bots zu einem Algorithmic Business -- Konzepte, Technologien und Best Practices*, Wiesbaden: Springer Fachmedien Wiesbaden, 2018, pp. 117–220.

[14] “Messaging with Businesses is Changing how People Shop Online | Facebook IQ | Facebook for Business.” https://www.facebook.com/business/news/insights/3-ways-messaging-is-transforming-the-path-to-purchase (accessed Jan. 03, 2022).

[15] T. D. Rieke, “The relationship between motives for using a Chatbot and satisfaction with Chatbot characteristics in the Portuguese Millennial population: an exploratory study,” pp. 4–12, 2018.

[16] K. Nimavat and T. Champaneria, “Chatbots: An overview. Types, Architecture, Tools and Future Possibilities,” 2017.

[17] P. Kucherbaev, A. Bozzon, and G.-J. Houben, “Human-Aided Bots,” *IEEE Internet Comput.*, vol. 22, no. 6, pp. 36–43, 2018, doi: 10.1109/MIC.2018.252095348.

[18] H. T. Hien, P.-N. Cuong, L. N. H. Nam, H. L. T. K. Nhung, and L. D. Thang, “Intelligent Assistants in Higher-Education Environments: The FIT-EBot, a Chatbot for Administrative and Learning Support,” in *Proceedings of the Ninth International Symposium on Information and Communication Technology*, 2018, pp. 69–76, doi: 10.1145/3287921.3287937.

[19] R. Dobbs, J. Manyika, and J. Woetzel, “Global growth: Can productivity save the day in an aging world ?,” *McKinsey Glob. Inst.*, no. January, p. 148, 2015.

[20] J. Cahn *et al.*, “Chatbot: architecture, design & development,” *Sci. Rep.*, vol. 19, no. 1, p. 46, 2017, [Online]. Available: https://www.academia.edu/attachments/57035006/download_file%0Ahttps://doi.org/10.1038/s41598-020-68764-y%0Ahttp://dx.doi.org/10.1038/s41467-020-18349-0%0Ahttp://dx.doi.org/10.1038/s41467-019-13807-w%0Ahttps://www.academia.edu/download/57035006/CHATBOT_the.

[21] M. Mohammed and M. M. Aref, “Chatbot System Architecture.” 2022.

[22] E. D. Liddy, “SURFACE SURFACE Center for Natural Language Processing School of Information Studies (iSchool) 2001 Natural Language Processing Natural Language Processing Natural Language Processing 1,” 2001, [Online]. Available: https://surface.syr.edu/cnlp.

[23] J. F. Allen, “Natural Language Processing,” in *Encyclopedia of Computer Science*, GBR: John Wiley and Sons Ltd., 2003, pp. 1218–1222.

[24] A. Heikkilä, “Natural Language Processing Techniques in Chatbot Development : How Does a Chatbot Process Language?,” pp. 1–72, 2020.

[25] D. Braun, A. Hernandez Mendez, F. Matthes, and M. Langen, “Evaluating Natural Language Understanding Services for Conversational Question Answering Systems,” in *Proceedings of the 18th Annual {SIG}dial Meeting on Discourse and Dialogue*, 2017, pp. 174–185, doi: 10.18653/v1/W17-5522.

[26] J. Allen and M. Core, “Draft of {DAMSL}: Dialog Act Markup in Several Layers,” 1997.

[27] D. Goddeau, H. Meng, J. Polifroni, S. Seneff, and S. Busayapongchai, “A form-based dialogue manager for spoken language applications,” in *Proceeding of Fourth International Conference on Spoken Language Processing. ICSLP ’96*, 1996, vol. 2, pp. 701–704 vol.2, doi: 10.1109/ICSLP.1996.607458.

[28] J. D. Williams, A. Raux, and M. Henderson, “The Dialog State Tracking Challenge Series: A Review,” *Dialogue & Discourse*, vol. 7, no. 3, pp. 4–33, 2016, doi: 10.5087/dad.2016.301.

[29] J. Masche and N.-T. Le, “A Review of Technologies for Conversational Systems,” 2018, pp. 212–225, doi: 10.1007/978-3-319-61911-8_19.

[30] Y. Jiang Zhao, Y. Ling Li, and M. Lin, “A Review of the Research on Dialogue Management of Task-Oriented Systems,” *J. Phys. Conf. Ser.*, vol. 1267, no. 1, 2019, doi: 10.1088/1742-6596/1267/1/012025.

[31] Z. Yan *et al.*, “{D}oc{C}hat: An Information Retrieval Approach for Chatbot Engines Using Unstructured Documents,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 516–525, doi: 10.18653/v1/P16-1049.

[32] Y. Song, C. Te Li, J. Y. Nie, M. Zhang, D. Zhao, and R. Yan, “An ensemble of retrieval-based and generation-based human-computer conversation systems,” *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 2018-July, pp. 4382–4388, 2018, doi: 10.24963/ijcai.2018/609.

[33] K. Palasundram, N. M. Sharef, N. A. Nasharuddin, K. A. Kasmiran, and A. Azman, “Sequence to sequence model performance for education chatbot,” *Int. J. Emerg. Technol. Learn.*, vol. 14, no. 24, pp. 56–68, 2019, doi: 10.3991/ijet.v14i24.12187.

[34] I. El Naqa and M. J. Murphy, “What Is Machine Learning?,” in *Machine Learning in Radiation Oncology: Theory and Applications*, I. El Naqa, R. Li, and M. J. Murphy, Eds. Cham: Springer International Publishing, 2015, pp. 3–11.

[35] T. M. Mitchell, *Machine Learning*, 1st ed. USA: McGraw-Hill, Inc., 1997.

[36] G. Bonaccorso, *Machine learning algorithms: Popular algorithms for Data Science and Machine Learning*. Packt, 2018.

[37] V. Nasteski, “An overview of the supervised machine learning methods,” *Horizons.B*, vol. 4, no. December 2017, pp. 51–62, 2017, doi: 10.20544/horizons.b.04.1.17.p05.

[38] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. The MIT Press, 2018.

[39] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[40] F. Bre, J. Gimenez, and V. Fachinotti, “Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks,” *Energy Build.*, vol. 158, 2017, doi: 10.1016/j.enbuild.2017.11.045.

[41] G. Keilbar, “Modelling Systemic Risk using Neural Network Quantile Regression,” 2018.

[42] A. Graves and J. Schmidhuber, “Framewise phoneme classification with bidirectional LSTM networks,” in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, 2005, vol. 4, pp. 2047–2052 vol. 4, doi: 10.1109/IJCNN.2005.1556215.

[43] C. Nicholson, “A Beginner’s Guide to LSTMs and Recurrent Neural Networks | Pathmind.” https://wiki.pathmind.com/lstm (accessed Jan. 15, 2022).

[44] A. Vaswani *et al.*, “Attention Is All You Need.” arXiv, 2017, doi: 10.48550/ARXIV.1706.03762.

[45] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” arXiv, 2018, doi: 10.48550/ARXIV.1810.04805.

[46] Y. Liu *et al.*, “RoBERTa: A Robustly Optimized BERT Pretraining Approach.” arXiv, 2019, doi: 10.48550/ARXIV.1907.11692.

[47] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners,” 2019.

[48] S. T. Dumais, “Latent semantic analysis,” *Annu. Rev. Inf. Sci. Technol.*, vol. 38, no. 1, pp. 188–230, 2004, doi: https://doi.org/10.1002/aris.1440380105.

[49] G. S. Kim, “Text Mining – Topic Modeling – Gyungsun Sonny Kim’s blog.” https://gyungsunsonnykim.wordpress.com/2019/01/30/plsa-lda/ (accessed Jan. 15, 2022).

[50] J.-T. Hsieh, C. Li, and W. Liu, “Effective Word Representation for Named Entity Recognition,” pp. 1–11, 2017.

[51] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient Estimation of Word Representations in Vector Space.” [Online]. Available: http://ronan.collobert.com/senna/.

[52] “Word Mover’s Embedding: Universal Text Embedding from Word2Vec | IBM Research Blog.” https://www.ibm.com/blogs/research/2018/11/word-movers-embedding/ (accessed Jan. 25, 2022).

[53] T. Bunk, D. Varshneya, V. Vlasov, and A. Nichol, “DIET: Lightweight Language Understanding for Dialogue Systems.” arXiv, 2020, doi: 10.48550/ARXIV.2004.09936.

[54] “Aqui o trabalho acontece | Slack.” https://slack.com/intl/pt-pt/ (accessed Oct. 30, 2021).

[55] M. Nilsson and D. Dunér, “Scalability of push and pull based event notification,” 2020.

[56] “14 most powerful platforms to build a Chatbot [November 2020 Update].” https://marutitech.com/14-powerful-chatbot-platforms/ (accessed Jan. 18, 2022).

[57] “No Code Chatbot Platform | Free Chatbot Platform | WotNot.” https://wotnot.io/ (accessed Jun. 26, 2022).

[58] “Intuitive Conversational Chatbot Builder | Landbot.io.” https://landbot.io/ (accessed Jan. 20, 2022).

[59] S. A. Thorat and V. Jadhav, “A Review on Implementation Issues of Rule-based Chatbot Systems,” *SSRN Electron. J.*, no. Icicc, pp. 1–6, 2020, doi: 10.2139/ssrn.3567047.

[60] M. Canonico and L. De Russis, “A Comparison and Critique of Natural Language Understanding Tools,” 2018.

[61] “Documentação do Dialogflow | Google Cloud.” https://cloud.google.com/dialogflow/docs (accessed Jan. 25, 2022).

[62] “IBM Watson.” https://cloud.ibm.com/developer/watson/documentation (accessed Jan. 25, 2022).

[63] H. Qi *et al.*, “Benchmarking Commercial Intent Detection Services with Practice-Driven Evaluations,” pp. 304–310, 2021, doi: 10.18653/v1/2021.naacl-industry.38.

[64] “Rasa: Developer Documentation Portal | Rasa.” https://rasa.com/docs/ (accessed Jan. 25, 2022).

[65] “spaCy · Industrial-strength Natural Language Processing in Python.” https://spacy.io/ (accessed Jan. 25, 2022).

[66] “TensorFlow.” https://www.tensorflow.org/ (accessed Jan. 25, 2022).

[67] “Keras: the Python deep learning API.” https://keras.io/ (accessed Jan. 25, 2022).

[68] P. Muangkammuen, “Automated Thai-FAQ Chatbot using RNN-LSTM,” pp. 1–4, 2018.

[69] P. Belliveau, A. Griffin, and S. Somermeyer, *The PDMA toolbook 1 for new product development*. John Wiley & Sons, 2002.

[70] P. Koen *et al.*, “Providing clarity and a common language to the ‘fuzzy front end,’” *Res. Technol. Manag.*, vol. 44, no. 2, pp. 46–55, 2001, doi: 10.1080/08956308.2001.11671418.

[71] C. Namugenyi, S. L. Nimmagadda, and T. Reiners, “SWOT ANALYSIS : A THEORETICAL REVIEW Design of a SWOT Analysis Model and its Evaluation in Diverse Design of a SWOT Analysis Model and its Evaluation in Diverse,” *Procedia Comput. Sci.*, vol. 159, no. 2019, pp. 1145–1154, 2017.

[72] H. S. Neap and T. Celik, “Value of a Product: A Definition,” *Int. J. Value-Based Manag.*, vol. 12, no. 2, pp. 181–191, 1999, doi: 10.1023/A:1007718715162.

[73] L. D. Miles, *Techniques of value analysis and Engineering*. Eleanor Miles Walker, 1989.

[74] A. Osterwalder and Y. Pigneur, “An Ontology For E-Business Models,” in *Value Creation*

from E-Business Models, 2004.

[75] S. Nicola, “Análise de Valor.”

[76] J. Buchanan and L. Gardiner, “A comparison of two reference point methods in multiple objective mathematical programming,” *Eur. J. Oper. Res.*, vol. 149, no. 1, pp. 17–34, 2003, [Online]. Available: https://ideas.repec.org/a/eee/ejores/v149y2003i1p17-34.html.

[77] T. L. Saaty, “How to make a decision: The analytic hierarchy process,” *Eur. J. Oper. Res.*, vol. 48, no. 1, pp. 9–26, 1990, doi: https://doi.org/10.1016/0377-2217(90)90057-I.

[78] S. Nicola, “Análise de Valor AHP.”

[79] S. Easterbrook, “What is Requirements Engineering ?,” *Requir. Eng.*, pp. 2–18, 2004.

[80] A. S. for P. Affairs, “Use Cases,” Oct. 2013.

[81] A. Avram, F. Marinescu, D. B. Johnsson, V. Gitlevich, and E. Evans, *Domain-driven design quickly: A summary of Eric Evans’ domain-driven design*. C4Media, 2006.

[82] M. Fowler, *Patterns of enterprise application architecture*. Addison-Wesley, 2002.

[83] S. Brown, “Visualising software architecture,” *Softw. Archit. Dev.*, p. 4, 2014.

[84] M. Richards, *Software Architecture Patterns*. O’Reilly Media, Inc., 2015.

[85] “Your relational data. Objectively. - Hibernate ORM.” https://hibernate.org/orm/ (accessed Jun. 20, 2020).

[86] “NLU Training Data.” https://rasa.com/docs/rasa/nlu-training-data/ (accessed Jun. 13, 2022).

[87] “React – A JavaScript library for building user interfaces.” https://reactjs.org/ (accessed Jun. 13, 2022).

[88] “Conhecer o Azure | Microsoft Azure.” https://azure.microsoft.com/pt-pt/overview/ (accessed Jun. 20, 2020).

[89] “Introduction to Node.js.” https://nodejs.dev/learn (accessed Jun. 02, 2020).

[90] W. Maroengsit, T. Piyakulpinyo, K. Phonyiam, S. Pongnumkul, P. Chaovalit, and T. Theeramunkong, “A Survey on Evaluation Methods for Chatbots,” in *Proceedings of the 2019 7th International Conference on Information and Education Technology*, 2019, pp. 111–119, doi: 10.1145/3323771.3323824.

[91] A. Rademaker, F. Chalub, L. Real, C. Freitas, E. Bick, and V. de Paiva, “Universal Dependencies for Portuguese,” in *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling)*, 2017, pp. 197–206.

Anexo A Formulário do Agente

Esqueleto do formulário desenvolvido e as respostas de um utilizador. No total, obtiveram-se 19 respostas, das quais originaram 127 novos exemplos de teste.

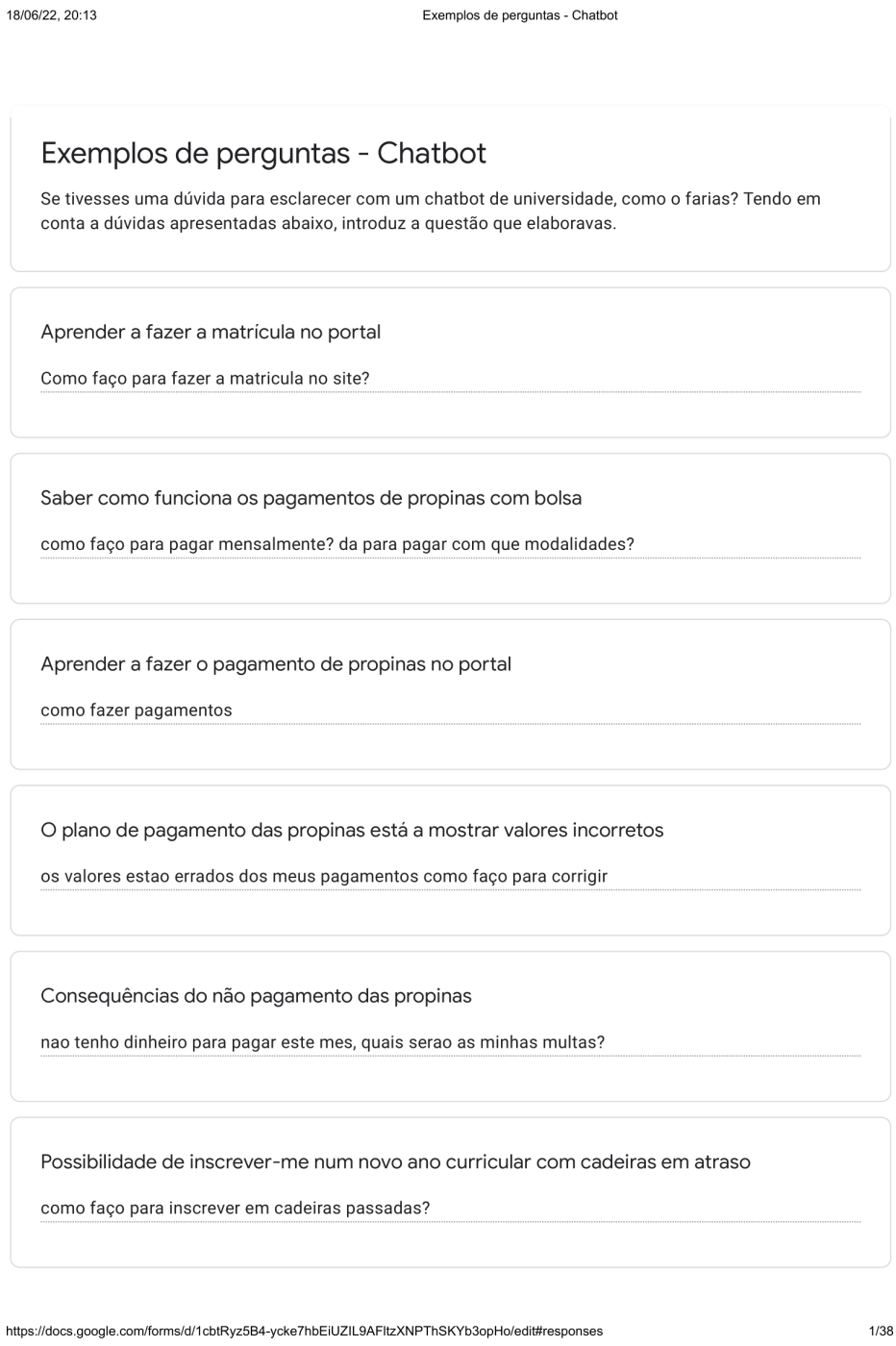

Figura 79. Página 1 do formulário relativo às dúvidas dos estudantes

Saber quanto tempo as propinas demoram a atualizar

ja passa uns dias do inicio do mes e as propinas ainda nao atualizaram. quanto tempo demora para o fazerem?

Como obter a declaração de matrícula no portal

como obtenho a matricula para receber o subsidio de transporte

Como obter a declaração de horário no portal

como recebo a declaração de horário para entregar na entidade patronal

Como obter a declaração de passe sub23 no portal

Como anular a matrícula

como posso cancelar a matricula

Fazer a escolha do horário

dá para fazer um horario com a minha disponiblidade?

Figura 80. Página 2 do formulário relativo às dúvidas dos estudantes

Anexo B Base de Conhecimento

Base de conhecimento do agente, expostos na Tabela 30.

Intenção	Descrição	Categoria
PROCEDIMENTO_REALIZACAO_MATRICULA	Como realizar a matrícula no portal	INSCRICAO
BOLSA_PAGAMENTO_PROPINAS	Pagamento das propinas aquando de uma candidatura de bolsa	PROPINAS
PROCEDIMENTO_PAGAMENTO_PROPINAS	Como pagar propinas portal	PROPINAS
PROCEDIMENTO_PAGAMENTO_PROPINAS	Instruções caso o plano de propinas estiver incorreto	PROPINAS
ATRASO_RENOVACAO_INSCRICAO	Renovação de inscrição com cadeiras em atraso	INSCRICAO
QUANTIDADE_CREDITOS_INSCRICAO	Informações relativas ao uso de créditos na inscrição	INSCRICAO
PERIODO_ATUALIZACAO_PAGAMENTO	Duração de atualização do estado de pagamento portal	PROPINAS
PROCEDIMENTO_DECLARACAO_MATRICULA	Como recolher a certidão de matrícula no portal	INSCRICAO
PROCEDIMENTO_DECLARACAO_SUB23	Como recolher a declaração do passe sub 23 no portal	PASSE_SUB23
PROCEDIMENTO_DECLARACAO_HORARIO	Como recolher a declaração de horário no portal	HORARIO
PROCEDIMENTO_RETIFICACAO_INSCRICAO	Como fazer a retificação de inscrição no portal	INSCRICAO
PROCEDIMENTO_ESCOLHA_HORARIO	Como escolher o horário no portal	HORARIO
INSCRICAO_EXAME_EPOCA_NORMAL	Como funciona a inscrição em época normal	EXAME
INSCRICAO_EXAME_EPOCA_ANORMAL	Como funciona a inscrição em época de recurso e especial	EXAME
INSCRICAO_EXAME_MELHORIA	Como funciona a inscrição em melhoria	EXAME
ESTATUTO_TRABALHADOR_ESTUDANTE	Como obter estatuto trabalhador estudante	ESTATUTOS
PROCEDIMENTO_RETIFICACAO_EXAME	Possibilidade de anular a inscrição em exames	EXAME
PROCEDIMENTO_FALTAS_COVID	Justificação de faltas de COVID	FALTAS
INSCRICAO_ANO_ZERO	Informações sobre o ano zero	ANO_ZERO
INSCRICAO_MESTRADO	Informações sobre inscrição em mestrados	INSCRICAO
INSCRICAO_MAIS_23_ANOS	Candidaturas 23 anos	INSCRICAO
INSCRICAO_CTESP	Condições de inscrição para CTESPS	INSCRICAO

Intenção	Descrição	Categoria
PROCEDIMENTO_MUDANCA_REGIMES	Instruções para mudanças de regimes	REGIME
INFORMACOES_CARTAO_ISEP	Instruções para o cartão do ISEP	CARTAO
INSCRICAO_UCS_ISOLADAS	Instruções sobre unidades curriculares isoladas	INSCRICAO
PROCEDIMENTO_EQUIVALENCIAS	Como pedir equivalências	EQUIVALENCIAS
INSCRICAO_REINGRESSO	Instruções sobre reingresso	INSCRICAO
CUMPRIMENTO_INICIAL	Cumprimento inicial	CUMPRIMENTOS
DESPEDIDA	Despedida	CUMPRIMENTOS
NUMERO_FALTAS	Quantidade de faltas que um estudante pode dar	FALTAS
MANTER_NOTA_FREQUENCIA	Processo necessário para manter a nota de frequência	INSCRICAO
INSCRICAO_LICENCIATURA	Informações sobre inscrições em Licenciaturas	INSCRICAO
CONTACTOS_DIVISAO_ACADEMICA	Lista de contactos para a Divisão Académica	CONTACTOS
TRANSICAO_ANO	Condições para transição de ano	INSCRICAO
REGULAMENTOS_ISEP	Local dos regulamentos do ISEP	CONTACTOS
CONCLUSAO_CURSO	Normas de conclusão de curso	INSCRICAO
BASE_INFORMACAO	Conhecimento do agente	CONTACTOS

Tabela 30. Base de conhecimento do agente final tabelados por tópico e categoria

Lista de sinónimos no ficheiro `nlu.yml`, identificado no Código 22.

```
- synonym: matrícula
    examples: |
      - inscrição
  - synonym: universidade
    examples: |
      - faculdade
      - instituto
      - escola
      - instituição
  - synonym: portal
    examples: |
      - site
      - online
  - synonym: cadeira
    examples: |
      - unidade curricular
      - disciplina
      - UC
  - synonym: CTESP
    examples : |
      - CTeSP
      - cursos técnicos profissionais
      - cursos técnicos superiores profissionais
```

Código 22. Lista de sinónimos configurados

Anexo C Dados relativos à utilização do protótipo

Tabela de frequência absoluta relativa ao teste da Divisão Académica (Tabela 31).

Tópico	Ocorrências	Média	Desvio Padrão
FALLBACK_TOPIC	39	0	0
PROCEDIMENTO_REALIZACAO_MATRICULA	4	3.75	1.64
BOLSA_PAGAMENTO_PROPINAS	1	4	0
PROCEDIMENTO_PAGAMENTO_PROPINAS	5	5	0
PLANO_INCORRETO_PROPINAS	1	5	0
QUANTIDADE_CREDITOS_INSCRICAO	6	2.17	1.67
PERIODO_ATUALIZACAO_PAGAMENTO	1	5	0
PROCEDIMENTO_DECLARACAO_MATRICULA	1	5	0
PROCEDIMENTO_DECLARACAO_SUB23	1	5	0
PROCEDIMENTO_DECLARACAO_HORARIO	1	5	0
PROCEDIMENTO_RETIFICACAO_INSCRICAO	5	2	1.00
PROCEDIMENTO_ESCOLHA_HORARIO	2	5	0
INSCRICAO_EXAME_EPOCA_NORMAL	1	5	0
INSCRICAO_EXAME_EPOCA_ANORMAL	6	4.2	1.60
INSCRICAO_EXAME_MELHORIA	1	5	0
PROCEDIMENTO_ESTATUTO_TRABALHADOR_ESTUDANTE	4	5	0
PROCEDIMENTO_FALTAS_COVID	1	5	0
INSCRICAO_ANO_ZERO	1	5	0
INSCRICAO_MESTRADO	2	4	1.00
INFORMACOES_CARTAO_ISEP	1	5	0
PROCEDIMENTO_EQUIVALENCIAS	4	5	0

INSCRICAO_REINGRESSO	3	4.67	0.47
CUMPRIMENTO_INICIAL	9	0	0
NUMERO_FALTAS	1	5	0

Tabela 31. Tabela de frequência da primeira fase de testes

Tabela de frequência absoluta relativa ao teste dos estudantes do Mestrado de Engenharia Informática (Tabela 32).

Tópico	**Ocorrências**	**Média**	**Desvio Padrão**
FALLBACK_TOPIC	37	0	0
PROCEDIMENTO_REALIZACAO_MATRICULA	4	4.25	0.83
BOLSA_PAGAMENTO_PROPINAS	1	5	0
PROCEDIMENTO_PAGAMENTO_PROPINAS	11	4.75	0.66
PLANO_INCORRETO_PROPINAS	1	5	0
QUANTIDADE_CREDITOS_INSCRICAO	2	4	1
PERIODO_ATUALIZACAO_PAGAMENTO	2	5	0
PROCEDIMENTO_DECLARACAO_MATRICULA	3	4.67	0.47
PROCEDIMENTO_DECLARACAO_HORARIO	1	5	0
PROCEDIMENTO_RETIFICACAO_INSCRICAO	5	4.6	0.8
PROCEDIMENTO_ESCOLHA_HORARIO	6	4.75	0.43
PROCEDIMENTO_ESTATUTO_TRABALHADOR_ESTUDANTE	2	5	0
PROCEDIMENTO_RETIFICACAO_EXAME	2	4.5	0.5
INSCRICAO_ANO_ZERO	1	0	0
INSCRICAO_MESTRADO	4	3.67	0.94
INSCRICAO_MAIS_23_ANOS	2	5	0
INSCRICAO_CTESP	1	4	0
INFORMACOES_CARTAO_ISEP	1	4	0

INSCRICAO_REINGRESSO	2	5	0
CUMPRIMENTO_INICIAL	6	0	0
DESPEDIDA	1	0	0
CONTACTOS_DIVISAO_ACADEMICA	2	5	0
TRANSICAO_ANO	1	3	0
REGULAMENTOS_ISEP	3	5	0
BASE_INFORMACAO	16	4.67	0.47

Tabela 32. Tabela de frequência da segunda fase de testes

Anexo D Dados relativos ao estudo dos modelos de classificação (1ª fase)

O presente anexo mostra as matrizes de confusão e os diagramas de distribuição de confiança para a melhor iteração de cada modelo de classificação.

1. Configuração `config_bert_base.yml`

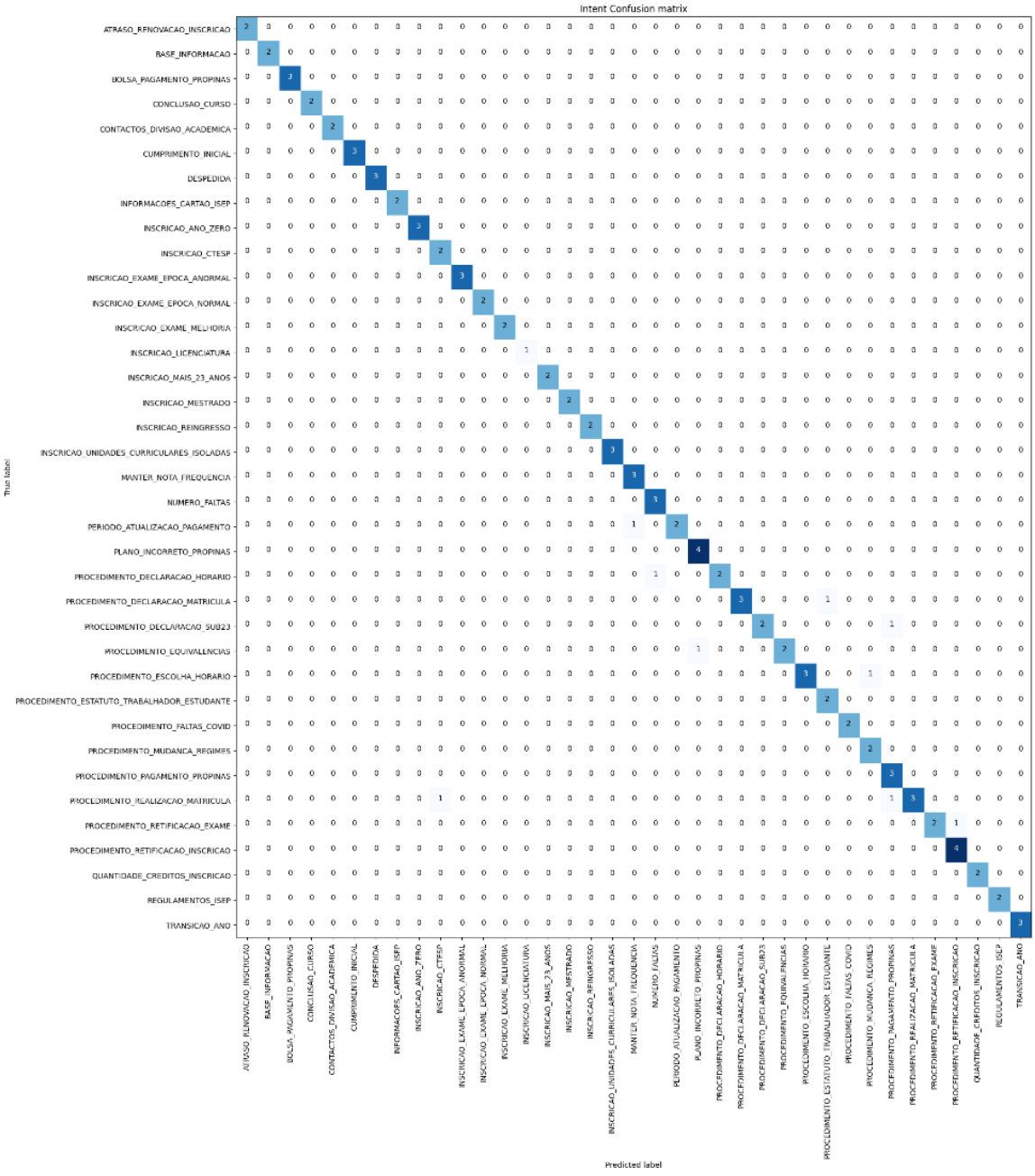

Figura 81. Matriz de confusão para a `config_bert_base.yml`

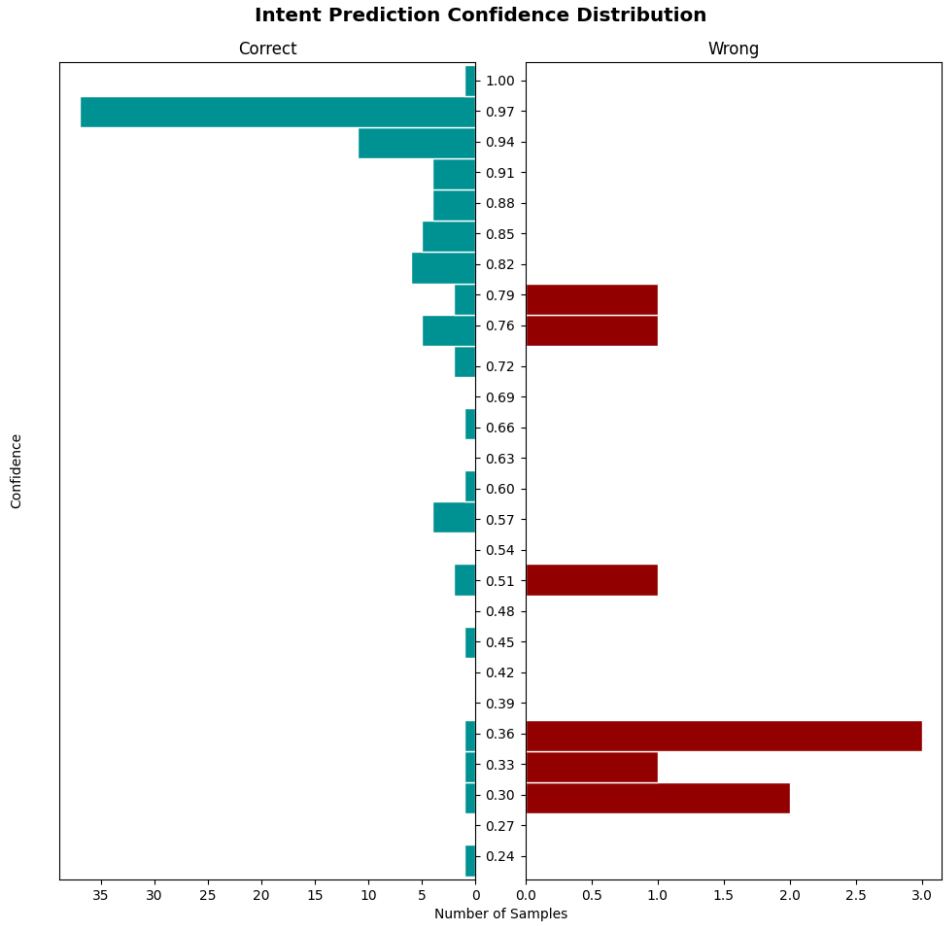

Figura 82. Diagrama de distribuição de confiança para a `config_bert_base.yml`

2. Configuração `config_bert_sparse_features.yml`

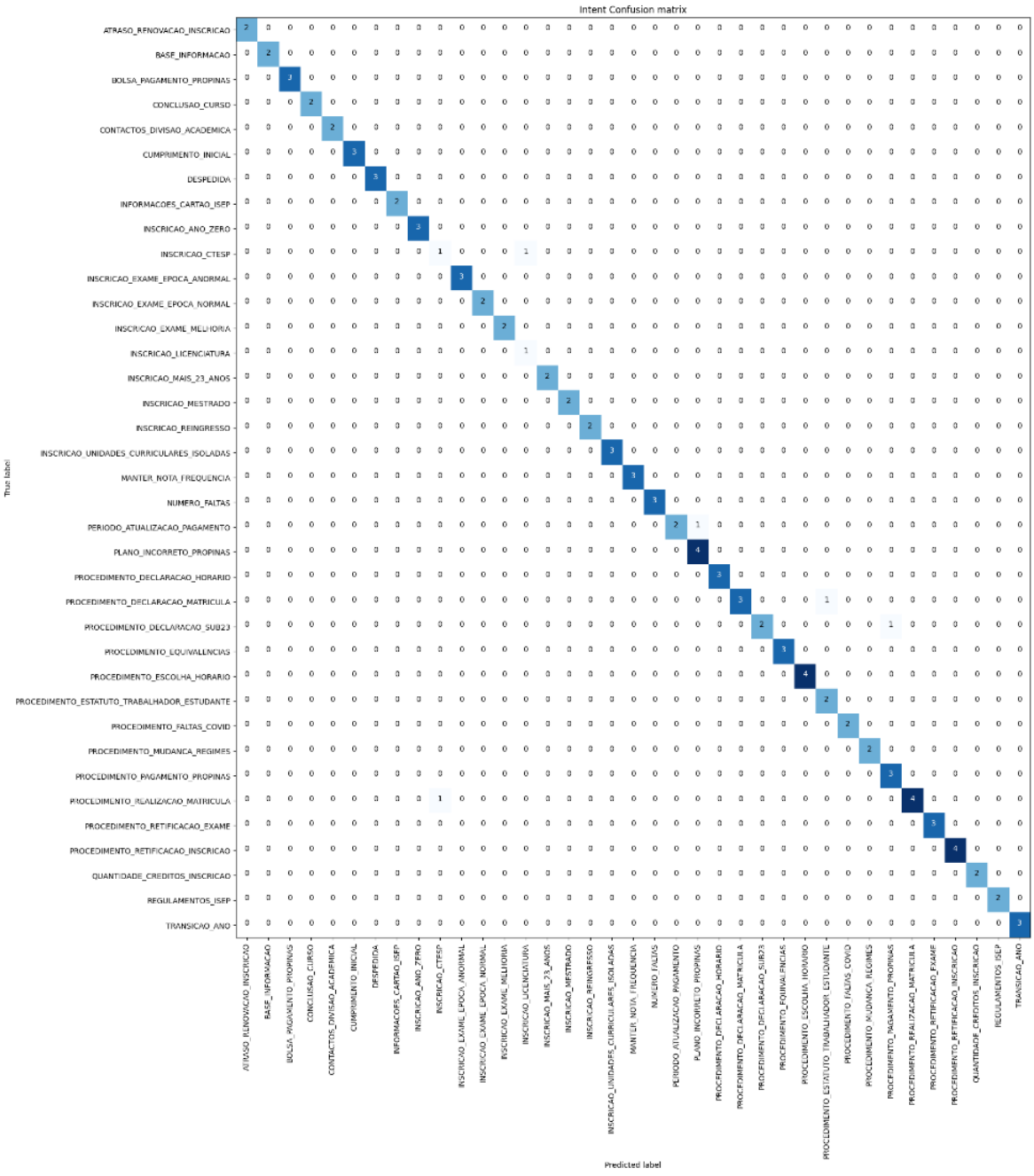

Figura 83. Matriz de confusão para a `config_bert_sparse_features.yml`

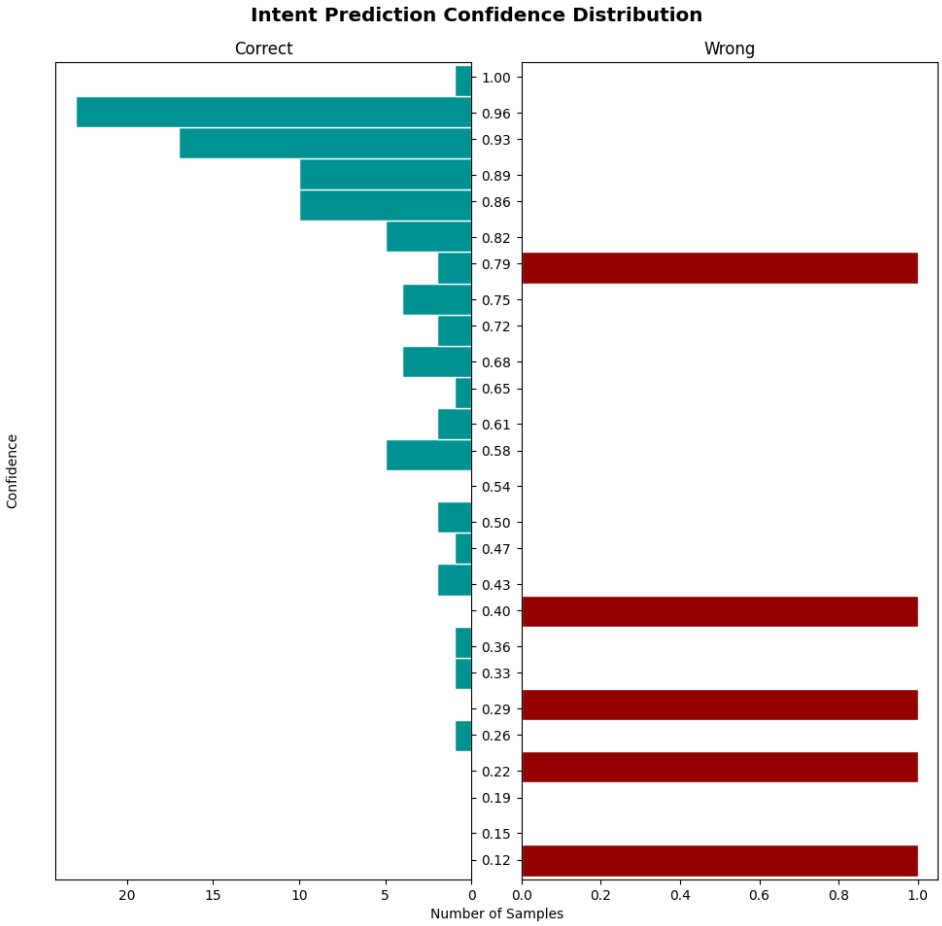

Figura 84. Diagrama de distribuição de confiança para a `config_bert_sparse_features.yml`

3. Configuração `config_roberta.yml`

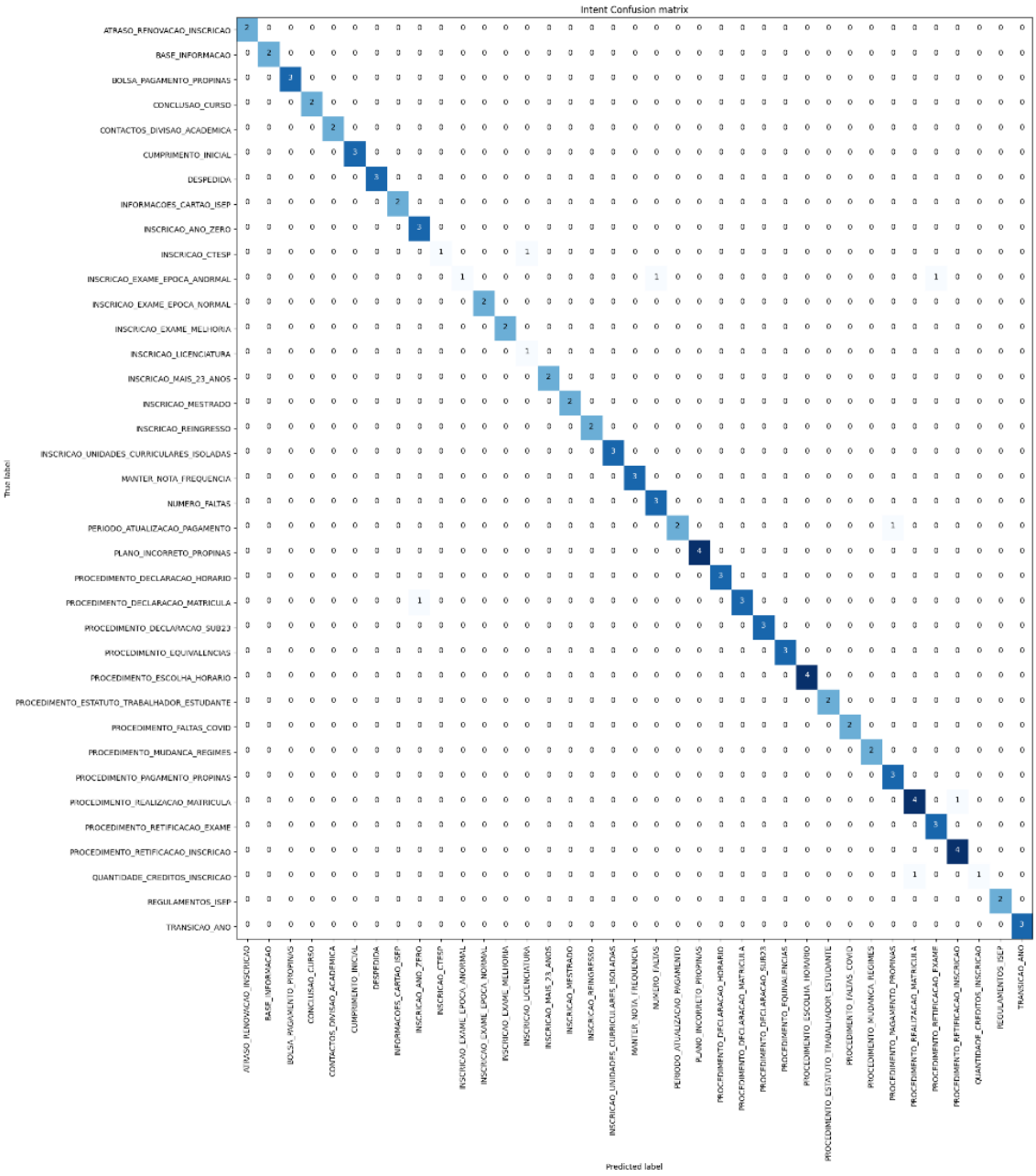

Figura 85. Matriz de confusão para a `config_roberta.yml`

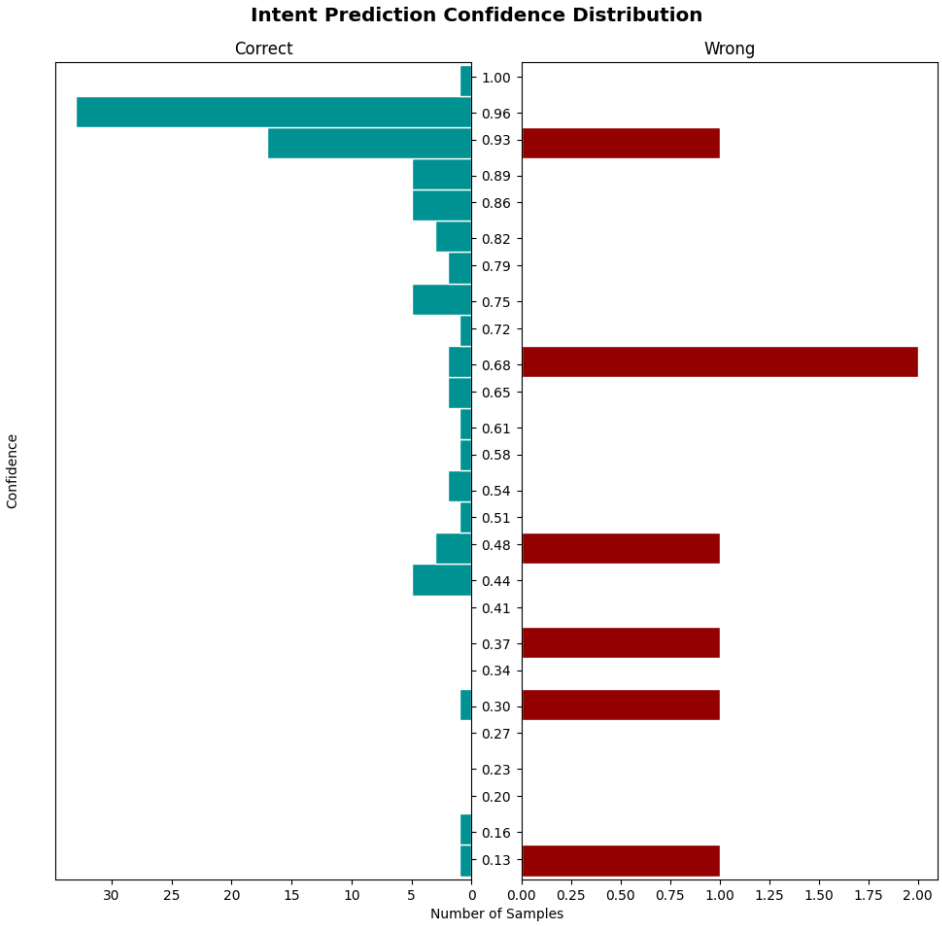

Figura 86. Diagrama de distribuição de confiança para a `config_roberta.yml`

4. Configuração `config_spacy_portuguese.yml`

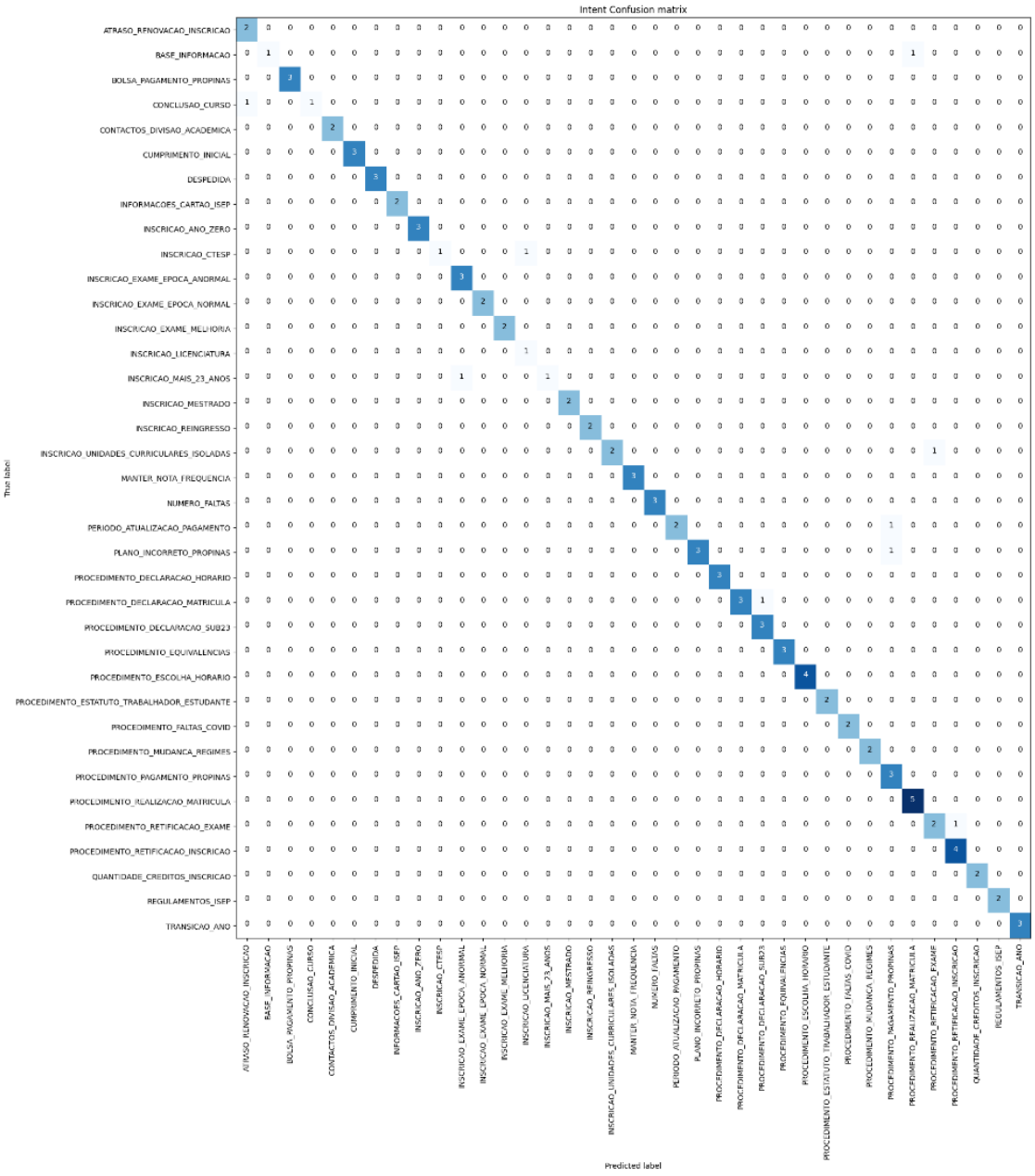

Figura 87. Matriz de confusão para a `config_spacy_portuguese.yml`

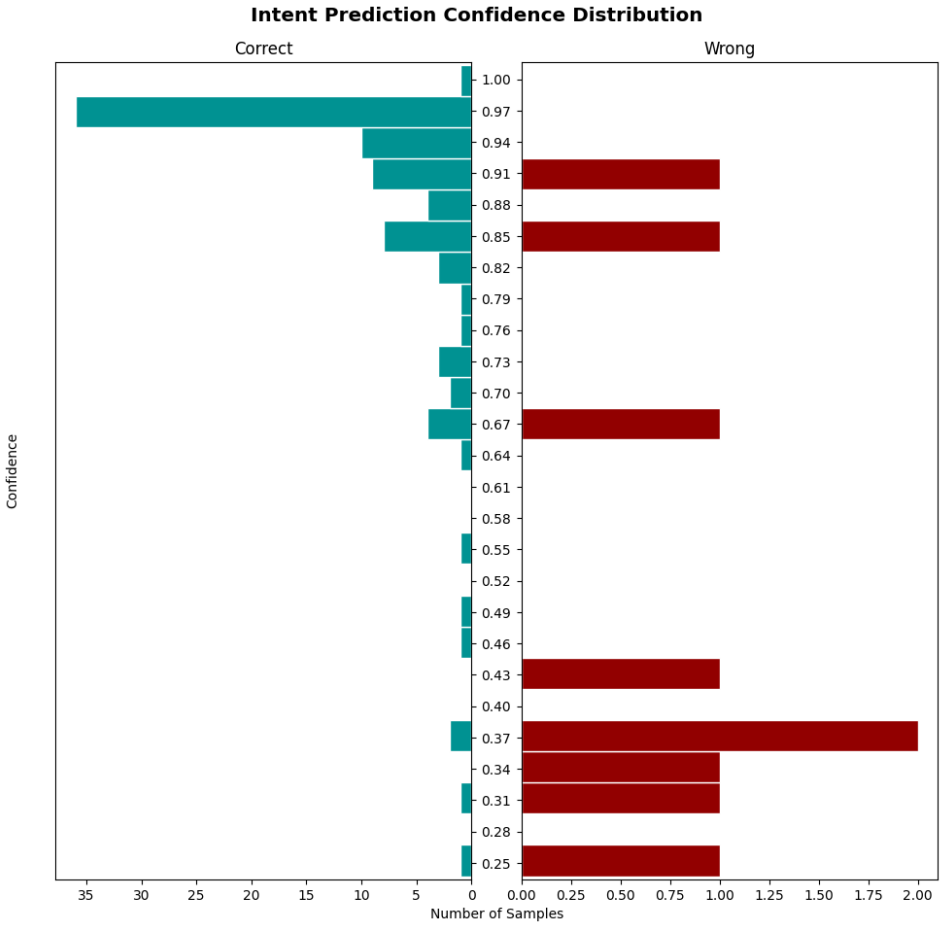

Figura 88. Diagrama de distribuição de confiança para a `config_spacy_portuguese.yml`

5. Configuração `config_domain.yml`

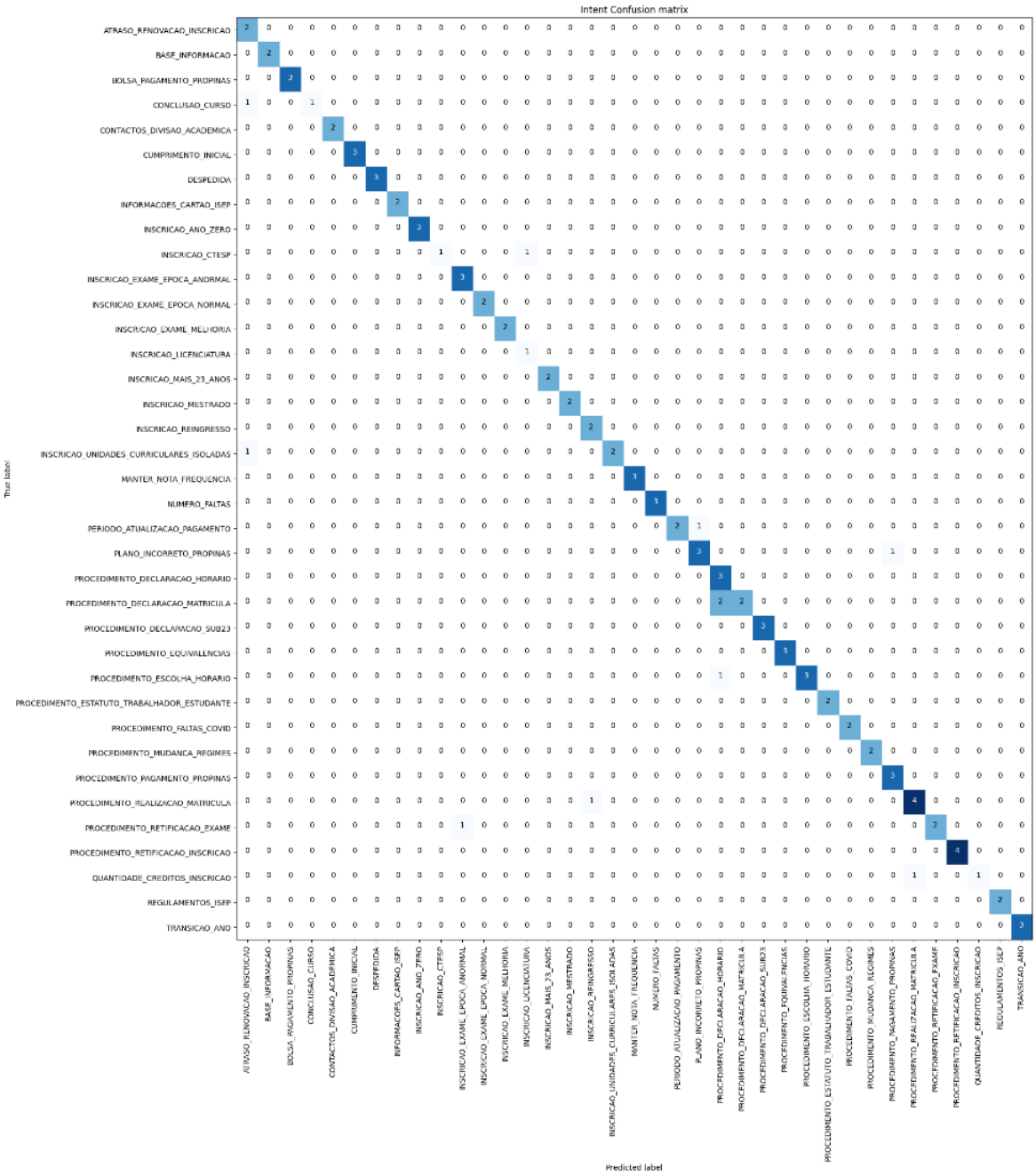

Figura 89. Matriz de confusão para a `config_domain.yml`

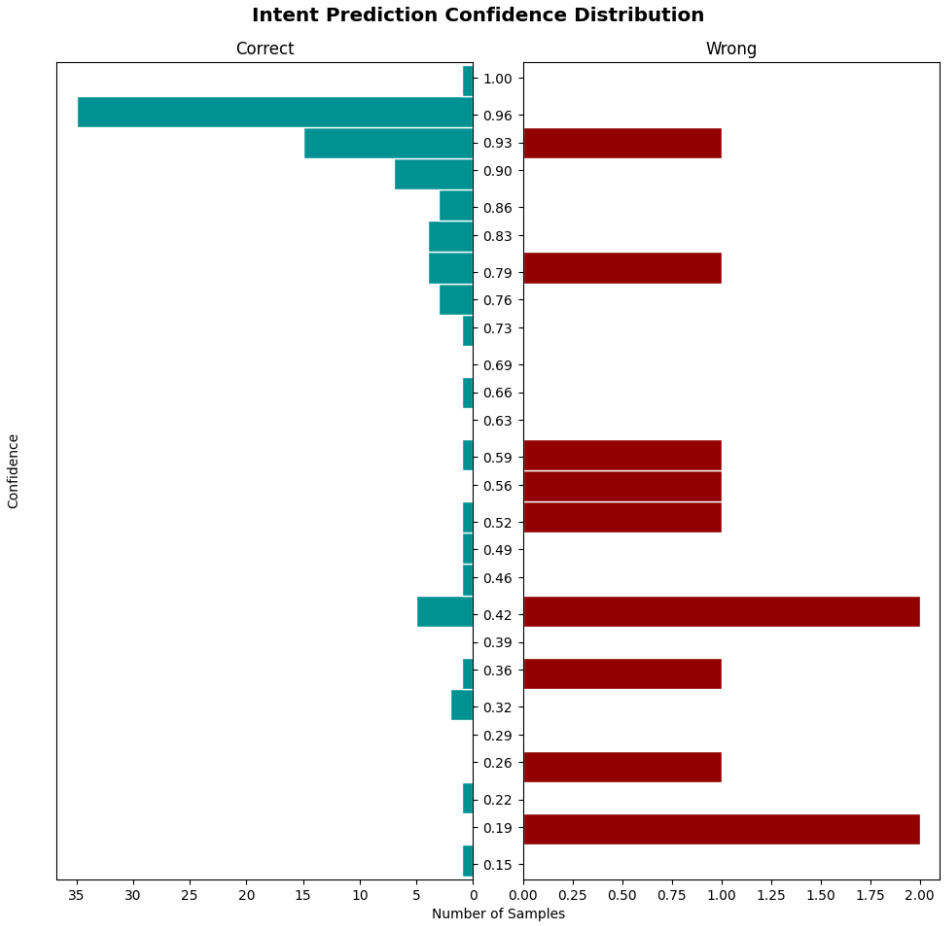

Figura 90. Diagrama de distribuição de confiança para a `config_domain.yml`

Anexo E Dados relativos ao estudo dos modelos de classificação (2ª fase)

1. Configuração `config_bert_diet0.yml`

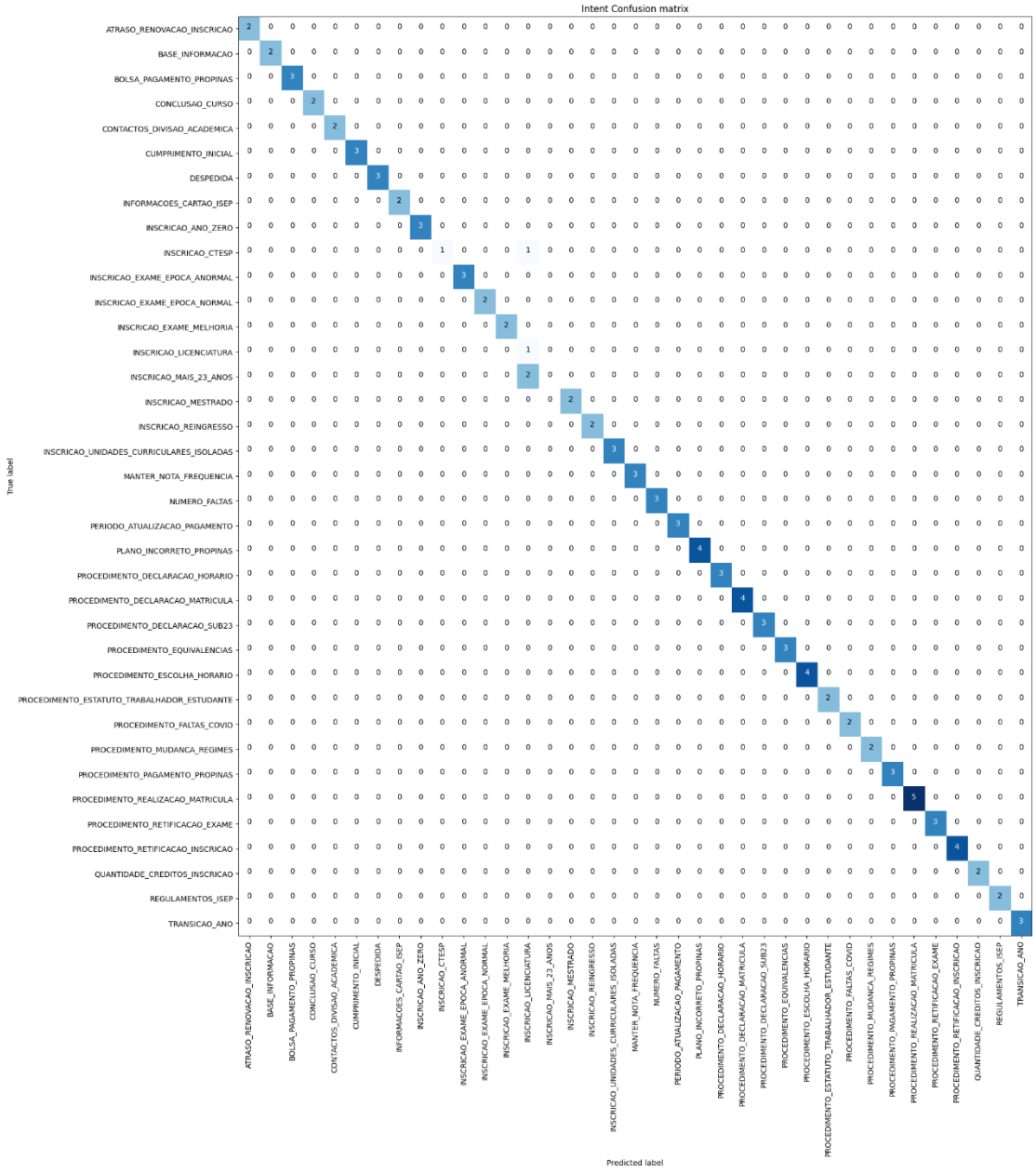

Figura 91. Matriz de confusão para a `config_bert_diet0.yml`

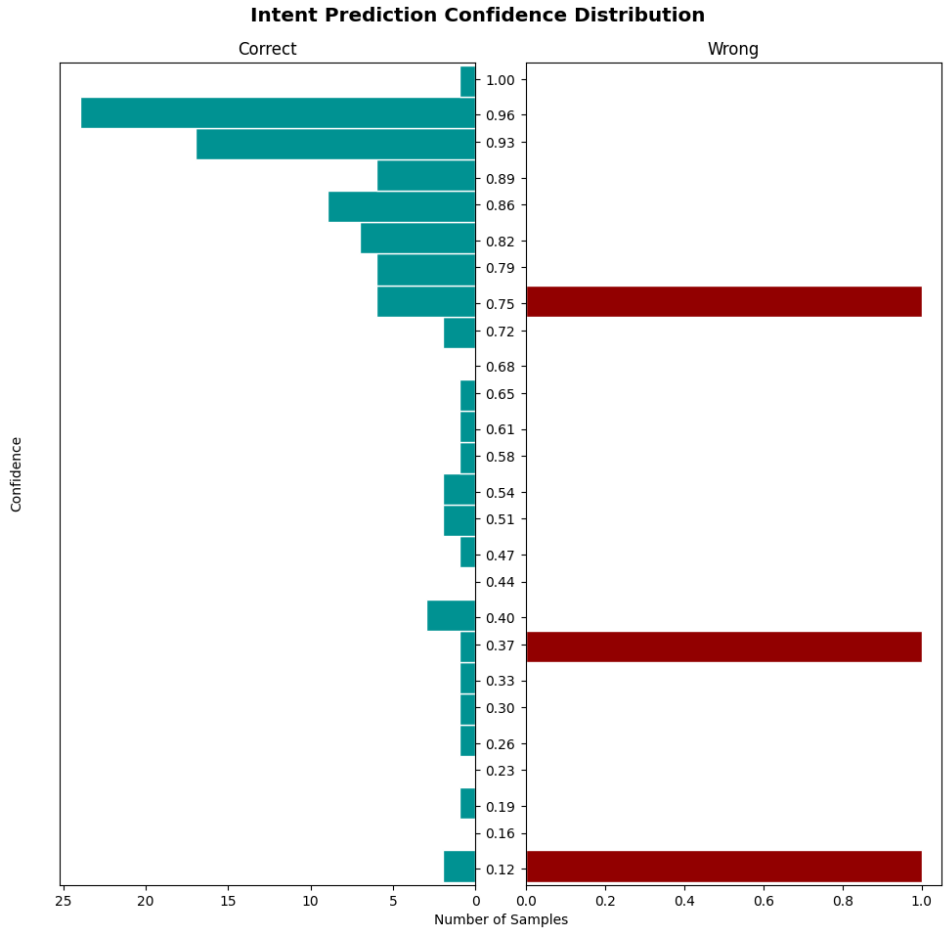

Figura 92. Diagrama de distribuição de confiança para a `config_bert_diet0.yml`

2. Configuração `config_roberta_diet0.yml`

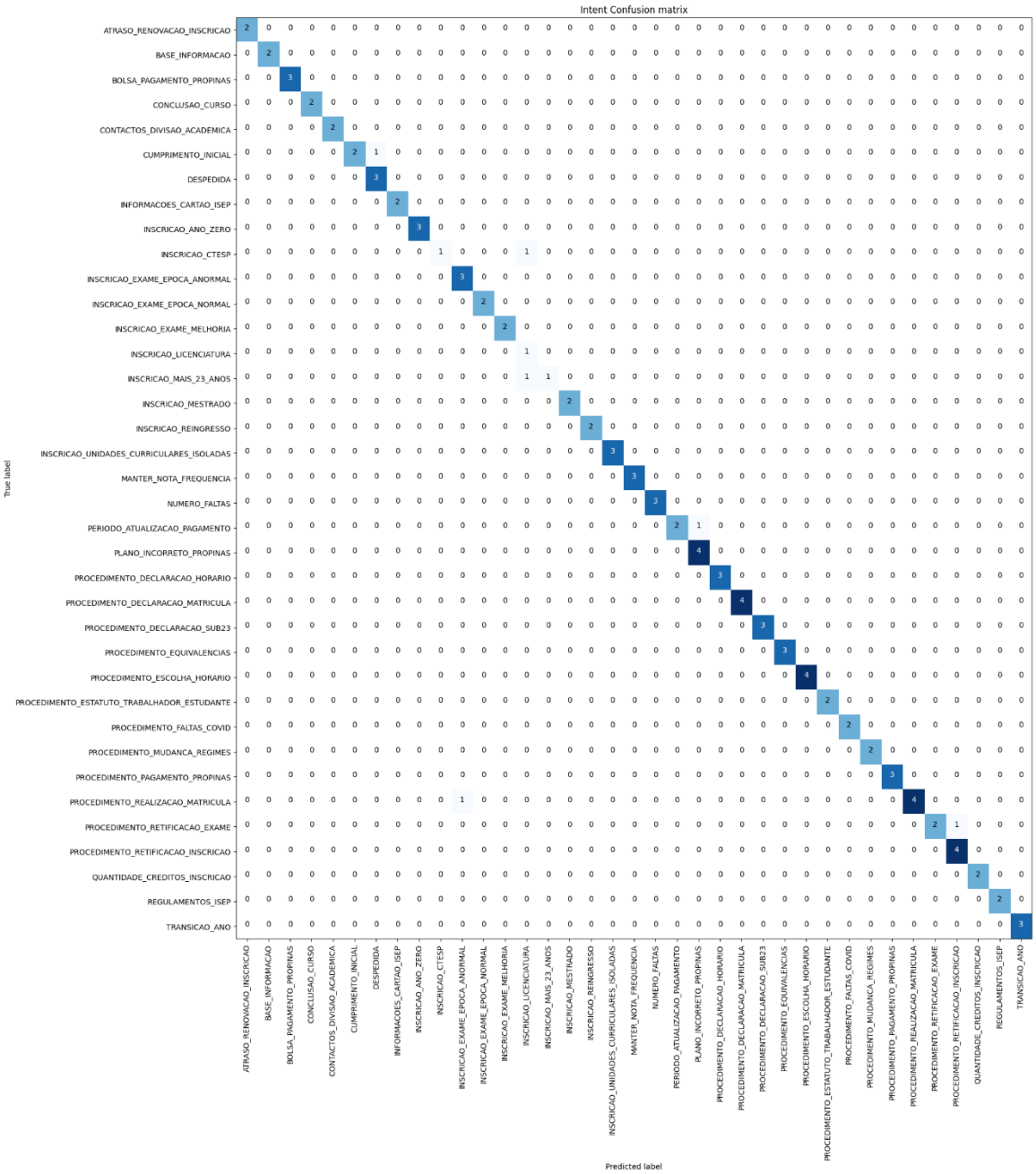

Figura 93. Matriz de confusão para a `config_roberta_diet0.yml`

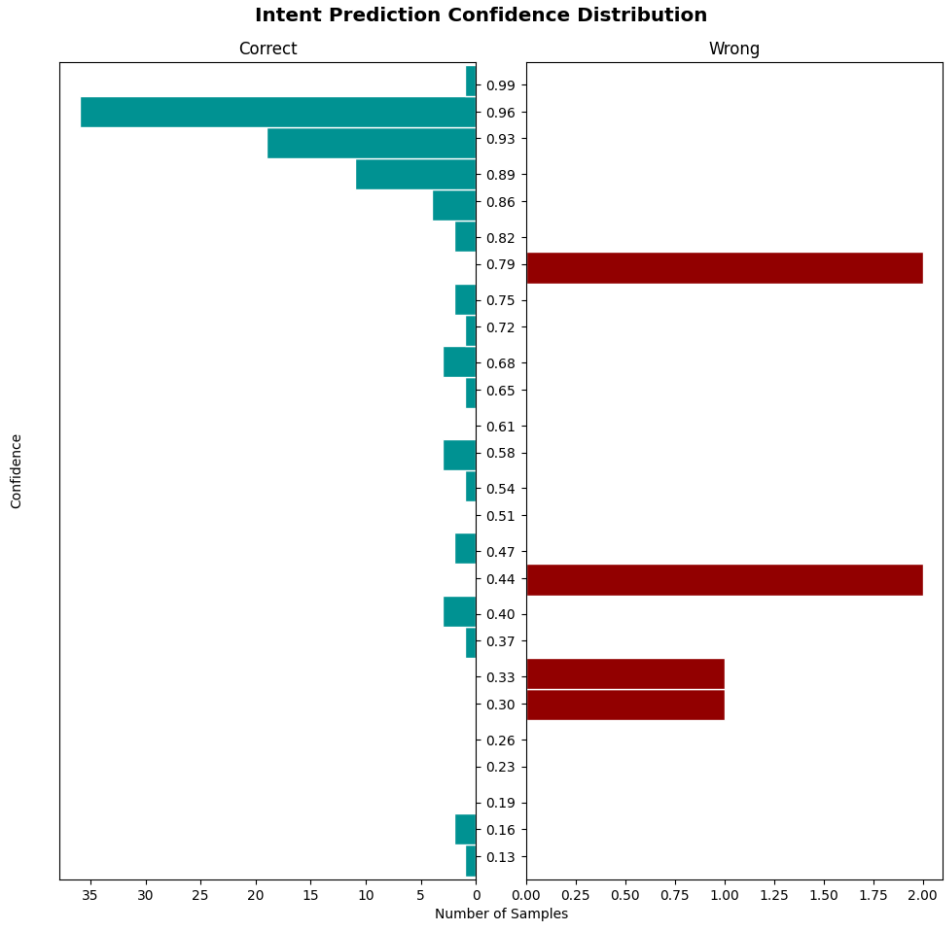

Figura 94. Diagrama de distribuição de confiança para a `config_roberta_diet0.yml`

3. Configuração `config_bert_diet1.yml`

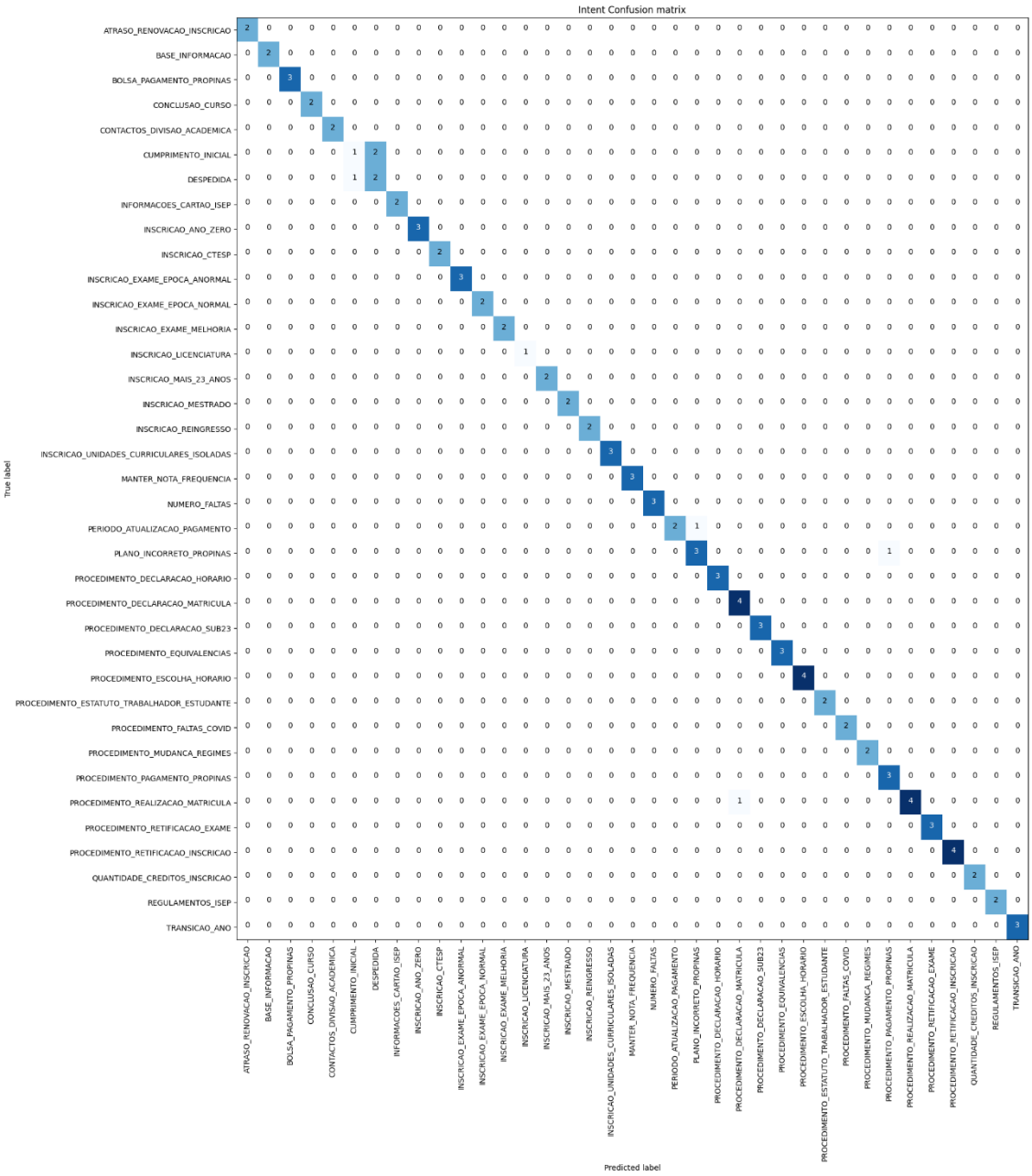

Figura 95. Matriz de confusão para a `config_bert_diet1.yml`

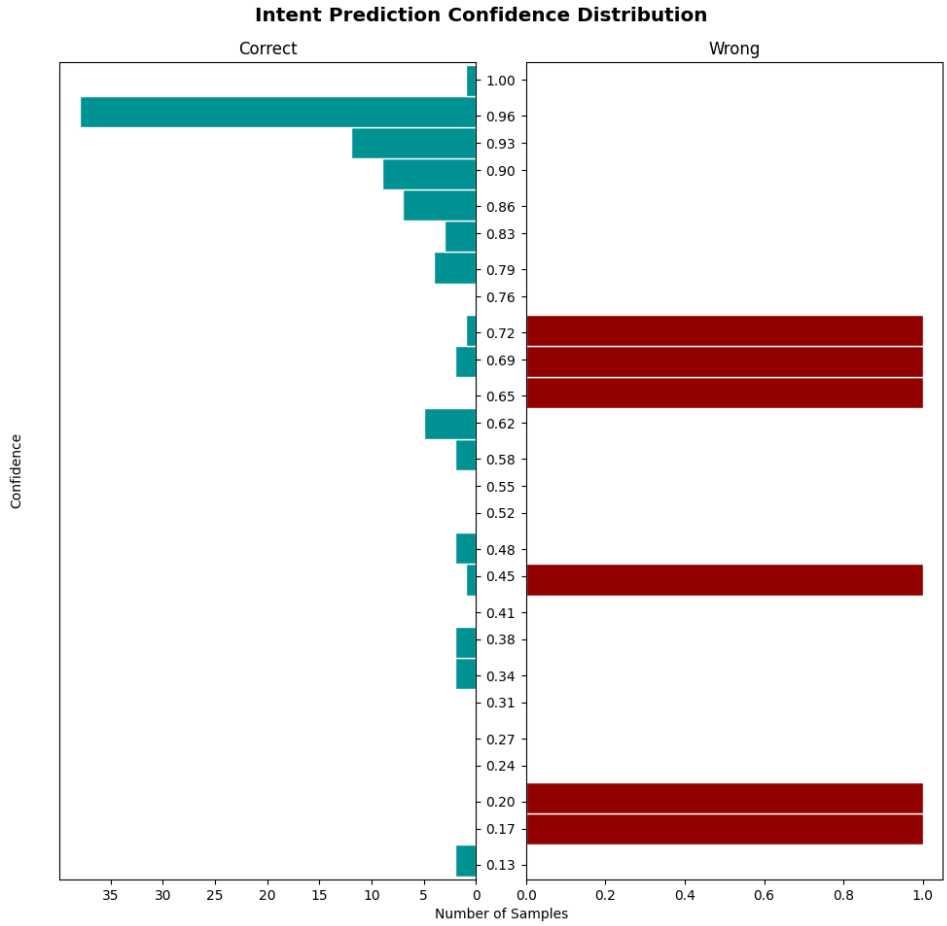

Figura 96. Diagrama de distribuição de confiança para a `config_bert_diet1.yml`

4. Configuração `config_roberta_diet1.yml`

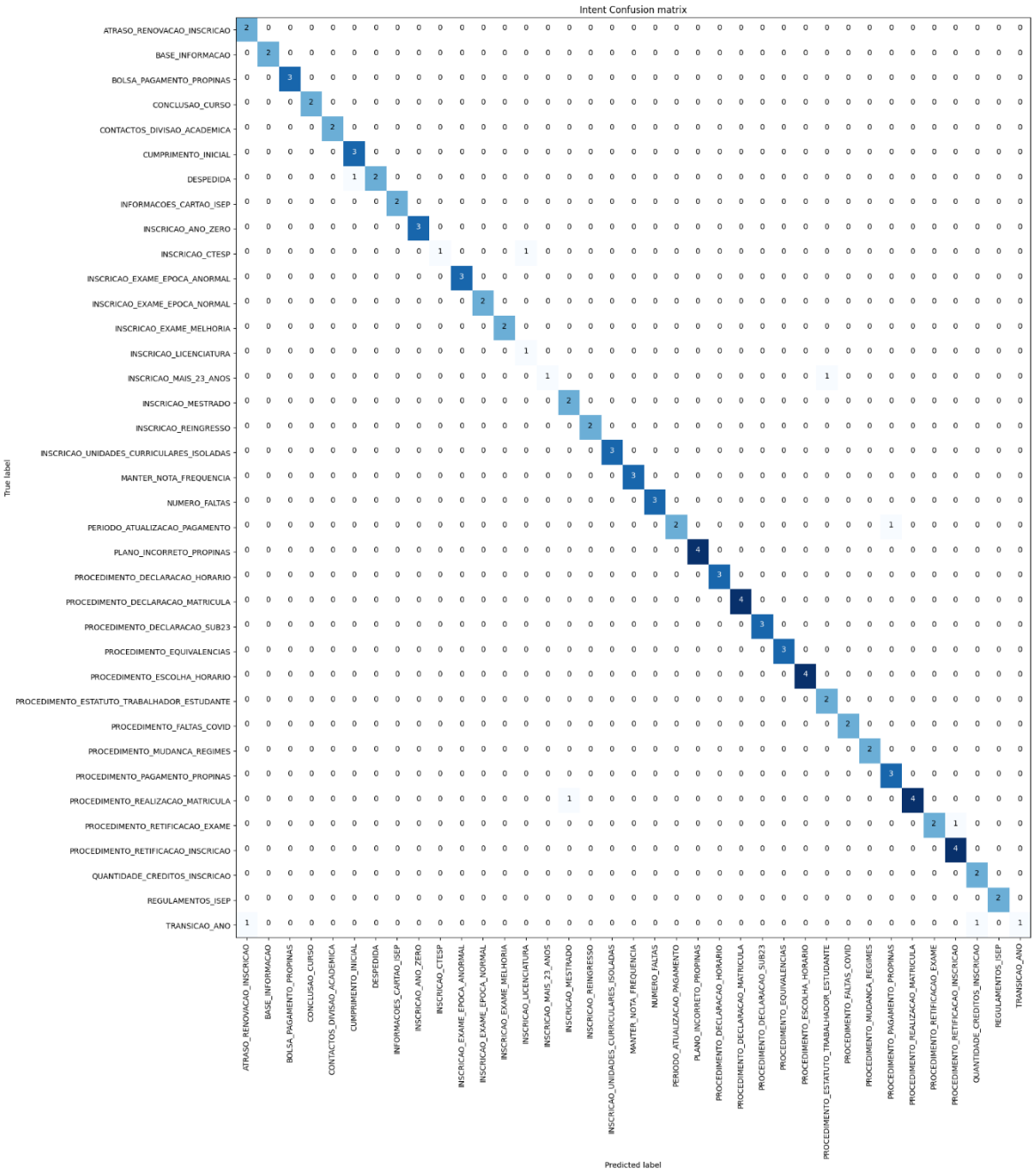

Figura 97. Matriz de confusão para a `config_roberta_diet1.yml`

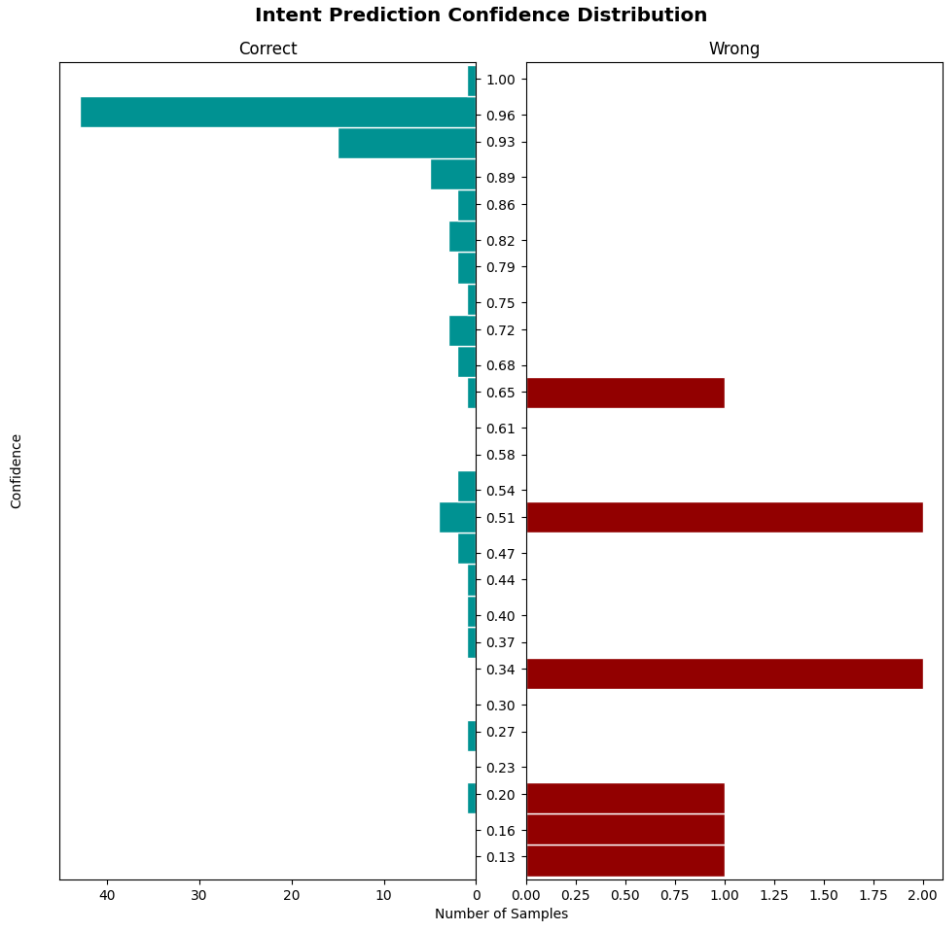

Figura 98. Diagrama de distribuição de confiança para a `config_roberta_diet1.yml`

5. Configuração `config_bert_diet2.yml`

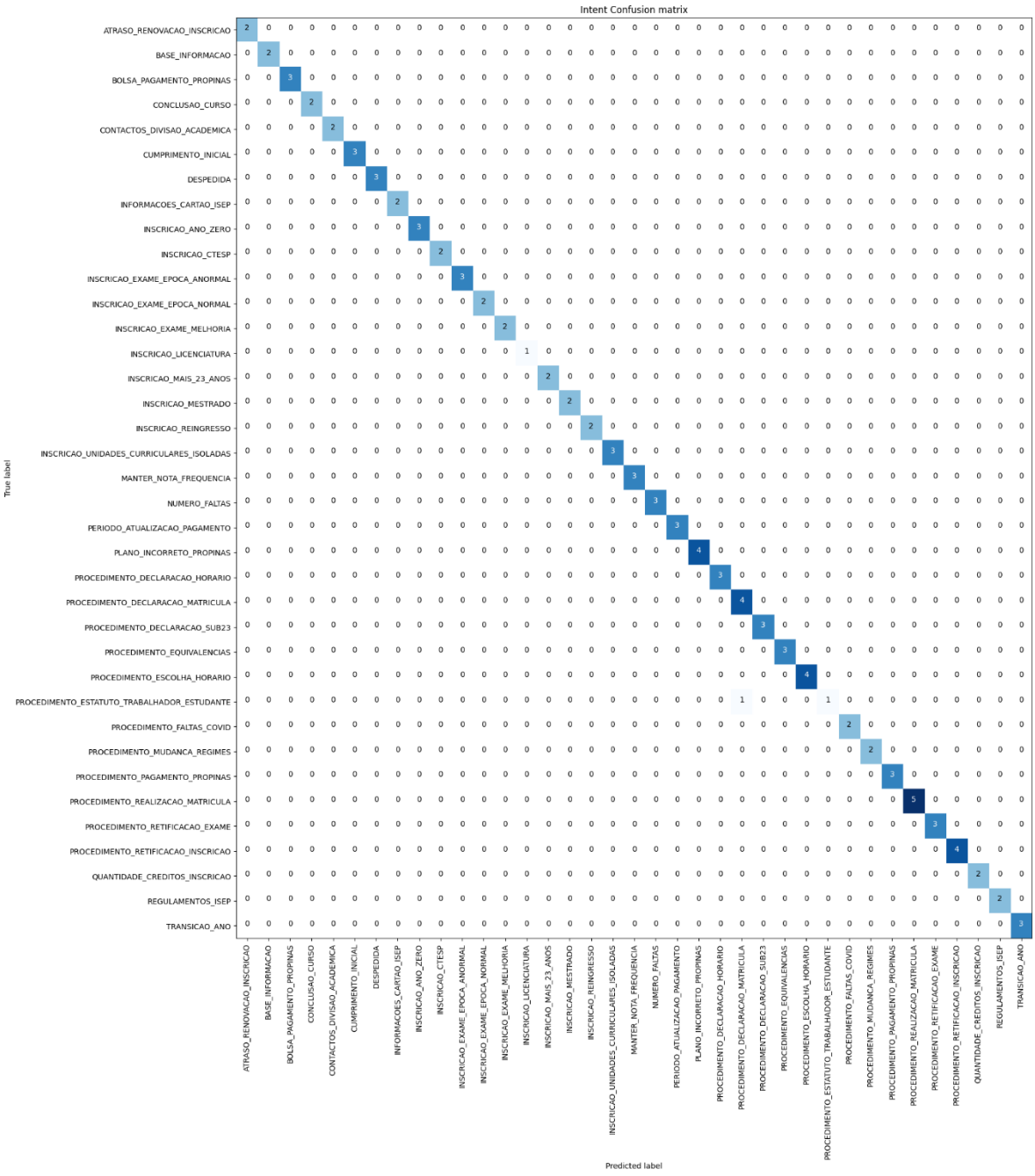

Figura 99. Matriz de confusão para a `config_bert_diet2.yml`

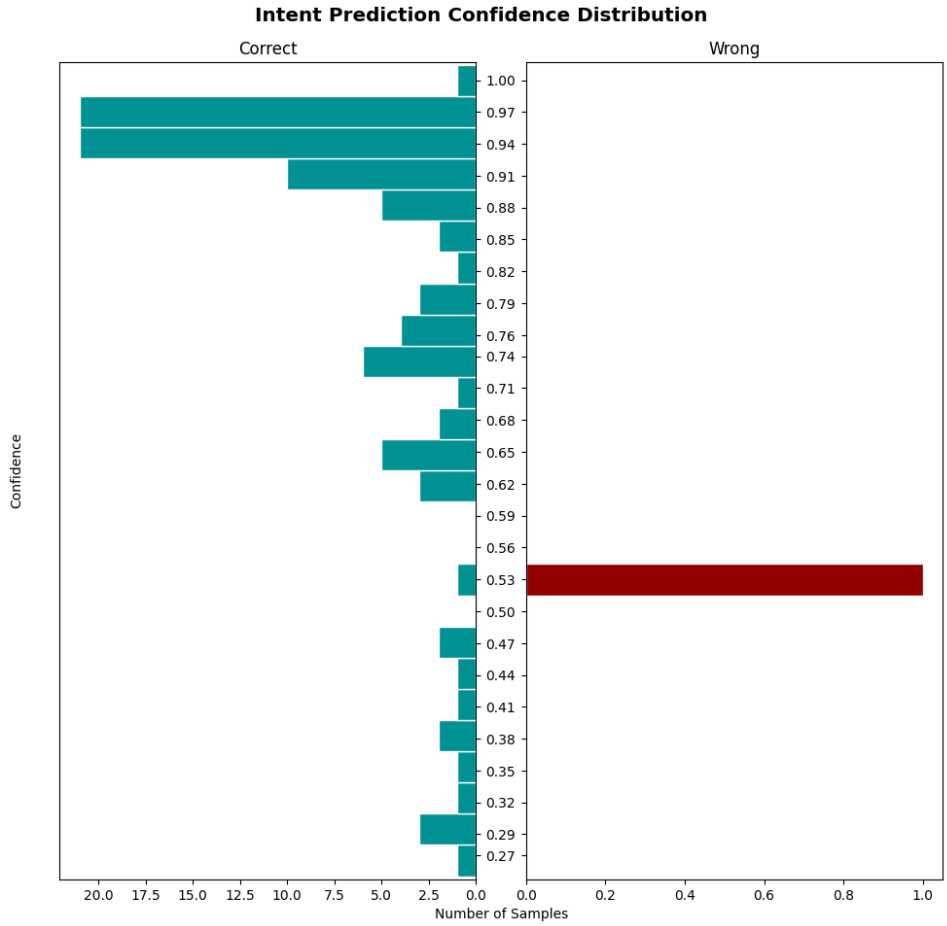

Figura 100. Diagrama de distribuição de confiança para a `config_bert_diet2.yml`

6. Configuração `config_roberta_diet2.yml`

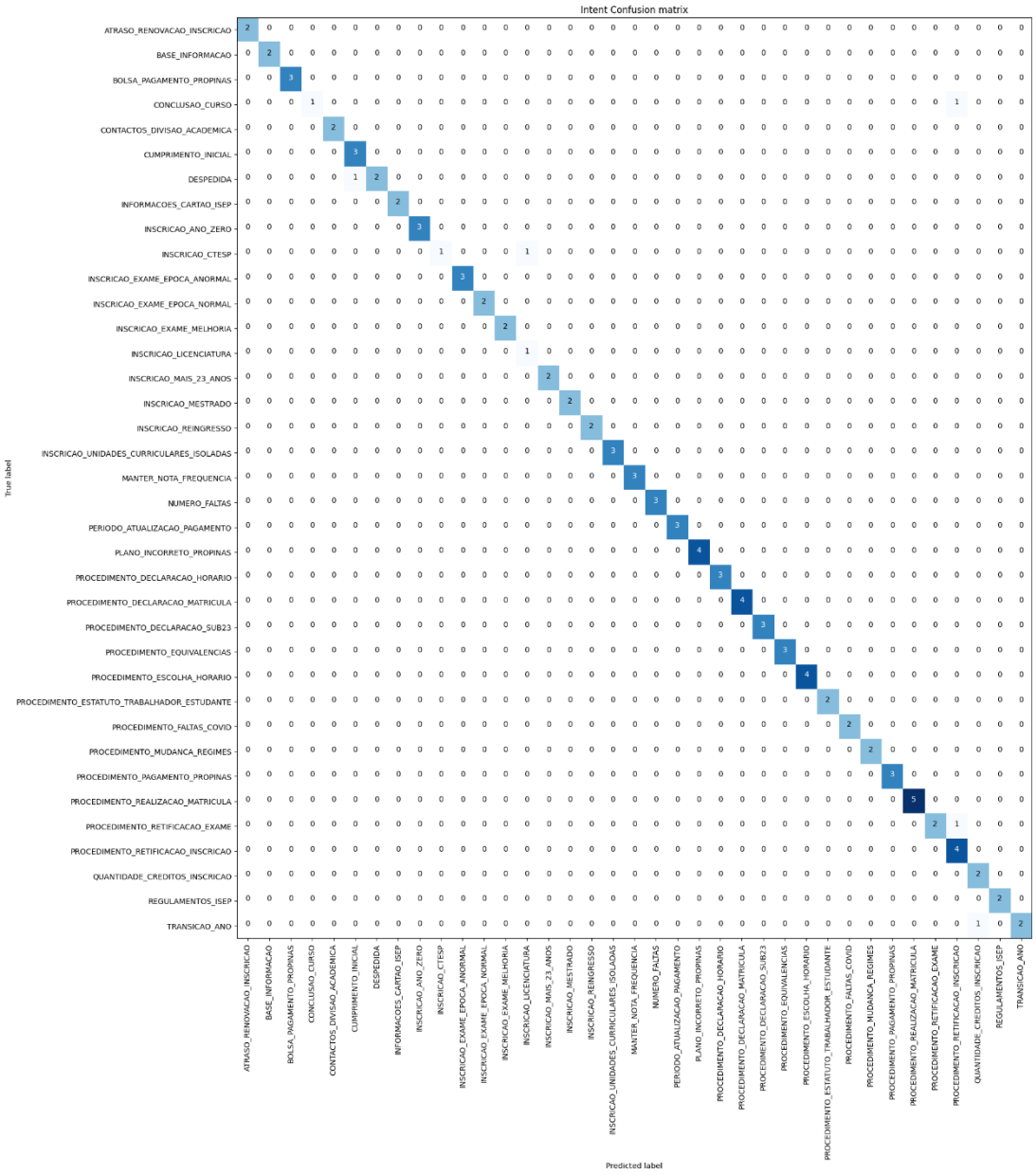

Figura 101. Matriz de confusão para a `config_roberta_diet2.yml`

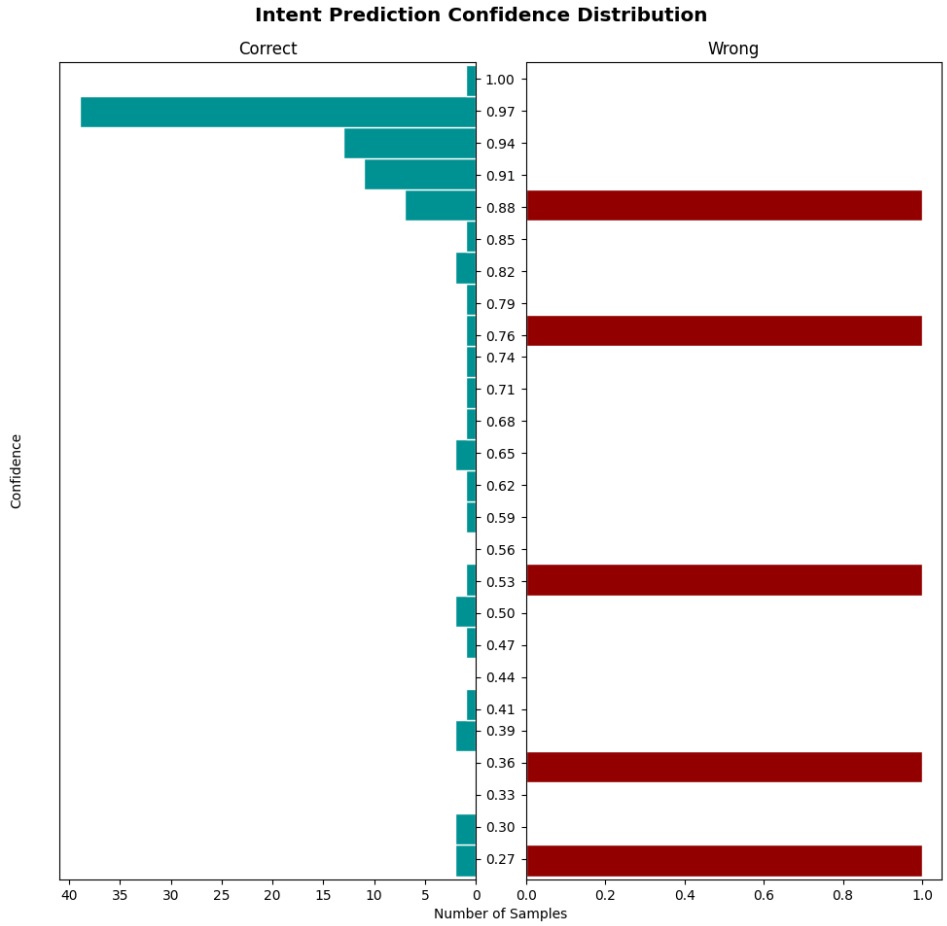

Figura 102. Diagrama de distribuição de confiança para a `config_roberta_diet2.yml`