Universidade do Minho
Escola de Engenharia
Departamento de Informática

Jaime Ricardo Faria Leite

Integração de Mecanismos de Raciocínio Adaptativo em Sistemas de Avaliação

Dezembro 2021

Universidade do Minho
Escola de Engenharia
Departamento de Informática

Jaime Ricardo Faria Leite

Integração de Mecanismos de Raciocínio Adaptativo em Sistemas de Avaliação

Master dissertation
Integrated Master's in Informatics Engineering

Dissertation supervised by
Professor Doutor Orlando Manuel de Oliveira Belo

Dezembro 2021

AGRADECIMENTOS

Gostaria de agradecer aos meus pais, a quem dedico esta dissertação, por me terem dado a possibilidade de vivenciar toda esta experiência académica e por me terem apoiado em todos os momentos. Estes anos na Universidade do Minho representaram uma longa aprendizagem, a nível pessoal e académico/profissional.

Um agradecimento também para o meu orientador, por me ter dado a possibilidade de trabalhar num projeto de sua autoria e pela paciência que demonstrou em todos os momentos (reuniões, escrita da dissertação, desenvolvimento da parte prática da mesma, entre outras coisas mais).

Aos meus colegas e amigos de curso deixo também um agradecimento por me terem ajudado a tornar-me uma pessoa diferente (e melhor, penso eu) no que se refere ao trabalho no seio de uma comunidade informática.

Por último, mas não menos importante, gostaria de agradecer a todos os professores que fizeram parte do meu processo académico e pelo tempo que dispenderam a lecionar o conteúdo académico e a solucionar todas as dúvidas.

DECLARAÇÃO DE INTEGRIDADE

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

RESUMO

Integração de Mecanismos de Raciocínio Adaptativo em Sistemas de Avaliação

Um sistema de raciocínio pode ser caracterizado como um agregado de componentes de *software* que realizam em conjunto processos de tomada de decisão complexos. Este tipo de sistema está bastante ligado a uma das áreas de trabalho mais mediáticas atualmente, a Inteligência Artificial. Algumas iniciativas de desenvolvimento dentro desta área tendem a incorporar este tipo de ferramenta em sistemas de avaliação, mais concretamente em tutores inteligentes, com o intuito de ajudar os estudantes no seu processo de aprendizagem.Nesta dissertação apresenta-se a conceção e a implementação de um conjunto de mecanismos de raciocínio baseado em casos e baseado em regras. Estes dois tipos de mecanismos foram idealizados para integrar o atual módulo de avaliação do sistema *Leonardo*, uma plataforma que complementa o estudo presencial dos alunos da Universidade do Minho. Os novos mecanismos, em particular os de raciocínio baseados em casos, complementam o processo de avaliação do sistema *Leonardo* aumentando as suas capacidades de raciocínio aquando da realização dos processos de avaliação estendendo as sessões de *Quizz*. Quanto aos mecanismos baseados em regras, estes representam uma importante camada entre o módulo de avaliação e a *interface* do tutor do sistema, visto que não permite apresentar questões de escolha múltipla na *interface* que não estejam de acordo com critérios estabelecidos por peritos. Nesta dissertação veremos como tais mecanismos foram fundamentados, desenvolvidos e integrados no sistema *Leonardo*.

PALAVRAS-CHAVE Sistemas de *eLearning*, Sistemas de Avaliação, Raciocínio Baseado em Casos, Raciocínio Baseado em Regras, Motor de Raciocínio

ABSTRACT

Integration of Adaptive Reasoning Mechanisms in Assessment Systems

A reasoning system can be characterized as an aggregate of software components that jointly perform complex decision-making processes. This type of elements is closely linked to one of the most popular areas today, Artificial Intelligence. Some development initiatives within this area tend to incorporate this type of tool into evaluation systems, specifically in intelligent tutors, in order to help students in their learning process. This dissertation presents the design and implementation of a set of case-based and rule-based reasoning mechanisms. This two types of mechanisms were design in order to integrate the current evaluation module of the *Leonardo* system, a platform that complements the in person study of students of University of Minho. New mechanisms, in particular case-based reasoning, complements the evaluation process of the *Leonardo* system increasing their reasoning skills when carring out evaluation processes by extend the quizz sessions. As for rule-based mechanisms, these represent an important layer between the evaluation module and the system tutor interface, as it does not allow to present multiple choice questions in the interface that are not in accordance with criteria established by experts. In this dissertation we will see how such mechanisms were substantiated, developed and integrated in the *Leonardo* system.

KEYWORDS eLearning Systems, Evaluation Systems, Case-based Reasoning, Rule-based Reasoning, Reasoning Engine

ÍNDICE

LISTA DE SIGLAS

CBR *Case-based Reasoning*. vi, 7, 18, 21, 22, 23, 24, 26, 27, 28, 29, 30, 32, 35, 37, 38, 39, 40, 43, 44, 47, 49, 55, 56, 57, 58

JSON *JavaScript Object Notation*. vi, 37, 43, 46, 50

CAI *Computer-assisted Instruction*. 3

ICAI *Intelligent Computer-assisted Instruction*. 3
ITS *Intelligent Tutoring Systems*. 3

RDF *Resource Description Framework*. 9

CQELS *Continuous Query Evaluation over Linked Stream*. 9

α**NLI** *Abductive Natural Language Inference*. 14
α**NLG** *Abductive Natural Language Generation*. 14

ART *Abductive Reasoning in Narrative Text*. 14

OWL *Ontology Web language*. 17

HTML *Hypertext Markup Language*. 18

BDI *Belief-Derire-Intentions*. 19

XOR *Exclusive Or*. 28

LISTA DE FIGURAS

LISTA DE TABELAS

1

INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

Ao longo dos últimos anos, várias áreas do conhecimento tem evoluído consideravelmente. A complexidade dos problemas nelas estudados é cada vez maior. Desde o estudo da espécie humana até áreas mais específicas podem encontrar-se problemas de variadas dimensões. A área do ensino é uma das áreas que tem suscitado mais interesse, principalmente por ser nela que se tenta compreender como automatizar os processos de aprendizagem.

Corbett et al. (1997) indicaram que os "computadores têm sido utilizados para uma variedade de objetivos educacionais desde o início dos anos 60" e que "A instrução baseada por computador foi introduzida com sucesso em todos os mercados da educação e do treino: escolas, casas, universidades, negócios e governo". Estas primeiras definições de instruções auxiliadas por computador foram designadas por CAI (*Computer-assisted Instruction*).

Nos inícios dos anos 70, alguns pesquisadores associaram o tutor humano como o modelo educacional da instrução baseada por computador. Para além disto, "procuraram utilizar técnicas de inteligência artificial para aplicar esse modelo como instruções 'inteligentes' baseadas por computador" (Corbett et al., 1997). Com o passar do tempo, o termo CAI seria substituído por um outro designado por ICAI (*Intelligent Computer-assisted Instruction*). O desenvolvimento deste tipo de sistemas continuou ocorrer até serem obtidos outro tipo de sistemas, os ITS (*Intelligent Tutoring Systems*). Usualmente, estes sistemas designam-se por tutores inteligentes.

Nwana (1990), por exemplo, definiu os tutores inteligentes como "programas computorizados que incorporam técnicas da comunidade de inteligência artificial para fornecer tutores que sabem o que ensinar, a quem ensinar e como ensinar". Este autor acrescenta que o planeamento e desenvolvimento destes sistemas resultam da junção de três áreas de estudo: as ciências da computação, a psicologia cognitiva e a pesquisa educacional. Uma definição mais atual destes sistemas foi apresentada por Nkambou et al. (2010), na qual se definem os tutores inteligentes como sendo programas complexos que "gerem tipos heterogéneos de conhecimento".

Todas estas definições são exemplos de caracterização dos tutores inteligentes. De uma forma geral, é possível caracterizar estes sistemas como componentes digitais que guardam e manipulam informação, com o intuito de fornecerem conteúdo adaptável aos seus utilizadores. Como se pode observar na Figura 1, um sistema de tutoria inteligente é tipicamente composto pelo modelo pedagógico, modelo da interface do utilizador, modelo do domínio e modelo do estudante. A ligação entre os modelos é um fator fundamental e necessário para que

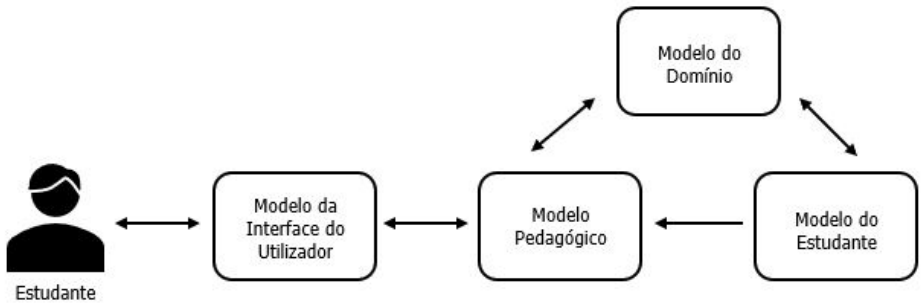

Figura 1: Arquitetura funcional de um sistema inteligente de tutoria - figura adaptada de Almasri et al. (2019)

os utilizadores/estudantes possam testar o seu conhecimento com material de estudo adaptável, resultante de estratégias de ensino desenvolvidas por peritos em determinadas áreas do conhecimento. De seguida, abordaremos, com mais detalhe, cada um destes modelos.

O Modelo Pedagógico, como indicado por Almasri et al. (2019), "funciona em função da estratégia de ensino adotada pelo sistema, tendo em conta o tempo de resposta do aluno e o seu perfil". Ou seja, seleciona o material de estudo dependendo das características dos utilizadores. Este modelo, por vezes designado por *teaching model* ou *expert model*, interliga-se com o Modelo do Estudante, que cria e mantém o perfil dos estudantes. O perfil dos estudantes é tipicamente composto pelas suas informações pessoais e pelo resultado das suas interações com o tutor inteligente (respostas corretas, incorretas, ajudas fornecidas pelo sistema, entre outras coisas mais).

O Modelo do Domínio, que pode também ser designado por modelo de conhecimento, "representa o domínio de conhecimento e como o expert atua no domínio de conhecimento" (Almasri et al., 2019). Ou seja, possui as informações (conhecimento) acerca do material de estudo de determinadas áreas do ensino, como por exemplo ciências da computação ou matemática, e um conjunto de estratégias que modelam o comportamento do tutor inteligente nessas áreas.

O Modelo da Interface do Utilizador permite definir os meios necessários para se realizar a comunicação entre o utilizador e o tutor. A informação transmitida aos utilizadores resulta da interação dos restantes componentes apresentados na Figura 1. Ou seja, este modelo permite transmitir ao utilizador, através de estratégias de resolução específicas, o tipo de material de estudo que o sistema considera mais adequado às suas necessidades.

1.2 MOTIVAÇÃO

O raciocínio adaptativo presente nos tutores inteligentes permite adaptar o material de estudo ao comportamento dos utilizadores. Os tutores inteligentes tornar-se-ão mais sólidos, se as técnicas de raciocínio implementadas forem bastante eficazes, variadas e se complementarem entre si. Estas técnicas permitem aplicar métricas de avaliação, analisar dados, facilitar processos de tomada de decisão, entre outras coisas mais. De

uma forma mais concreta, automatizam mecanismos de interpretação e de desenvolvimento cognitivo, que não seriam tão eficientes se fossem aplicados por seres humanos.

Na Universidade do Minho foi projetado um sistema de avaliação, designado por *Leonardo* (Belo et al., 2019), que complementa o estudo presencial dos alunos em várias áreas do conhecimento. Os alunos têm a possibilidade de participar em sessões de *Quizz* e receber *feedback* por parte do sistema relativamente às suas prestações. Desde há algum tempo que têm sido introduzidos novos componentes para melhorar o desempenho do *Leonardo*, desde sistemas de opinião até mecanismos que permitem os utilizadores comunicar com o sistema através da voz. Para além disso, nas fases iniciais de desenvolvimento, foram implementados mecanismos de *profiling*, para representar as informações das *performances* dos alunos, e mecanismos de avaliação, que lançam o material de estudo tendo em conta estas *performances*.

O *Leonardo* ainda está em fase de desenvolvimento e como tal é possível a implementação de novas técnicas de raciocínio. A inclusão de novos mecanismos de raciocínio adaptativo é bem-vinda, para permitir "perceber" melhor a evolução do conhecimento dos estudantes ao longo do tempo, bem como adaptar em tempo real os processos de avaliação dos estudantes. Aspetos como estes, conduziram à principal motivação dos trabalhos desta dissertação. Dada a importância que este tipo de sistemas assume na área do ensino, a introdução de novas técnicas num tutor inteligente complementa a sua estrutura, a sua "forma de pensar" e introduz mais variedade de soluções no seu comportamento. Desta forma contribui-se positivamente para a comunidade académica e educativa.

1.3 OBJETIVOS

O principal objetivo dos trabalhos desta dissertação foi tornar mais completa a arquitetura do um sistema de tutoria inteligente, *Leonardo*, já apresentado anteriormente na Secção 1.2. Tendo sido analisadas as necessidades deste sistema, pretendeu-se desenvolver e implementar novos mecanismos de raciocínio, mais concretamente desenvolver e integrar no sistema um comjunto de mecanismos de raciocínio baseado em casos e um mecanismo de raciocínio baseado em regras. Estes novos mecanismos deveriam ser interligados posteriormente com os módulos já implementados no tutor.

O mecanismo baseado em casos deveria funcionar como um sistema de "suporte" ao módulo de avaliação do tutor e estender as sessões de estudo, quando o módulo de avaliação indicar o fim da avaliação dos alunos. Dessa forma será possível dar "mais uma oportunidade" aos alunos para testar o seu conhecimento. Este prolongamento das sessões de estudo deveria ser acompanhado da geração e apresentação de novo material de estudo, mais concretamente de novas questões de escolha múltipla. Na geração das questões dever-se-ia considerar o percurso dos vários utilizadores do sistema e assim fornecer a questão mais interessante para o momento atual da sessão de estudo. De realçar, também, que se pretendia criar uma nova alternativa à inserção das questões de escolha múltipla no sistema. Conjuntamente com esta nova forma, uma questão pode agora ser inserida no sistema diretamente na base de dados ou através da utilização do módulo de edição, dedicado aos peritos numa área do conhecimento, como por exemplo professores.

Por último, mas não menos importante, o mecanismo de raciocínio baseado em regras deveria condicionar o comportamento do tutor inteligente através da execução de regras, que terão em consideração certas propriedades calculadas e mantidas pelo módulo de *profiling*. Este condicionamento deveria atuar, contrariamente ao sistema baseado em casos, sempre que uma questão de escolha múltipla estivesse pronta para ser apresentada na *interface* do sistema. Seria função deste novo mecanismo verificar a validade da questão com o auxílio das regras.

A manutenção dos elementos da base de conhecimento é bastante importante para o bom funcionamento de qualquer sistema de avaliação. As regras que modelam o mecanismo baseado por regras não são exceção. Assim, pretendeu-se, também desenvolver um editor de regras, que permitisse aos professores credenciados no sistema acederem e manipularem estes elementos, de forma rápida e intuitiva. Este editor deveria ser incluído na plataforma do tutor inteligente. Para além de edição das regras, as funcionalidades de remoção e criação de regras também foram integradas neste novo componente. Na secção seguinte, poderemos ver como os objetivos e o trabalho enunciado foi concretizado.

1.4 TRABALHO REALIZADO

O trabalho desta dissertação pode dividir-se em duas grandes etapas: uma análise teórica aos à implementação de mecanismos de raciocínio adaptativo em sistemas de avaliação e a implementação de um conjunto adicional de mecanismos de raciocínio para o sistema *Leonardo*. De seguida aborda-se mais pormenorizadamente as várias fases que compõem estas etapas.

Numa primeira fase dos trabalhos desta dissertação foi efetuada uma análise cuidada aos trabalhos desenvolvidos na área de sistemas de raciocínio, sendo grande parte destes expressos sob a forma de artigos científicos. Para além disto, analisou-se a forma como tipicamente se interligam os sistemas de raciocínio e os sistemas de avaliação para compreender melhor como se poderia responder corretamente aos requisitos práticos desta dissertação. Posteriormente, foi realizado um levantamento de requisitos acerca das necessidades do sistema *Leonardo*. Isto permitiu realizar a modelação de novos mecanismos de raciocínio que colmatassem essas necessidades. Foram também estudadas as ferramentas utilizadas no desenvolvimento do tutor inteligente, mais concretamente a linguagem de programação *Python* (Python.org, 2021), a ferramenta de gestão e desenvolvimento de servidores *web Flask* (Pallets, 2021) e o sistema de gestão de bases de dados *MongoDB* (MongoDB, 2021).

Os trabalhos referidos permitiram o desenvolvimento de um sistema de raciocínio baseado em casos e de um mecanismo de raciocínio baseado em regras, que deram origem a dois novos módulos de raciocínio do *Leonardo*. Estes dois módulos foram interligados com os módulos que já estavam implementados no tutor, dos quais se destacam o módulo de *profiling* e o módulo de avaliação.

O sistema baseado em casos gera automaticamente questões de escolha múltipla nos casos nos quais as sessões de *Quizz* terminam. Esta geração tem em consideração o percurso dos utilizadores ao longo das várias sessões de estudo visto que, como se vai poder verificar mais adiante neste documento, o processo de raciocínio deste sistema inicia-se com uma filtragem das questões respondidas pelos alunos, para um determi-

nado domínio de conhecimento e para um nível de dificuldade. De realçar que, as novas questões têm a mesma estrutura daquelas que foram definidas anteriormente na base de dados do tutor inteligente.

O mecanismo baseado em regras funciona como camada de ligação entre o módulo de avaliação e a *interface* do tutor, submetendo as questões de escolha múltipla a um processo de análise antes destas serem apresentadas aos utilizadores nas sessões de *Quizz*. As questões são apresentadas na *interface* do sistema se corresponderem a determinadas regras definidas por peritos numa dada área de conhecimento.

Para além dos dois mecanismos de raciocínio referidos foi também desenvolvido um editor de regras para criar, visualizar, editar e remover as regras que modelam o mecanismo de raciocínio baseado em regras. Atualmente, este editor encontra-se implementado na plataforma do tutor.

1.5 ORGANIZAÇÃO DA DISSERTAÇÃO

Neste primeiro capítulo apresentámos a contextualização, motivação e objetivos que sustentaram o trabalho desenvolvido ao longo do processo de dissertação. Para além do presente capítulo, esta dissertação contém mais quatro capítulos:

- **Capítulo 2 - Sistemas de Raciocínio –** que apresenta os pontos mais relevantes dos sistemas de raciocínio, como por exemplo a arquitetura típica de sistemas deste tipo e os diferentes tipos de sistemas de raciocínio. Para além disto, são apresentados alguns exemplos de aplicação destes sistemas na área do ensino.

- **Capítulo 3 - Avaliação Baseada em Casos –** que indica as principais características da utilização de casos no processo de avaliação. Mais concretamente, aborda o raciocínio baseado em casos (CBR), desde a sua definição e processo de desenvolvimento até à sua utilidade e aplicação. Adicionalmente, são indicados também alguns exemplos de aplicação, bem como são apresentadas as vantagens e as desvantagens do CBR.

- **Capítulo 4 - Integração de CBR num Sistema de Avaliação –** que aborda, com detalhe, os mecanismos de raciocínio desenvolvidos nesta dissertação para integração num sistema de *eLearning*. De forma a revelar o processo de desenvolvimento de tais mecanismos, apresentam-se os pontos mais importantes da implementação de um sistema de raciocínio baseado em casos num sistema de avaliação, bem como se apresentado um método complementar para verificação das questões lançadas pelo módulo de avaliação do sistema de ensino escolhido.

- **Capítulo 5 - Conclusões e Trabalho Futuro –** que explicita as considerações mais relevantes acerca do trabalho desenvolvido, englobando um resumo do trabalho realizado, uma avaliação crítica daquilo que foi realizado, e uma apresentação dos pontos positivos e negativos do trabalho que considerámos dignos de realce. Por fim, apresentamos algumas linhas de orientação para trabalho futuro.

2

SISTEMAS DE RACIOCÍNIO

2.1 APRESENTAÇÃO E ARQUITETURA FUNCIONAL

Os sistemas de raciocínio são elementos computorizados que permitem aplicar um conjunto de técnicas de raciocínio num determinado contexto, com o objetivo de gerar conclusões com base num dado conjunto de dados. No âmbito destes sistemas, a realização de estudos é algo frequente. Contudo não é fácil encontrar uma definição concreta que caracterize os sistemas de raciocínio. Apesar disso, verifica-se que estes elementos podem ser utilizados como o "cérebro" de sistemas de maior dimensão, como, por exemplo, em sistemas de avaliação ou em sistemas de diagnóstico. De seguida, apresenta-se um exemplo no qual se pode observar as características que acabámos de mencionar.

Shih et al. (2010) apresentaram uma proposta de desenvolvimento de um sistema de telemedicina que auxilia a monitorização e classificação de batimentos cardíacos em idosos. A estrutura deste sistema pode ser observada na Figura 2. Este sistema é composto pelas unidades do paciente (*Patient Unit*) e do profissional de saúde (*Clinician Terminal*), do hospital (*Hospital Information System*) e de gestão (*Management Unit*). A primeira destas unidades é responsável por anotar, ao longo do tempo, os batimentos cardíacos do idoso e por enviar essa informação para a unidade de gestão. A segunda unidade permite que os profissionais de saúde tenham um contacto permanente com as informações dos batimentos cardíacos dos pacientes. A terceira unidade do hospital tem o servidor e a base de dados do hospital associado a este sistema. A quarta unidade é responsável, entre outras coisas, por verificar a anormalidade dos batimentos cardíacos, através de um sistema de raciocínio sustentado por regras de produção *fuzzy* (*Fuzz production rules*). Como indicado pelos autores, o modelo de *Fuzzy Petri Net* "pode ser utilizado para descrever a maioria das relações e dos comportamentos de um sistema dinâmico de eventos discretos". Este tipo de modelo foi utilizado para se compreender os diferentes tipos de batimentos cardíacos anormais. As regras de produção representam o funcionamento deste modelo no contexto do sistema de raciocínio, isto é, representam a forma como este reage aos dados de *input*.

A unidade do paciente monitoriza constantemente os batimentos cardíacos dos idosos e envia sinais *ECG* – que podem ser vistos como "um gráfico produzido por um eletrocardiógrafo, que recolhe a voltagem elétrica no coração na forma de um gráfico" (Shih et al., 2010) – para a unidade de gestão.

Na unidade de decisão (*Decision Module*) verifica-se se o sinal recebido representa ou não um batimento cardíaco anormal (*abnormal heartbeat*). Para tal, aplica as regras mencionadas anteriormente, que estão armazenadas na base de conhecimento do sistema. Caso sejam detetadas anomalias nos batimentos cardíacos

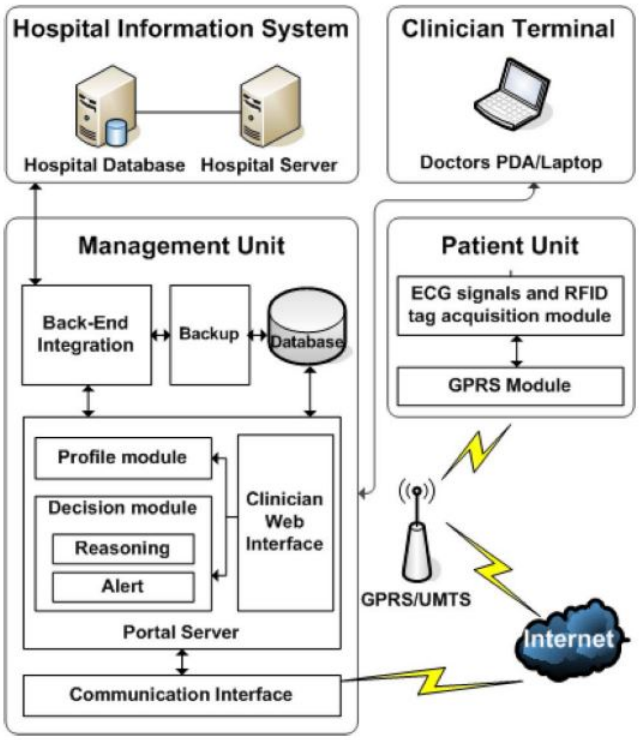

Figura 2: Arquitetura do sistema de telemedicina - imagem extraída de Shih et al. (2010)

do paciente é enviado um sinal de alerta para os profissionais de saúde, que, assim, podem tratar de forma mais adequada do idoso em questão. Na Figura 3 apresenta-se um exemplo de como os profissionais de saúde podem monitorizar o estado dos pacientes com este sistema. A janela do sistema que está apresentada na Figura 3 revela-nos a informação pessoal do paciente (*Patient Profile*), o número de vezes que ocorreram os diferentes tipos de sinais *ECG* (*ECG Signal Statistics*) e a indicação da condição atual do paciente (*Patient Condition*).

O estudo deste e de outros sistemas de raciocínio permitiu identificar quais os componentes que desempenham um papel chave no funcionamento deste tipo de sistemas. Usualmente, estes componentes costumam desempenhar tarefas específicas e relacionam-se entre si de uma forma bastante própria. A partir desse estudo conseguimos idealizar uma arquitetura funcional para um sistema de raciocínio (Figura 4). Na arquitetura idealizada incluem-se vários módulos funcionais com as seguintes responsabilidades:

- **Sensorização** – Este módulo deteta a ocorrência de um ou mais eventos que pode despoletar o processo de raciocínio. Por exemplo, Mileo et al. (2013) conceberam o sistema *StreamRule*, que aplica o processo de raciocínio sobre fluxos de dados *web*. O primeiro contacto que os dados provenientes da *web*, em formato RDF (*Resource Description Framework*), têm com este sistema é através de um "sensor", que realiza processos de abstração e filtragem sobre os dados recolhidos. Para tal são utilizadas *queries* CQELS (*Continuous Query Evaluation over Linked Stream*) – extensão da linguagem *SPARQL* que é utilizada para manipular informação de ontologias – para o efeito. Estes dados são transformados

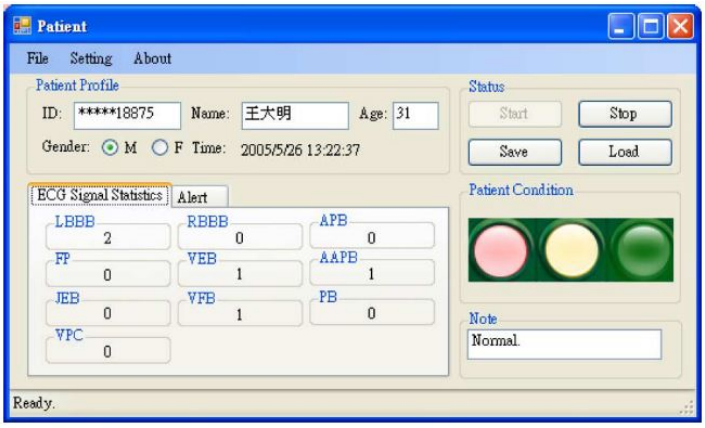

Figura 3: Janela de controlo da unidade de gestão - imagem extraída de Shih et al. (2010)

posteriormente em factos por uma camada intermédia, *Middle Layer Processor*, para serem utilizados como *input* do motor de raciocínio do sistema.

- **Reação** – Este módulo representa o resultado obtido do processo de raciocínio. Utilizando o exemplo apresentado anteriormente, o *output* do processo de raciocínio é representado pelo resultado do processamento efetuado pelo motor de raciocínio do sistema, *OClingo*. Este motor segue uma abordagem baseada em *Answer Set Programming* (Gebser et al., 2012).

- **Memória de trabalho** – Este módulo trata do "armazenamento de curto prazo e manipulação de informação sensorial" (Ma et al., 2014). Permite reter dados necessários para o processo de raciocínio (Baddeley, 2010). Apesar das características anteriormente mencionadas resultarem de uma análise no âmbito do raciocínio em animais, com grande incidência nos seres humanos, estas podem transpor-se para o contexto de um sistema de raciocínio. Sendo assim, a memória de trabalho de um sistema de raciocínio pode ser composta por vários componentes. Um destes pode efetuar a gestão da memória de trabalho enquanto que os outros fazem a ligação com a memória de longo termo. Esta última pode representar os casos anteriormente analisados ou, ainda, informação técnica que o sistema de raciocínio necessita para funcionar corretamente. Nesta informação destacam-se a linguagem a utilizar e a sua semântica, entre outras coisas. A memória de trabalho pode também ser utilizada para acumular a informação criada durante o processo de raciocínio, para permitir que este tenha um fio condutor entre as várias etapas.

- **Base de conhecimento** – Este módulo armazena informações factuais (Wang et al., 2018) e pode ser utilizado para recuperação de informação ou fornecimento de conhecimento estruturado (Socher et al., 2013). Como indicado por Grant e Wang (2015), as bases de conhecimento são repositórios compostos por factos ligados a um motor de inferência. Apesar de, neste exemplo, se ter mencionado a relação entre motor de inferência (Sharma et al., 2012) e a base de conhecimento, é possível efetuar uma associação análoga entre esta última e um motor de raciocínio. Isto deve-se ao facto de, à semelhança de um motor

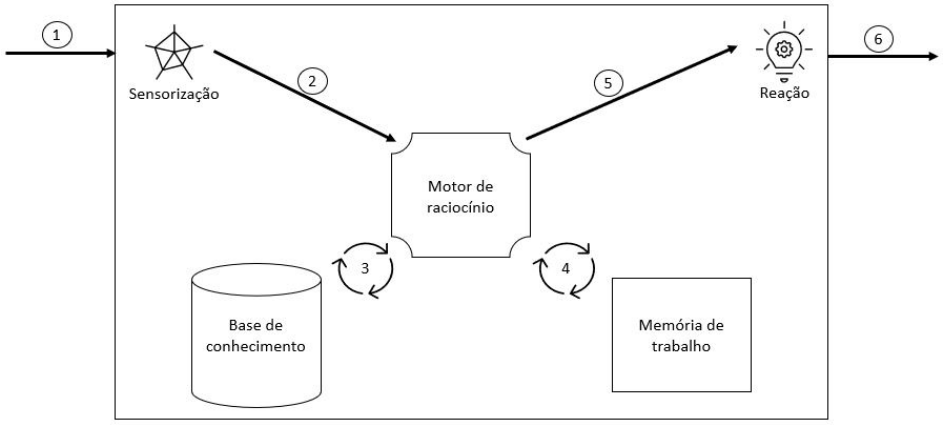

Figura 4: Arquitetura típica de um sistema de raciocínio

de inferência, um motor de raciocínio poder efetuar a geração de informação, tendo em conta o conteúdo de uma base de conhecimento.

- **Motor de raciocínio** – Este módulo pode ser considerado como um dos elementos centrais de um sistema de raciocínio, visto que pode utilizar algoritmos, regras ou outros mecanismos para fazer evoluir o sistema de raciocínio cognitivamente. Como foi referido no módulo anterior, é possível caracterizar um motor de raciocínio utilizando as características de um motor de inferência. Assim, de uma forma mais pormenorizada, é possível definir este módulo como um elemento que realiza um conjunto específico de processos de inferência, tendo em conta o conteúdo de uma base conhecimento.

De uma forma sucinta, estes módulos podem interligam-se da forma com está apresentada na Figura 4. Inicialmente o sistema de raciocínio deteta um ou mais eventos (1) através do módulo de sensorização. Depois, essa informação é utilizada como *input* do sistema de raciocínio, que por sua vez a transmite ao motor de raciocínio (2). Este, utilizando técnicas específicas que estejam na base do tipo de raciocínio que o modela, pode interligar-se à base de conhecimento (3) e à memória de trabalho (4) para efetuar o raciocínio sobre os dados de entrada.

A base de conhecimento fornece factos sobre os quais o motor infere informação. A memória de trabalho armazena outras informações que vão sendo utilizadas durante o processo de raciocínio. O resultado do processo de raciocínio pode ser visto como a reação (5) do sistema ao *input* que foi detetado inicialmente. O sistema torna-se mais rico em termos de conhecimento através dos dados que são inferidos pelo motor. Assim, o sistema de raciocínio obtém uma resposta (6) que pode ser utilizada por componentes externos, como por exemplo, uma *interface*. Isto acontece se o sistema de raciocínio for utilizado diretamente por humanos ou estiver embebido num sistema de maior dimensão.

2.2 MÉTODOS DE RACIOCÍNIO

As conclusões obtidas pelo sistema de raciocínio, relacionadas com o tipo de *input* detetado pelo módulo de sensorização, dependem do funcionamento do motor de raciocínio. Este módulo é o "núcleo" do sistema e dada a sua importância é necessário efetuar uma análise mais pormenorizada sobre este módulo. Mais concretamente, deve-se analisar as formas que estes elementos utilizam tipicamente para raciocinar e gerar nova informação, tendo em conta a base de conhecimento do sistema de raciocínio.

Assim, de seguida, apresentam-se os métodos de raciocínio que hoje podemos encontrar implementados em sistemas computorizados, mais concretamente aqueles que modelam os motores de raciocínio. Estes podem ser categorizados de acordo com as suas formas de raciocínio, nomeadamente:

- **Dedutivo** - Neste método de raciocínio não existe uma dependência lógica entre a conclusão obtida e as proposições (Goel, 2005). É apenas necessário que se encontrem premissas que sejam verdadeiras para que seja possível encontrar uma solução com o valor verdadeiro. No âmbito deste método de raciocínio, Shynkaruk e Thompson (2006) realizaram duas experiências para determinar a confiança e a precisão, bem como a relação entre ambas, em situações nas quais se aplicado raciocínio dedutivo. Numa das experiências, os participantes tinham à sua disposição um conjunto de testes. Para cada um dos testes existiam duas premissas e uma conclusão. Os participantes tinham de indicar se consideravam que a conclusão derivava corretamente das premissas. Cada teste era respondido duas vezes. Na primeira vez eram dados dez segundos para responder e na segunda cerca de um minuto. A segunda experiência é em grande parte semelhante à primeira, mas sem a restrição do tempo e com mais uma etapa na avaliação, na qual os participantes indicam, para cada teste, se estão confiantes na correção da sua resposta. Uma das conclusões que se pode retirar deste estudo relaciona-se com o facto de as conclusões passíveis de serem acreditáveis (*beliavable*) ou inacreditáveis (*unbeliavable*) serem um fator muito importante para as precisões das respostas dadas e para a confiança dos participantes, contrariamente aquelas que são consideradas neutras. Na primeira vez em que os testes são respondidos (relativamente à primeira experiência), os participantes utilizavam mais a sua intuição e o que acreditavam ser verdade. Já na segunda vez, tentavam justificar a resposta que tinham dado na tarefa anterior.

- **Indutivo** - Este método de raciocínio possui características opostas ao que apresentámos anteriormente, uma vez que a informação é inferida a partir de uma análise a casos específicos. No raciocínio dedutivo, a conclusão é necessariamente verdadeira, caso as proposições também o sejam, ao ponto que, no indutivo, as conclusões obtidas refletirem probabilidades da análise realizada aos casos de estudo (informações das premissas). Ou seja, as características que estes casos têm em comum tendem a ser generalizadas. A aplicação do raciocínio indutivo foi referida por Zalaghi e Khazaei (2016). Estes autores apresentaram o papel do raciocínio dedutivo e indutivo no âmbito económico. Relativamente ao raciocínio indutivo, é corroborado o que anteriormente foi indicado, uma vez que que os autores indicam que neste método, pela observação de casos específicos, seria possível inferir uma "lei ou princípio generalizado". Na prática, para criar uma lei mais direcionada à área económica, podem ser usados como dados de estudo os movimentos efetuados por várias empresas, como, por exemplo, o fluxo de dinheiro,

o capital das empresas, as vendas ou as receitas, entre outras coisas. A similaridade entre as premissas e a conclusão, bem como o conhecimento prévio (*background knowledge*) podem influenciar o processo de raciocínio baseado na indução (Hayes et al., 2010). O raciocínio indutivo pode representar a forma de raciocínio aplicada pelos seres humanos, dado que, muitas vezes, a experiência acumulada ao longo dos anos e a similaridade entre as situações lhes permitem chegar a conclusões mais facilmente e de uma forma mais instintiva.

- **Por analogia** - Este método de raciocínio possui semelhanças com o raciocínio indutivo, na medida em que o raciocínio é realizado tendo por base semelhanças entre casos específicos e não partindo de proposições, como no caso do raciocínio dedutivo. Suponha-se que, tendo-se dois casos X e Y com características em comum, assume-se que Y também terá uma dada característica que está presente em X. Este método de raciocínio pode ser também caracterizado por três estados principais e sequenciais entre si (Gentner e Maravilla, 2018): Mapeamento Analógico (*Analogical mapping*), Projeção de Inferências (*Inference projection*) e Avaliação (*Evaluation*). O primeiro destes refere-se ao estabelecimento dos pontos em comum entre dois casos, o caso de base e o caso *target*. O segundo estado é a projeção das inferências, ou seja, a indicação dos novos atributos que o caso *target* poderá ter, tendo em conta o conteúdo do caso base. O terceiro estado permite avaliar as projeções efetuadas anteriormente. Nesta avaliação, podem ser verificados os valores de verdade das projeções ou se estas correspondem aos objetivos do contexto em questão. No raciocínio por analogia, existem outros aspetos que podem ser relevantes, como aqueles que foram apresentados no estudo efetuado por Smaling (2003). De entre estes aspetos destacam-se os seguintes: quantas mais semelhanças existirem entre o caso de estudo e o caso *target*, mais robustas serão as inferências realizadas; o raciocínio é tão mais plausível quanto mais importantes forem as semelhanças entre o caso de estudo e o *target* para a conclusão obtida; e quantos mais casos sustentarem o raciocínio que está a ser realizado, mais conciso é o processo de raciocínio por analogia. Ou seja, se existirem bastantes casos que contenham os mesmos atributos que o caso de estudo e se aquilo que se pretende inferir para o caso *target* também tiver acontecido nesses casos, o processo de raciocínio fica muito bem suportado. Para além disto, se a conclusão a que se pretende chegar tiver muita probabilidade de acontecer, o raciocínio realizado possivelmente terá mais hipóteses de ser considerado válido.

- **Abdutivo** - Ao raciocinar de forma abdutiva, as conclusões que são obtidas representam a melhor explicação para o que poderá ter acontecido numa dada situação. Na abdução pode ser necessária a existência de "evidências dedutivas e derivadas indutivamente" para tornar uma conclusão abdutiva mais credível, como referido num estudo apresentado por Lipscomb (2012). Neste estudo foi também abordado o conceito de abdução por analogia. Ou seja, as formulações ou hipóteses aplicadas em casos anteriores podem ser usadas como base da inferência para um novo contexto. Por sua vez, Bhagavatula et al. (2020) apresentaram o primeiro estudo acerca "da viabilidade do raciocínio abdutivo baseado na linguagem". Neste estudo, foi analisada uma outra abordagem ao raciocínio abdutivo. A ideologia que sustenta este método de raciocínio mantém-se, ou seja, continua-se a indicar as melhores hipóteses que possam explicar um determinado evento. Contudo este procedimento não é estudado no âmbito dos

ambientes de lógica formal, mas sim em contextos narrativos. Para tal, são utilizados dois componentes de raciocínio: αNLI (*Abductive Natural Language Inference*) e αNLG (*Abductive Natural Language Generation*). O primeiro componente permite escolher, de forma automatizada, a hipótese que melhor explica um dado contexto, que é representado por um conjunto de observações. O segundo componente gera a hipótese que melhor explica um determinado contexto. Estes dois componentes trabalham sobre um dataset, designado por ART (*Abductive Reasoning in Narrative Text*), que contém um conjunto de observações (agrupadas em pares) do senso comum e um conjunto de hipóteses. Assim, estes componentes têm uma estrutura sólida que lhes permite serem "treinados" corretamente. Apesar destes componentes, na altura deste estudo, não terem uma *performance* excelente, é importante realçar que o raciocínio abdutivo pode ser aplicado em contextos mais próximos aos dos seres humanos, visto que estes utilizam muitas vezes a formulação de hipóteses para tentar explicar acontecimentos.

Para resumir a informação que utilizámos na apresentação e explicação dos diversos métodos de raciocínio, na Tabela 1 apresenta-se um resumo das suas características mais relevantes. O raciocínio dedutivo tem a si associado a propriedade da "Derivação", uma vez que as conclusões derivam das premissas de uma forma lógica/sustentada. Associou-se ao raciocínio abdutivo a propriedade "Explicação", visto que as conclusões representam explicações para uma determinada situação, para um determinado conjunto de premissas. No raciocínio indutivo e por analogia dá-se a "comparação" entre algumas das propriedades de contextos específicos, não generalizados, e as características do novo contexto.

	Raciocínio Dedutivo	Raciocínio Indutivo	Raciocínio por Analogia	Raciocínio Abdutivo
Derivação	✓			
Explicação				✓
Comparação		✓	✓	

Tabela 1: Comparação das características dos métodos de raciocínio.

2.3 DESENVOLVIMENTO DE UM SISTEMA DE RACIOCÍNIO

O desenvolvimento de um sistema de raciocínio engloba a criação, interligação dos módulos apresentados anteriormente na Figura 4 e a (possível) integração do sistema num sistema de maior dimensão. A sequência dos passos que constituem este desenvolvimento, bem como a apresentação dos constituintes exatos de cada um dos módulos varia bastante, devido ao facto de existirem diferentes tipos de sistemas de raciocínio, que utilizam metodologias e técnicas de implementação específicas. Contudo, é possível indicar aspetos recorrentes no desenvolvimento de um sistema de raciocínio, que foram tidos em consideração na parte prática dos trabalhos desta dissertação. Para formular estes aspetos suportamo-nos na ideologia de ciclo de vida do desenvolvimento de um sistema inteligente, abordada por Guida e Tasso (1989). Esta ideologia, apesar de ser descrita no âmbito de sistemas inteligentes, pode ser associada aos sistemas de raciocínio. Tal como mencionado pelos autores, um sistema inteligente é "um sistema de *software* que consegue fornecer desempenho especializado na res-

olução de problemas num determinado domínio de competência, explorando uma base de conhecimento e um mecanismo de raciocínio". Um sistema de raciocínio também apresenta estas características, daí a associação desta idealogia a ambos os conceitos.

Os autores conceberam este ciclo de desenvolvimento com o intuito de fornecer uma base sólida para a criação de sistemas inteligentes. Este ciclo é composto por cinco fases, que são categorizadas da seguinte forma:

- **Fase 1 - Estudo de plausibilidade –** representa a análise à "viabilidade técnica" do sistema, "disponibilidade dos recursos necessários", como por exemplo *software* e *hardware*, entre outras coisas mais. Como resultado desta fase é gerado um *plausibility report* (relatório de plausibilidade), que congrega as tarefas realizadas até um determinado momento, bem como pode disponibilizar um guia/plano para o desenvolvimento do sistema. De realçar que este relatório pode ser atualizado ao longo das fases subsequentes.

- **Fase 2 - Construção do protótipo de demonstração –** refere-se à criação do primeiro "protótipo limitado do sistema", designado por *demonstration prototype* (protótipo de demonstração) ou "demonstrador". Esta criação pode ter vários objetivos, como por exemplo compreender melhor o contexto onde o sistema está inserido ou "ganhar envolvimento dos especialistas e dos utilizadores no projeto". Ou seja, permite compreender como o "futuro sistema" se irá comportar em determinadas condições.

- **Fase 3 - Construção do protótipo completo –** representa a criação de um sistema mais completo do que o da fase anterior. As (eventuais) revisões aplicadas no relatório de plausibilidade são tidas em consideração nesta fase, o que permite melhorar a qualidade do sistema. Contudo este protótipo ainda não representa a versão final, visto que ainda não se encontra otimizado e implementado em ambiente real. Ou seja, representa a última versão do sistema, em ambiente de desenvolvimento, que considera todos os aspetos relevantes do relatório de plausibilidade.

- **Fase 4 - Implementação e instalação do sistema objetivo (*target*) –** representa a implementação do sistema protótipo, devidamente otimizado e testado com dados reais, num ambiente real. Caso o ambiente de desenvolvimento e o ambiente real sejam demasiado díspares, "do ponto de vista de *hardware* e *software*", deverá ser realizada uma nova implementação do protótipo. Caso contrário, apenas será necessária a realização de alguns "refinamentos" para implementar o sistema no ambiente real.

- **Fase 5 - Operação, manutenção e extensão –** refere-se à manutenção e monitorização do sistema. Tal como indicado pelos autores, esta fase "tem uma importância fundamental por assegurar uma utilização longa e efetiva e por obter benefícios significativos de todo o projeto". De realçar que esta fase estará presente durante o tempo em que o sistema estiver operacional.

De realçar que estas fases são sequenciais e precedidas por uma análise, demominada por *opportunity analysis* (análise de oportunidade), que consiste em estudar quais as áreas de aplicação onde o desenvolvimento destes sistemas poderia ser encaixado. Este processo torna-se relevante para fornecer bons *inputs* para as fases iniciais do ciclo de desenvolvimento.

Dado o ciclo de desenvolvimento apresentado anteriormente, é possível indicar que o desenvolvimento de um sistema de raciocínio vai além da criação e integração de módulos. É também necessária uma análise prévia ao tipo de contexto onde o sistema será inserido. Para além disto, a constante manutenção do sistema é igualmente relevante. Por exemplo, se o volume de informação aumentar bastante ao longo do tempo, os constituintes dos módulos do sistema terão de ser alterados para tornar o processo de raciocínio mais eficiente.

2.4 TIPOS DE SISTEMAS DE RACIOCÍNIO

As técnicas utilizadas pelos sistemas de raciocínio dependem do contexto onde estes são inseridos. Desde a aprendizagem de casos à resolução de problemas matemáticos, pode-se aplicar várias técnicas nestes sistemas. Isto permite agrupar os sistemas de raciocínio em vários tipos, que podem ser categorizados da seguinte forma:

- ***Constraint solvers*** - Este tipo de sistemas tem no seu cerne um componente capaz de solucionar problemas matemáticos representados por restrições, cujas variáveis aí contidas podem pertencer a domínios variados. Estes sistemas podem ser usados, por exemplo, para a testagem de programas de *software* (Malburg e Fraser, 2011) ou para a verificação automática de invariantes (Bharadwaj e Sims, 2000). No primeiro caso (no âmbito da testagem de programas de *software*), foi realizado um estudo com o objetivo de analisar "a tarefa de geração de *inputs* adequados para programas utilizando o código-fonte do programa", combinando a técnica de teste baseada em pesquisa (*Search-based*) e a técnica de teste baseada em restrições (*Constraint-based*). A primeira pode ser entendida como uma técnica que "faz evoluir uma população inicial de soluções candidatas para satisfazer qualquer critério de cobertura escolhido". A segunda pode ser entendida como uma técnica que "gera dados de teste através da resolução de restrições produzidas por execução de símbolos". A junção destas técnicas permite aumentar a área de testagem do código e colmatar as lacunas inerentes a cada uma delas. De entre os componentes utilizados na implementação destas técnicas encontra-se o *constraint solver Choco* (Jussien et al., 2008), que efetua o cálculo das restrições matemáticas presentes no código. No segundo caso é apresentado um mecanismo designado por *Salsa*. Este mecanismo verifica invariantes num dado sistema. Dos constituintes deste mecanismo encontram-se os "procedimentos de decisão para lógica proposicional, a teoria das enumerações não ordenadas e a aritmética linear de inteiros". É neste último domínio que é referido o *constraint solver*. Este calcula a viabilidade de um conjunto de restrições matemáticas no domínio de números inteiros. Para resolver estas restrições, o *constraint solver* recorre a autómatos.

- ***Theorem provers*** - Estes sistemas são utilizados como suporte de prova da validade de teoremas (suposições ou asserções). Nesta prova podem ser usados vários mecanismos que incluem a inferência de informação, cálculos matemáticos, a utilização de algoritmos, entre outras coisas. Um exemplo de um sistema deste tipo foi apresentado por Korovin (2008): o sistema *iProver*. Este sistema tem como principal objetivo verificar se um conjunto de cláusulas de lógica de primeira ordem são satisfatórias (que são provadas como válidas) ou insatisfatórias (que são provadas como inválidas). Neste processo é efetuada uma abstração ao conjunto inicial das cláusulas e verifica-se se se pode concluir se estas são ou

não insatisfatórias. Caso o sejam, o processo de prova termina. Caso contrário, à abstração efetuada anteriormente, são acrescentadas instâncias. Para tal, são utilizados os cálculos do sistema de inferência *DSInst-Gen* e também é eliminada a informação considerada redundante. Toda esta informação "atualizada" é analisada por um algoritmo designado por *Inst-Gen*, que verifica se as cláusulas são ou não satisfatórias. Um outro exemplo deste tipo de sistema foi apresentado alguns anos antes por Schulz (2002): o sistema *E*. É importante realçar que este sistema, tal como o anterior, tem um mecanismo que elimina a informação redundante e infere novas cláusulas. Para além disto, o algoritmo implementado para efetuar a prova é designado por *DISCOUNT*. Este algoritmo possui algumas semelhanças com o *Inst-Gen*, visto que existe uma separação entre as cláusulas inferidas e editadas (aquelas que tinham informação redundante) daquelas consideradas "passivas". Como cláusulas "passivas" entenda-se cláusulas que ainda não tinham sido utilizadas no processo de inferência. Com o auxílio deste algoritmo, é possível verificar se cláusulas são ou não satisfatórias. Este procedimento é realizado até ser atingido um estado no qual é possível obter uma conclusão acerca da validade ou não validade do conjunto inicial de cláusulas.

- ***Rule engines*** – Estes sistemas realizam o processo de raciocínio com base num conjunto de regras lógicas (ex: $a \wedge b \implies c$). Em algumas situações, este processo é estruturado em três subprocessos principais, sequenciais entre si, nomeadamente: 1) agrupamento das regras que podem ser utilizadas no contexto do sistema de raciocínio; 2) filtragem, do conjunto de regras resultante do subprocesso anterior das regras que melhor se adequam à inferência que se pretende realizar num dado momento do processo de raciocínio; 3) execução das regras mencionadas anteriormente. Estes subprocessos podem repetir-se várias vezes, permitindo que o sistema de raciocínio se torne mais rico em termos de conhecimento. Um exemplo de um sistema baseado em regras é o sistema *Jess*. Este sistema é composto por três componentes principais (O'Connor et al., 2005): 1) um conjunto de regras, 2) um conjunto de factos e 3) um motor que executa as regras. Um dos objetivos do estudo apresentado por estes autores consistia na integração do sistema *Jess* num ambiente onde era possível editar e construir regras que manipulavam ontologias OWL (*Ontology Web language*). Novos indivíduos e relações de uma ontologia podiam ser inferidos, recorrendo à execução do sistema *Jess*. Para tal, é necessário, primeiramente, converter os conceitos da ontologia, bem como as respetivas regras, para o formato deste sistema. Numa fase posterior, os resultados obtidos da inferência eram convertidos no sentido inverso, ou seja, para o formato OWL. Tendo isto, seria possível ir melhorando a ontologia em termos qualitativos e quantitativos.

- ***Machine learning systems*** - Estes sistemas aplicam técnicas de *machine learning*, ou seja, técnicas de aprendizagem automática que lhes permitem evoluir cognitivamente consoante um conjunto inicial de valores (*dataset*). De entre estas técnicas encontram-se a regressão, a classificação, a seleção de "features", ou a redução de dimensionalidade, entre outras, como indicado por Kraska et al. (2013), num estudo no qual explicitou uma das primeiras versões do sistema *MLbase*. Este sistema pretende, entre outros coisas, tornar as tarefas inerentes a *machine learning* mais acessíveis aos utilizadores. É possível aplicar um variado conjunto de algoritmos a um conjunto de dados por forma a "explorar" o significado e a relação entre os atributos aí contidos, para assim serem realizadas as previsões necessárias. Um

exemplo de um sistema de *machine learning* é o *OpinionMiner* (Jin et al., 2009). Este sistema permite automatizar a mineração de opiniões dadas num ambiente *web* por clientes sobre determinados produtos. Os textos que contêm as opiniões são previamente filtrados de páginas HTML (*Hypertext Markup Language*), de onde também se retira conteúdo indesejado, por exemplo identificadores (*tags*) HTML e anúncios, entre outras coisas. No processo de mineração, o sistema possui uma fase de treino e uma fase de avaliação. A primeira destas fases pode ser dividida em duas subfases. Numa delas é efetuado, entre outras coisas, o processo de identificação híbrida (*hybrid tagging*). Neste processo, caracterizam-se as palavras contidas nos textos das opiniões dadas pelos clientes. Nesta caracterização é atribuída uma categoria (característica física ou não física do produto, funcionalidades do produto ou opinião expressa por um cliente) às palavras e indica-se se estas são entidades independentes ou se são componentes de uma entidade. Para além disto, realiza-se também uma associação de sinónimos e de antónimos às palavras das opiniões, por exemplo. Numa outra subfase, o processo de caracterização das opiniões é automatizado, sendo designado por *Bootstrapping*. Tendo isto, o sistema pode aprender com os resultados obtidos das duas subfases anteriores e inferir a orientação das opiniões, sendo atribuídas os melhores identificadores híbridos (*hybrid tags*) recorrendo ao algoritmo *Viterbi*. A orientação das opiniões é um processo que permite atribuir um valor de caráter, positivo ou negativo, a cada opinião.

- ***Case-based reasoning systems*** - Este tipo de sistema utiliza eventos passados para solucionar um problema atual. Como se vai poder observar mais adiante nos trabalhos desta dissertação, estes sistemas são modelados por um paradigma de raciocínio designado por *case based reasoning* (CBR), que significa raciocínio baseado em casos. Aamodt e Plaza (1994) foram dos primeiros a caracterizar este paradigma. Indicaram que o funcionamento deste é composto por quatro etapas principais: recolha (*retrieve*), reutilização (*reuse*), revisão (*revise*) e retenção (*retain*). Estas fases representam todo o processo que vai desde a recolha de casos/experiências passadas, semelhantes ao caso que se pretende solucionar, até ao momento em que se resolve o novo problema. O sistema *BTCBRsys* (Gu et al., 2017) é um exemplo de um *case-based reasoning system* que auxilia oncologistas no processo de diagnóstico do cancro da mama, "complementando o seu próprio conhecimento experimental". Um caso do sistema é composto pelas informações clínicas de um determinado paciente e pelo diagnóstico que lhe foi realizado. Quando surge um novo caso, os oncologistas podem indicar as características deste no sistema e selecionar os diagnósticos passados que mais se assemelham com o novo caso. Em fases posteriores adapta-se, se necessário, algumas das características do caso mais similar para se adaptar ao novo problema e obtém-se o diagnóstico previsto pelo sistema. Os oncologistas adquirem assim mais conhecimento e uma maior segurança para realizarem os diagnósticos de cancro da mama. O repositório onde tipicamente se guardam os casos de um sistema baseado em casos designa-se por *case base* (base de casos). A decisão acerca da inclusão de um novo diagnóstico neste repositório é da responsabilidade dos oncologistas, dependendo da qualidade e necessidade deste.

- ***Procedural reasoning systems*** – Este tipo de sistema tem um mecanismo de raciocínio que consiste na resolução de problemas recorrendo a planos. Ou seja, para um determinado evento, o sistema executa um plano previamente analisado e associado ao respetivo problema do contexto. Mais concretamente,

como se pode verificar no estudo efetuado por Begoli (2014), este tipo de sistema pode funcionar numa ideologia semelhante à do modelo BDI (*Belief-Derire-Intentions*), no qual três componentes principais "guiam" o sistema no processo de tomada de decisão. O primeiro componente representa o conjunto de factos de que este tem conhecimento e que vai acumulando ao longo das várias inferências/resolução de problemas. O segundo componente considera os objetivos do sistema de raciocínio, ou seja, aquilo que o sistema pretende alcançar no final das inferências. Por fim, o terceiro componente pretende representar as formas como se podem alcançar os objetivos, recorrendo a conjuntos de ações. Um exemplo de um sistema de raciocínio baseado em procedimentos pode ser encontrado num estudo desenvolvido por Mousavi et al. (2010). Nesse estudo é apresentado o sistema *O-PRS*. Este sistema age conforme o modelo apresentado anteriormente. Contudo a informação de que o sistema necessita em cada uma das fases está representada por uma ontologia. Das classes da ontologia fazem parte a classe *Believe* (Crença), *Event* (Evento) e *Plan* (Plano). O sistema é composto por um conjunto de agentes que comunicam entre si através de mensagens. Isto possibilita a execução de um plano de forma mais automatizada, quando ocorre uma mudança no sistema. A consulta da informação representada pela ontologia é realizada através de queries *SPARQL*.

2.5 APLICAÇÃO DE SISTEMAS DE RACIOCÍNIO NA ÁREA DO ENSINO

Uma das áreas de maior aplicação dos sistemas de raciocínio é a área do ensino. Isto deve-se ao facto de existir a necessidade de automatizar alguns dos processos realizados pelos humanos neste domínio de actividade. Usualmente, estes processos são muito dispendiosos, quer em termos de tempo, quer em termos de dinheiro e, por vezes, não conseguem acompanhar regularmente os estudantes no seu processo de aprendizagem.

Na área do ensino, existem várias categorias que caracterizam a aprendizagem dos estudantes para lá da sala de aula. Esta aprendizagem permite que os estudantes tenham acesso constante a material de estudo (para além do material físico), como também lhes permite monitorizar regularmente o seu conhecimento numa dada área de estudo. Entre estes conceitos encontram-se *Distance learning*, *eLearning* e *Online learning* (Moore et al., 2011). É possível definir a primeira destas categorias como o processo de aprendizagem que ocorre por meio de uma distância física e temporal. Neste processo estão envolvidos dois intervenientes (quem ensina e quem aprende). Já a segunda categoria é relativa aos ambientes de aprendizagem suportados por ferramentas digitais/informáticas, desde aplicações informáticas, a *websites*, entre outras coisas mais. Estes ambientes suportam o ensino dos estudantes (Wastiau et al., 2013). *Online learning* pode ser entendido como uma atualização do conceito *Distance learning* e ao mesmo tempo mais pormenorizado, uma vez que tende a caracterizar os ambientes de aprendizagem num contexto específico, o contexto *online*.

Existem alguns sistemas que exemplificam o estudo para lá da sala de aula, como por exemplo o *Blackboard Learn* (Blackboard, 2021) e o *Moodle LMS* (Moodle, 2021). Apesar de por vezes estes sistemas serem integrados numa dada categoria, nesta dissertação optamos por integra-los todos na mesma categoria: *eLearning*.

O sistema *Blackboard Learn* disponibiliza diversas funcionalidades aos seus utilizadores, mais concretamente a professores e estudantes. Uma das grandes vantagens do *Blackboard Learn* é a facilidade de acesso, visto

que os utilizadores podem aceder a partir de vários dispositivos. Os professores podem disponibilizar material de estudo, como por exemplo testes de anos anteriores, exercícios de uma determinada matéria lecionada, por forma a permitir aos estudantes complementar o seu estudo. Este sistema está também associada a uma ferramenta para videochamadas, designada por *Blackboard Collaborate Ultra*, o que permite um maior contacto e facilidade de comunicação entre alunos e professores.

O sistema *Moodle LMS* é bastante semelhante ao *Blackboard Learn*. Para além da disponibilização de material de estudo e da facilidade de acesso, este sistema, tal como anteriormente apresentado, possibilita a realização de testes e o envio de documentos. O envio de documentos é importante para facilitar a entrega de trabalhos de determinadas unidades curriculares.

Uma das formas de se incluírem os sistemas de raciocínio nesta aprendizagem paralela aos estabelecimentos de ensino é desenvolvê-los no contexto de sistemas de avaliação, mais concretamente em tutores inteligentes. Grande parte dos tutores inteligentes são ferramentas informáticas que permitem acompanhar regularmente o processo de aprendizagem dos estudantes, fornecendo-lhes ajuda e informação relevante acerca do seu desempenho. Um exemplo de um tutor inteligente é o sistema *Leonardo* (Belo et al., 2019). Este tem um mecanismo de raciocínio capaz de se adaptar aos perfis de conhecimento revelados pelos alunos em diversas áreas de conhecimento. Consoante as respostas que cada aluno dá para cada questão apresentada pelo sistema, o sistema atualiza o perfil do aluno envolvido tendo em conta vários parâmetros, como, por exemplo, o domínio escolhido, a questão apresentada, o grau de dificuldade da questão, a resposta que foi dada, o tempo que demorou a responder à questão, entre outras coisas mais. Com base nessa informação, o sistema consegue determinar, de acordo com o perfil estabelecido para o aluno, qual a próxima questão a apresentar ao aluno.

Luckin (2017) apresentou um estudo sobre inteligência artificial e a associação desta num contexto educativo. Neste estudo é abordado o sistema de avaliação *AIAsses*, cujo funcionamento assenta em dois aspetos fundamentais: quantidade de ajuda fornecida aos utilizadores para completarem as tarefas que lhes são propostas e a consciência dos utilizadores relativamente à ajuda que consideram necessitar para completarem as tarefas. De uma forma mais concreta, este sistema é constituído por três componentes principais e interligados entre si. O primeiro contém todas as informações acerca das áreas de estudo, permitindo acompanhar o aluno em cada uma das etapas de uma dada resposta. O segundo componente acumula as várias respostas que são dadas, o tipo de ajudas que o aluno teve, entre outras coisas mais. Por fim, o terceiro componente agrupa a informação dos dois anteriores e avalia o conhecimento potencial e a consciência metacognitiva do estudante. Este componente está sempre disponível para mostrar o resultado da prestação do aluno num dado momento da avaliação. Ou seja, é possível indicar que o processo de raciocínio deste sistema de avaliação consiste, entre outras coisas, em analisar as interações que os utilizadores têm com o sistema de avaliação com o intuito de os ir avaliando formativamente.

3

AVALIAÇÃO BASEADA EM CASOS

3.1 DEFINIÇÃO E PROCESSO

Um dos objetivos dos trabalhos desta dissertação foi o desenvolvimento de um sistema de raciocínio baseado em casos especificamente para um sistema de avaliação. Como tal, foi importante abordar em maior detalhe o mecanismo de raciocínio que modelasse um sistema deste tipo. Mais concretamente irá abordar-se nesta secção o CBR e as suas aplicações, em particular, no domínio dos sistemas de avaliação.

O CBR foi estudado e desenvolvido por vários autores ao longo do tempo. Como tal existe um leque de trabalhos muito interessante e diversificado nesta área. Aamodt e Plaza (1994), por exemplo, definiram o CBR como um paradigma para a resolução de problemas que "utiliza conhecimento específico de situações experienciadas anteriormente". Por sua vez, de Mantaras (2001) indicou que neste paradigma as soluções não são criadas de raiz, "from scratch". Em vez disso, refere que se analisam as semelhanças entre o problema que temos em mãos e os problemas anteriores e se adaptam as soluções aplicadas nestes últimos para o problema atual. Watson e Marir (1994) também se reveram nessa definição de CBR.

As várias definições aplicadas ao CBR tendem a convergir para a definição de um mecanismo que tem em consideração a relação entre momentos passados, as suas características e a forma como se resolveram os problemas aí encontrados. Alguns autores referem que por vezes se indica que o CBR e o raciocínio por analogia (*analogical reasoning*) são semelhantes. Contudo, como indicado por Aamodt e Plaza (1994) e Richter e Aamodt (2005), existem diferenças entre estes dois conceitos, visto que o CBR atua dentro de um determinado domínio, enquanto que o raciocínio por analogia atua tipicamente entre vários domínios.

Um dos aspetos mais importantes do CBR são os casos. Como indicado por Main et al. (2001), casos "são registos de experiências passadas ou problemas". Já Watson e Marir (1994) definiram um caso como uma "peça contextualizada de conhecimento que representa uma experiência". Em concreto, um caso contém normalmente informações do problema que "descreve o estado do mundo quando o caso ocorreu", da solução utilizada para resolver o problema ou do resultado que "descreve o estado do mundo após o caso ocorrer".

O funcionamento típico de um sistema de CBR pode ser caracterizado por um ciclo, dividido am quatro fases: recolha (*retrieve*), reutilização (*reuse*), revisão (*revise*) e retenção (*retain*). Este ciclo pode ser observado na Figura 5.

Ao analisar pormenorizadamente o ciclo de CBR, é possível observar que o processo de raciocínio se inicia quando um novo problema é detetado. O CBR trata este problema como um novo caso que necessita de

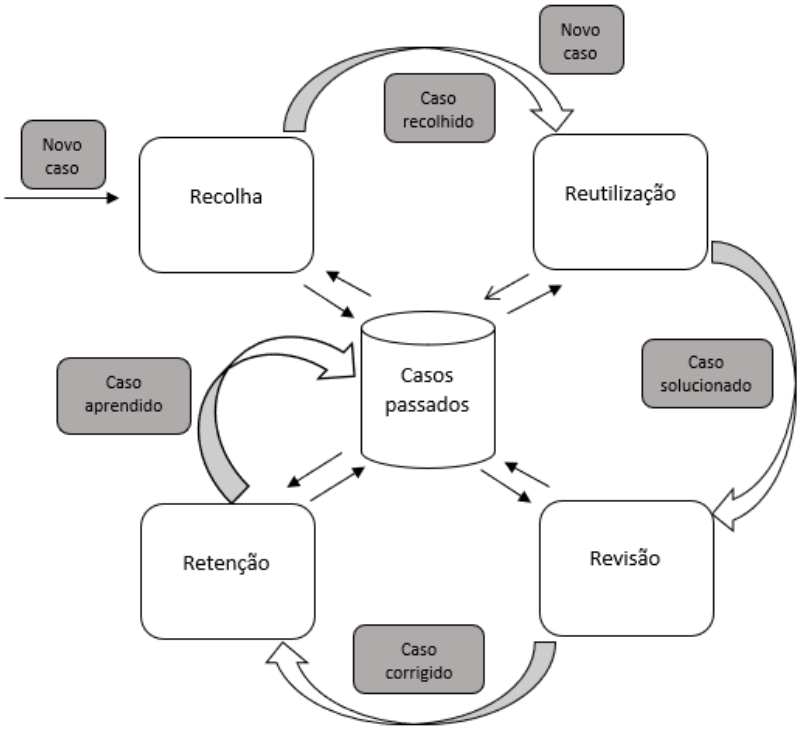

Figura 5: Ciclo CBR - figura adaptada de Aamodt e Plaza (1994)

"verificação". Na fase de recolha dá-se a seleção do "caso recolhido" apresentado no ciclo, que é o melhor *match* com o novo caso. Ou seja, nesta fase, filtra-se do conjunto de experiências passadas o caso que tem um maior número de características em comum com o novo caso. De realçar que, podem ser considerados vários fatores na verificação da similaridade entre casos, algo que depende da forma como estes estão representados.

Na fase de reutilização, a solução do "caso recolhido" é copiada ou adaptada ao novo caso, dependendo da existência de muitas ou poucas semelhanças entre os casos, respetivamente. Esta fase permite obter uma primeira proposta de solução, designada por solução sugerida (*Suggested Solution*). Nesta fase o caso inicial muda o seu estado, passando a ser designado por "caso solucionado".

Na fase de revisão verifica-se a existência de erros na solução proposta na fase anterior. Como indicado por Aamodt e Plaza (1994), a revisão e a avaliação de uma solução é normalmente feita em ambiente real, mais concretamente através de intervenção humana. Importante referir que o caso inicial muda mais uma vez de estado, de "caso solucionado" para "caso corrigido".

A última fase representa a junção da informação obtida no ciclo com o conhecimento adquirido *à priori*. Desta junção destaca-se a associação do "caso aprendido" aos "casos passados", que permite reforçar o conjunto de experiências passadas que são utilizadas nas iterações de um ciclo.

De realçar os estados que uma solução e um caso podem ter, que culminam na obtenção do "caso aprendido". Isto comprova tudo o que foi anteriormente dito acerca do CBR, uma vez que este paradigma de raciocínio analisa e adapta as estratégias de resolução utilizadas em eventos passados para compreender novos problemas.

O sistema apresentado por Gu et al. (2017), que foi abordado na Secção 2.4, engloba as quatro fases do ciclo apresentado anteriormente. Seleciona-se do conjunto de casos de cancro da mama aquele que mais se assemelha com o diagnóstico que é necessário realizar. Tal como no ciclo apresentado nesta secção, da fase de reutilização e revisão resultam a solução sugerida (*Suggested solution*) e solução confirmada (*Confirmed solution*), respetivamente. De realçar que apenas alguns casos – que contêm algumas das características do paciente a quem vão ser associado o diagnóstico - são inseridos na base de casos (*case base*). Como mencionado pelos autores, "apenas aqueles considerados de alta qualidade, valor e necessidade pelos oncologistas serão retidos na base de casos".

3.2 UTILIDADE E APLICAÇÃO

Através da análise a alguns trabalhos realizados no âmbito do CBR é possível observar que este paradigma pode ser utilizado num vasto conjunto de situações. Contudo estas situações têm de ter características específicas para que a aplicação do CBR seja útil. Como indicado por Main et al. (2001) – suportados pelos trabalhos de Kolodner (1993), Kolodner (1992) e Kolodner e Mark (1992) - a utilização do CBR é útil em problemas e domínios em que se observem:

- **Exceções ou novos casos** – Em domínios que não tenham uma ocorrência frequente de novos casos ou casos que sejam uma exceção à regra, é preferível utilizar outro tipo de paradigmas, como por exemplo raciocínio baseado em regras.
- **Benefícios na adaptação de soluções passadas** – Se a adaptação de uma solução aplicada num caso passado, comparativamente com a criação de uma questão de raiz, não permitir uma melhoria significativa em termos de processamento e de tempo gasto, a aplicação de CBR não é útil.
- **Modelos subjacentes** - A ocorrência aleatória de processos é um fator que influencia a não utilização de CBR. Se não existir uma dependência percetível entre as causas da correta ou incorreta aplicação de soluções para um determinado problema, o raciocínio baseado em casos torna-se "fútil".
- **Recorrência de casos** - Se num determinado domínio for muito pouco provável a ocorrência de casos repetidos ou de casos muito semelhantes, a aplicação de CBR acrescenta muito pouco valor nesse domínio. Isto relaciona-se com um dos fundamentos do CBR, que é que a adaptação de estratégias utilizadas em casos anteriores para se solucionarem novos problemas.
- **Possibilidade de recolha de informação relevante** – Se não for possível recolher os dados necessários para representar as características mais relevantes dos problemas e das respetivas soluções, as tarefas de análise e adaptação de soluções para os novos problemas tornam-se complicadas.

De realçar que, como indicado por Main et al. (2001) , a ocorrência simultânea destes fatores possibilitará que o "raciocínio baseado em casos seja aplicável e relevante".

Os pontos apresentados anteriormente caracterizam, de uma forma geral, os domínios nos quais tipicamente se aplica o CBR. Por forma a reforçar estes pontos apresentam-se de seguida dois exemplos destes domínios, a área da saúde e a área do ensino.

Na área da saúde o diagnóstico de problemas é útil na prevenção de doenças graves. Estes diagnósticos dependem do histórico de saúde dos pacientes, dos sintomas que estes apresentam ao longo do tempo ou do tipo de exames clínicos que realizaram. A realização destes diagnósticos é tipicamente realizada por profissionais de saúde, como, por exemplo, os médicos. A utilização de um paradigma baseado em casos como um programa de *software*, poderia melhorar o processo de realização de diagnósticos. Este processo seria mais célere e a deteção de anomalias tornar-se-ia mais precisa, visto que por vezes alguns erros escapam à análise humana.

Na área do ensino, por vezes, a avaliação tradicional dos alunos não é muito assertiva e não tem em grande consideração a evolução do conhecimento dos alunos ao longo do tempo. Para além disto, normalmente os professores não são capazes de acompanhar o processo de estudo dos alunos de forma constante. Para solidificar a avaliação dos estudantes e colmatar estas lacunas, poderia optar-se por abordagens alternativas, como, por exemplo, pela utilização de sistemas de avaliação, mais concretamente de sistemas de tutoria inteligente. O CBR poderia ser incluído nestes sistemas para lhes permitir fornecer o material de estudo de forma adaptativa, tendo em conta as interações entre os alunos e os sistemas. Assim, os sistemas de avaliação apresentariam o mesmo material de estudo a alunos com as mesmas capacidades cognitivas.

3.3 EXEMPLOS DE APLICAÇÃO

Como vimos anteriormente, o CBR pode ser aplicado em vários contextos. Contudo, como os trabalhos desta dissertação se encontram no âmbito da área do ensino, realizou-se um estudo mais pormenorizado de exemplos de aplicação de CBR nesta área em particular, mais concretamente em sistemas de avaliação. Os exemplos escolhidos permitem demonstrar formas distintas de representação dos casos na base de conhecimento de sistemas inteligentes. Para além disto, demonstram a utilização de CBR como elemento de fornecimento inteligente de respostas, de recomendação de material de estudo e de fornecimento de *feedback* acerca das respostas corretas e incorretas dos utilizadores dos sistemas de avaliação. De seguida apresentam-se os exemplos que selecionámos no âmbito desta dissertação.

3.3.1 *(Azeta et al., 2009)*

Azeta et al. (2009) apresentaram um sistema inteligente de *eLearning* que permite aos seus utilizadores interagir com este através da voz. Este sistema foi desenvolvido com o objetivo de colmatar as dificuldades que os alunos com deficiências (visuais) sentiam ao interagir com sistemas de *eLearning*. A utilização de um paradigma baseado em casos permite fornecer a informação aos utilizadores de forma inteligente. Na Figura 6 podemos ver a esquematização (adaptada) do funcionamento do sistema de Azeta et al. (2009).

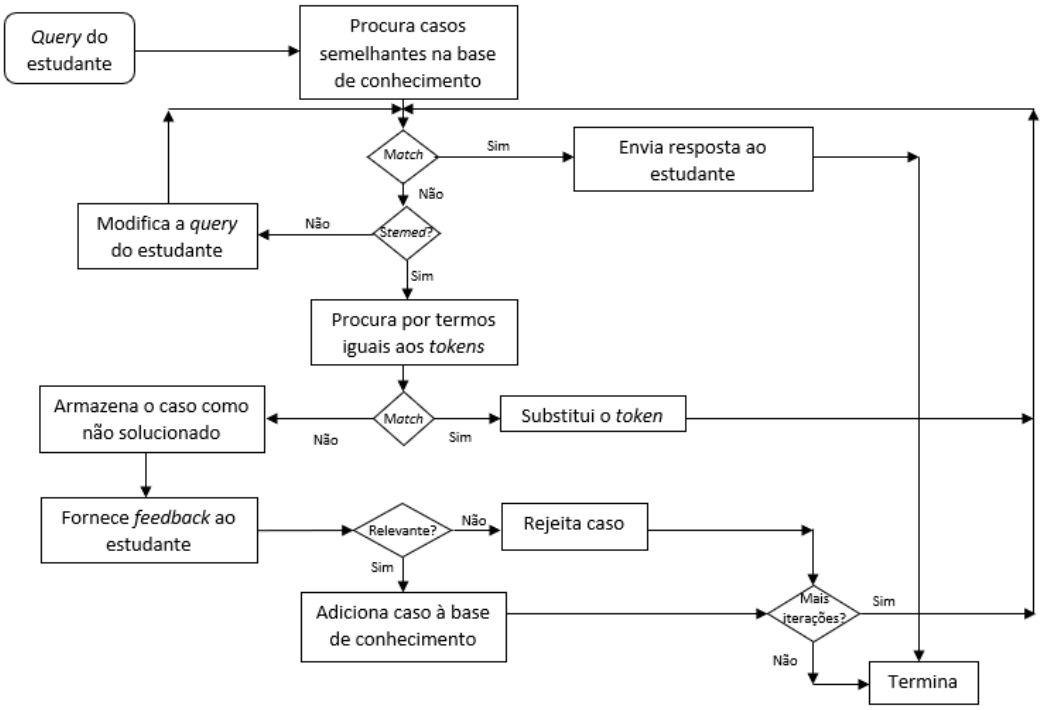

Figura 6: Diagrama do funcionamento do sistema de *eLearning* - figura adaptada de Azeta et al. (2009)

Analisando com maior pormenor o diagrama da Figura 6, podemos verificar que os utilizadores podem questionar o sistema através da submissão de *queries*. Caso tenha na sua base de conhecimento uma resposta associada a essas *queries*, o sistema informa os utilizadores através de uma resposta interativa por voz (ação '***Envia resposta ao estudante***'). Um caso da base de conhecimento deste sistema pode ser entendido como um agregado de termos - representativos de uma *query* - que tem uma solução associada. Por uma questão de eficiência na procura e retenção de casos passados, os termos do domínio de conhecimento são representados pelos seus *stems*, ou seja, pelas palavras de quais são originárias (por exemplo, o stem de 'cantou' é 'cantar'). Assim, reduz-se o número de termos que têm de ser mantidos pelo sistema.

Nas situações em que não existe uma correspondência na base de conhecimento com a *query* proferida inicialmente pelos utilizadores, esta sofre um conjunto de modificações. Estas modificações são relativas à ação '***Modifica a query do estudante***' do diagrama. Os constituintes da *query*, designados por *tokens*, são substituídos pelos seus *stems* até se encontrar uma correspondência com um caso da base de conhecimento.

Nos casos em que as modificações não permitam obter nenhuma correspondência, as *queries* são tratadas pelos professores associados ao sistema inteligente. Nestas situações, o aluno recebe o *feedback* através do *email* (ação '***Fornece feedback ao estudante***'). Se estas queries representarem informação relevante para o sistema, acrescenta-se um novo caso na base de conhecimento, como se pode verificar pela ação '***Adiciona caso à base de conhecimento***'.

3.3.2 *(Oliveira Neto e Nascimento, 2012)*

Oliveira Neto e Nascimento (2012) propuseram a implementação de um mecanismo de tutoria inteligente num curso de *Distance learning*, na área da matemática financeira. O principal objetivo deste sistema era avaliar formativamente os estudantes através do fornecimento de *feedback* acerca das respostas corretas, incorretas e ajudas para consultar material de estudo. Este trabalho, apesar de não ter sido abordado pelos seus autores no âmbito do CBR, segue os fundamentos deste paradigma de raciocínio, visto que o *feedback* fornecido numa determinada situação depende das experiências passadas dos utilizadores. De seguida descreve-se os pontos mais relevantes que permitem incluir este sistema no âmbito do CBR.

A base de conhecimento deste sistema representa as informações das interações dos estudantes ao longo dos vários anos do curso. Mais concretamente, os erros cometidos pelos estudantes são utilizados como "regras" no final de cada ano. A resposta correta, a resposta incorreta e o *feedback* a fornecer ao aluno (com uma explicação para o erro e um *link* para material de estudo) formam um conjunto (Tabela 2) que permite ao sistema saber o que fazer quando um aluno errar uma questão.

Questão	Resposta correta	Resposta incorreta	*Feedback* para o estudante	*Link* para a teoria
Determine a taxa de juros que faz o capital multiplicar [var1] vezes após [var2] anos	((([var1] - [1]) / [var2]) / [6]) * [100]	((([var1] – [1] / [var2]) / [6])	O valor da taxa deve ser calculado como uma percentagem. (Multiplicar o valor por 100)	https:...

Tabela 2: Associação de alguns elementos de estudo à resposta incorreta dada por um estudante - tabela adaptada de Oliveira Neto e Nascimento (2012).

Os elementos de estudo apresentados na Tabela 2 indicam que um estudante errou a questão com o enunciado 'Determine a taxa de…'. A resposta incorreta ("Fórmula errada") associada à questão é representada pela fórmula ((([var1] – [1]) / [var2]) / [6]). A resposta correta para esta questão é a fórmula representada pelo atributo "Fórmula certa". De realçar que, o *feedback* fornecido pelo sistema, caso algum aluno tenha já respondido incorretamente à questão e o conjunto da Tabela 2 já se encontrasse na base de conhecimento do sistema, seria aquele que apresentamos na Figura 7. O sistema indicaria ao utilizador que este deveria multiplicar a sua fórmula por 100, visto que a taxa é estimada através de uma percentagem.

É importante mencionar que a associação destes elementos e a sua consequente introdução na base de dados do sistema fica a cargo dos professores no final de cada ano do curso. Estas conjuntos são como uma espécie de 'casos' do sistema. Assim, o *feedback* fornecido aos utilizadores nos próximos anos terá em consideração os erros dos estudantes dos anos transatos. O *feedback* ocorre em situações em que os alunos cometem erros como naquelas em que acertam nas questões, "ajudando-os a consolidar o seu conhecimento através de material suplementar".

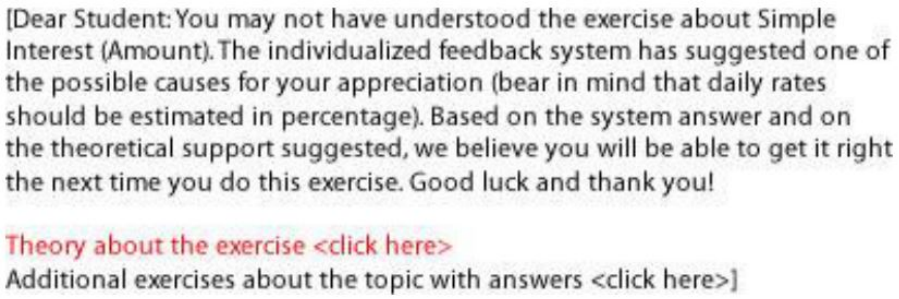

Figura 7: *Feedback* para uma resposta incorreta conhecida - imagem extraída de Oliveira Neto e Nascimento (2012)

O sistema de Oliveira Neto e Nascimento (2012) não segue ao pormenor o ciclo do CBR (Secção 3.1). Contudo, a ideia de utilizar experiências passadas para solucionar um determinado problema, que neste caso é o fornecimento de *feedback* para uma resposta de um aluno, fez com que o incluíssemos como um dos exemplos de aplicação de CBR em sistemas de avaliação.

3.3.3 *(Khamparia e Pandey, 2017)*

Khamparia e Pandey (2017) abordaram a integração de um mecanismo de raciocínio baseado em casos, juntamente com outras técnicas, num sistema de *eLearning*. O seu sistema foi direcionado para a aprendizagem de linguagens de programação na área das ciências da computação e fornece o material de estudo consoante as características dos utilizadores, ou seja, de forma adaptativa.

A informação do sistema é representada por casos, que caracterizam as relações existentes entre o género, o nível de ansiedade, personalidade, nível de aprendizagem dos estudantes, entre outras coisas mais. Os casos são gerados através de técnicas de mineração de dados. A aplicação destas técnicas foi sustentada por uma recolha prévia de informação, mais concretamente pela realização de testes a um conjunto de alunos. Estas técnicas permitiram assim relacionar as características dos estudantes, consoante os resultados obtidos nos testes, e representar esta informação em casos.

Na Tabela 3 é possível observar os casos gerados para um tipo específico de avaliação, 'SY', que corresponde ao estudo de deteção de erros de sintaxe numa linguagem de programação. Os atributos são representados na forma de *bits*, dependendo do número de possibilidades atribuídas a cada um. O género (*G*) pode ser representado por 'feminino' (01) ou 'masculino' (10). Para caracterizar o nível de ansiedade (*A*) existem os termos 'baixa' (001), 'média' (010) e 'alta' (100). O tipo de personalidade (*P*) pode ser representado por 'introvertido(a)' (10000), 'meio(a) introvertido(a)' (01000), 'neutro' (00100), 'meio(a) extrovertido(a)' (00010) e 'extrovertido(a)' (00001). O tipo de aprendizagem (*L*) caracteriza-se por 'pensamento e crença' (001), 'perceção de informação' (010) e 'ver e escutar' (100). A habilidade cognitiva (*C*) representa-se por 'field dependet' (01) e 'field independent' (10) e mostra se um estudante necessita ou não de fatores externos para aprender uma determinada matéria.

Casos	G	A	A	P	P	L	L	C	Resultado
		LB	UB	LB	UB	LB	UB		
1	01	001	NA	00001	NA	001	NA	00	M
2	01	001	NA	00010	NA	001	NA	00	H
3	10	001	NA	00010	NA	001	NA	00	M

Tabela 3: Exemplos de casos gerados para um tipo específico de avaliação do sistema de *eLearning* - tabela adaptada de Khamparia e Pandey (2017).

O caso número '3' (Tabela 3), por exemplo, indica que um homem cujo nível de ansiedade seja maior ou igual a 'baixo', nível de personalidade pelo menos igual a 'meia-extrovertida', valor mínimo de aprendizagem 'pensamento e crença' e sem estilo cognitivo definido, deverá receber material de estudo com o nível de dificuldade médio. Os valores 'NA' indicam que não foi possível gerar limites superiores (*UB*) para este caso, contrariamente aos limites inferiores (*LB*).

Na interação entre o sistema e o aluno, este indica o seu género, nível de ansiedade, personalidade, tipo de aprendizagem e habilidade cognitiva. O sistema cria um novo caso, através da junção dos *bits* que representam as características do aluno, e procura na *case base* o caso mais similar a este. Para realizar a comparação entre casos utiliza-se o operador XOR [1](*Exclusive Or*). O caso mais similar possibilita ao sistema obter o nível de dificuldade do material de estudo que deve apresentar ao aluno.

3.4 VANTAGENS E DESVANTAGENS

Nas secções anteriores apresentaram-se os pontos considerados mais relevantes para caracterizar o CBR. De realçar que aquilo que o CBR tem para oferecer depende bastante do contexto onde este é inserido. Caso seja aplicado em contextos com as condições apresentadas na Secção 3.2, apresenta mais vantagens do que desvantagens. Nesta altura, é importante que faça uma síntese dos pontos mais favoráveis e desfavoráveis do raciocínio baseado em casos.

Para abordar os pontos mais favoráveis do CBR suportamo-nos pelo trabalho realizado por Main et al. (2001), que por sua vez se suportaram nos trabalhos de Kolodner (1992), Kolodner e Mark (1992) e Ashley (1992). Esses autores indicaram que o CBR permite:

- **Reduzir o processo de aquisição de conhecimento** – As tarefas para aquisição de conhecimento têm menor complexidade quando comparadas com outro tipo de paradigma. Por exemplo, num paradigma baseado em regras seria necessário definir um conjunto de regras para que fosse possível criar novos dados, nova informação. No CBR, normalmente, a aquisição de conhecimento é sustentada apenas pela recolha de casos e consequentemente na sua representação e armazenamento num repositório comum, uma base de conhecimento.

1 XOR é um operador lógico que permite verificar a semelhança entre dois operandos. Caso estes sejam diferentes, retorna o valor lógico verdadeiro. Caso contrário, retorna o valor lógico falso.

- **Raciocinar com uma pequena quantidade de informação** – Uma das vantagens do CBR é poder ser aplicado sobre um conjunto inicial de dados de dimensão reduzida. As características adaptativas deste paradigma permitem que ao longo do tempo seja possível incrementar o conjunto de informação, ou seja a base de conhecimento, através da constante introdução de novos casos.

- **Aprender com o tempo** – O CBR molda-se a situações diferentes ao longo do tempo e "aprende" com o decorrer do processo de raciocínio. Outras técnicas de raciocínio necessitam de fases iniciais de treino e de teste para que as fases subsequentes do raciocínio sejam realizadas da forma mais assertiva possível. O CBR associa as soluções aplicadas em problemas passados e raciocina "no momento" acerca de qual deve ser a solução para o novo problema.

- **Prever a probabilidade de sucesso de uma solução** – O CBR também poderá prever se uma solução terá ou não sucesso quando aplicada ou adaptada para um novo problema. Se as soluções forem organizadas consoante o grau de sucesso que tiveram, o CBR poderá analisar o contexto do novo problema e perceber o que poderá acontecer com a adaptação de uma solução.

- **Evitar a ocorrência dos erros do passado** – Para além de prever o sucesso de uma solução, o CBR pode também prever a ocorrência de erros. Visto que é possível armazenar as causas que originaram erros no passado, um sistema que utilize CBR pode utilizar esta informação para prever a ocorrência desses mesmos erros no futuro.

- **Reutilizar etapas de resolução de problemas** – Como visto anteriormente (Secção 3.1), o CBR permite adaptar soluções aplicadas em situações anteriores. Isto permite que as etapas dessas soluções também sejam reutilizadas, o que evita a definição de soluções de "raiz".

- **Fornecer explicações para os problemas** – Os sistemas que implementem o CBR podem apresentar aos utilizadores a validade da solução que aplicam para um determinado problema. Para tal, basta indicar um caso passado e similar ao caso atual, cuja solução foi aplicada com sucesso, e a forma como se adaptou a solução para o novo caso.

- **Adaptar o raciocínio em vários contextos** – O CBR pode ser aplicado em domínios bastante distintos e, como visto anteriormente, com propósitos também diferentes. Isto deve-se ao facto da representação, retenção e adaptação de casos poderem ser aplicadas num "número aparentemente ilimitado de maneiras".

Assim, tal como referido no início desta secção, o CBR apresenta mais vantagens do que desvantagens. De entre as desvantagens deste paradigma de raciocínio, destacam-se dois pontos. O primeiro refere-se à qualidade das soluções obtidas na fase inicial do funcionamento do CBR. Se a base de conhecimento tiver uma dimensão reduzida, a solução obtida ou adaptada para um novo problema poderá não ser a ideal. Quanto maior e variado for o conjunto de casos, mais sólido se tornará o CBR. O outro ponto é relativo à definição de critérios para representar, filtrar e adaptar casos. Por vezes não é fácil decidir como se devem comparar os casos e como estes devem ser mantidos na base de conhecimento. Podem não se considerar critérios relevantes ou pode-se

considerar critérios em demasia para realizar estas operações. Encontrar um ponto de equilíbrio é complicado e como tal pode-se retirar ao CBR o seu carácter “simplista” e “adaptável”.

4

INTEGRAÇÃO DE CBR NUM SISTEMA DE AVALIAÇÃO

4.1 DOMÍNIO DE APLICAÇÃO

Um dos principais objetivos desta dissertação foi a integração de mecanismos de raciocínio num sistema de avaliação. O sistema de avaliação no qual foram inseridos estes novos mecanismos é o tutor inteligente *Leonardo* (Belo et al., 2019). Este tutor é um auxiliar educativo que possibilita o estudo em várias áreas do conhecimento e que apresenta os elementos de estudo (questões de escolha múltipla) aos utilizadores em sessões de *Quizz*. Para ser possível compreender melhor este sistema, apresenta-se na Figura 8 os seus principais componentes. Como se pode verificar nessa figura, a base de conhecimento do sistema é um elemento nuclear do tutor *Leonardo*, visto que é o principal repositório de informação do sistema.

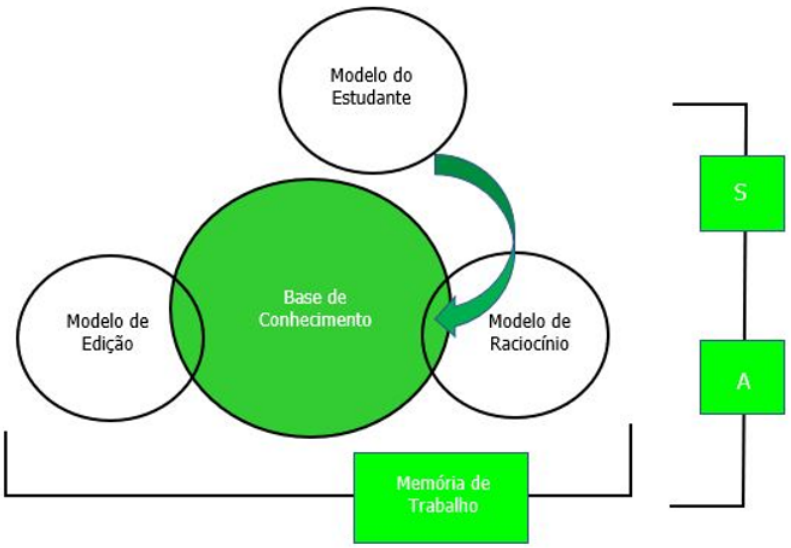

Figura 8: Modelo funcional do sistema *Leonardo* - adaptada de Belo et al. (2019)

O Modelo de Raciocínio avalia as respostas dadas nas questões do *Quizz*. Este módulo é também responsável por lançar as questões do *Quizz* de forma adaptativa, ou seja, tendo em conta o perfil dos estudantes e o estado atual da avaliação. Num processo de avaliação, este módulo comunica diretamente com o Modelo

do Estudante, que é o responsável pela criação, atualização e fornecimento de informação acerca do perfil dos estudantes.

O Modelo de Edição representa a introdução de nova informação por parte de peritos na base de conhecimento do sistema. Esta informação pode representar novas questões, novos domínios de conhecimento, entre outras coisas mais.

Por último, mas não menos importante, é possível observar uma memória de trabalho e dois componentes para interação com os utilizadores, os sensores (S) e os atuadores (A). A memória de trabalho representa a informação utilizada pelo sistema ao longo do processo de raciocínio e que permite estabelecer um fio condutor entre as várias fases do mesmo.

4.2 O MODELO DE CASOS ADOTADO

Um aspeto muito importante em qualquer sistema de CBR é a definição da estrutura dos casos que irão compor a sua base de dados (*case base*). Visto que o principal objetivo do sistema de raciocínio apresentado nesta dissertação é a geração de uma nova questão, decidimos que os casos são as questões mantidas na base de dados *MongoDB* do *Leonardo*. As questões assumem um papel muito importante no funcionamento deste sistema de avaliação. Para além da sua estrutura ser baseada na definição da estrutura do perfil dos alunos, as questões estão no centro da sua avaliação. O sistema de avaliação apresenta os elementos de estudo (questões) aos alunos de forma adaptativa, de acordo com as respostas dadas em cada questão. De seguida apresenta-se os vários atributos que constituem a estrutura de uma questão, bem como a descrição para cada um deles:

- ***id* –** Identificação do número da questão num domínio.
- ***language* –** Idioma no qual a questão está escrita.
- ***domain* –** Domínio de conhecimento da questão, subdividido em 3 campos:
  - ***study_cycle* –** Descrição do ciclo de estudos em que está inserido o domínio de conhecimento.
  - ***scholarity* –** Descrição da escolaridade do domínio de conhecimento dentro do ciclo de estudos.
  - ***description* –** Descrição concreta do domínio.
- ***subdomain* –** Designação do subdomínio da questão, dentro do domínio da mesma.
- ***difficulty_level* –** Nível de dificuldade da questão. O nível está contido num *ranking* de 1 a 5, inclusive.
- ***answering_time* –** Tempo máximo disponibilizado para responder à questão.
- ***header* –** Cabeçalho da questão (enunciado).
- ***body* –** Lista de opções de resposta da questão. Cada opção de resposta está subdividida em 3 campos:
  - ***answer* –** Corpo da resposta (texto descritivo da resposta).

  - ***correction –*** Grau de correção da resposta. O valor 0 significa uma resposta completamente errada e o valor 1 significa uma resposta totalmente certa.
  - ***mandatory –*** Valor representativo da obrigatoriedade ou falta da mesma desta opção de resposta aparecer como uma das opções apresentadas.

- ***solution –*** Solução da resposta.
- ***source –*** Origens de informação da resposta. Este elemento pode ser relativo à fonte de onde uma questão foi recolhida ou àquelas de onde é possível obter informações para auxílio ao estudo.
- ***notes –*** Apontamentos relativos à questão.
- ***inserted_by –*** Nome do utilizador que inseriu a questão na base de dados.
- ***inserted_at –*** Data de inserção da questão na base de dados.
- ***validated_by*** – Nome do utilizador que validou a questão na base de dados.
- ***validated_at –*** Data de validação da questão.
- ***status –*** Estado em que se encontra a questão: 'V' ou 'E'. O primeiro representa o estado "Validada e pronta para ser lançada ao aluno". O segundo representa o estado "Editada e a aguardar validação".
- ***repetitions –*** Número de vezes que a questão pode ser apresentada a um aluno.
- ***precedence –*** Lista dos identificadores das questões que devem ser lançadas anteriormente à questão em causa.
- ***display_mode –*** Modo como as opções de resposta devem ser apresentadas ao aluno: 'G' ou 'I'. O primeiro modo indica que as opções de resposta devem ser apresentadas de forma simultânea, ao contrário do segundo modo que indica que estas devem aparecer ao aluno de forma individual, uma a uma.

Na Figura 9 é possível observar um exemplo de uma questão para a coleção *questions* da base de dados do *Leonardo*. A questão é relativa ao domínio 'Sistemas de Bases de Dados', escolaridade 'Engenharia Informática', ciclo de estudos 'Ensino Superior' e subdomínio 'SQL'. Esta questão apresenta um nível de dificuldade igual a '3'. Para além disto, esta questão é composta por seis opções de resposta, das quais apenas uma é correta. Importante mencionar que a inserção e a validação desta questão foram realizadas pelo mesmo utilizador, 'Belo, O', num espaço temporal de dez dias. O atributo 'status' com o valor 'E' indica que a questão, caso estivesse realmente mantida na base de dados do *Leonardo*, não estaria apta para ser apresentada na *interface* do sistema.

```
{
    "_id": ObjectId("60c9ccc39b3e2c905927ff44"),
    "id": "PTEINSBDSQL021",
    "language": "pt",
    "study_cycle": "Ensino Superior",
    "scholarity": "Engenharia Informática",
    "domain": "Sistemas de Bases de Dados",
    "subdomain": "SQL",
    "subsubdomain": "",
    "difficulty_level": "3",
    "display_mode": "G",
    "answering_time": "60",
    "type": "",
    "precedence": [],
    "repetitions": "0",
    "header": "A query 'INSERT INTO Alunos VALUES ('1','João Manuel Castro', 'MIEI'),
              ('2','Silvia do Canto', 'LCC');' permite fazer:",
    "body":[
        {
            "answer": "Nada, uma vez que a query não está bem definida.",
            "correction": "0",
            "mandatory":"0",
            "eliminative": "0",
            "points": "0"
        },
        {
            "answer": "A inserção de um novo registo na tabela Alunos.",
            "correction": "0",
            "mandatory": "0",
            "eliminative": "0",
            "points":"0"
        },
        {
            "answer": "A inserção de um conjunto de registos na tabela Alunos, desde que
                       a ordem dos valores apresentados siga o esquema da tabela.",
            "correction": "1",
            "mandatory": "1",
            "eliminative": "0",
            "points": "1"
        },
        {
            "answer": "A inserção não pode ser realizada, uma vez que a operação envolve mais
                       do que um registo.",
            "correction": "0",
            "mandatory": "0",
            "eliminative": "0",
            "points": "0"
        },
        {
            "answer": "Dá erro, uma vez que a instrução apresentada tem erros de sintaxa.",
            "correction": "0",
            "mandatory": "0",
            "eliminative": "0",
            "points":"0"
        },
        {
            "answer": "Nenhuma das anteriores.",
            "correction": "0",
            "mandatory": "1",
            "eliminative": "0",
            "points": "0"
        }
    ],
    "solution": "",
    "images": "",
    "videos": "",
    "source": "Belo, O., Cadernos de Problemas - Sistemas de Bases de Dados, Departamento de
               Informática, Escola de Engenharia, Universidade do Minho, 2021",
    "notes": "Para utilização exclusiva da plataforma educacional Leonardo",
    "status": "E",
    "inserted_by": "Belo, O",
    "inserted_at": "2021-01-31",
    "validated_by": "",
    "validated_at": ""
}
```


Figura 9: Exemplo de uma questão do sistema *Leonardo*

4.3 IMPLEMENTAÇÃO DO SISTEMA

4.3.1 *Caracterização Base*

O sistema que iremos de seguida apresentar é um sistema de CBR, que prolonga as sessões de *Quizz* do sistema através da geração de questões adicionais de escolha múltipla. Como mencionado anteriormente, uma sessão do *Quizz* termina quando o módulo de avaliação do *Leonardo*, *Evaluator*, conclui que, pelo perfil que um determinado aluno apresenta, não é possível apresentar mais questões na *interface* do sistema. Por forma a manter uma coerência com os restantes módulos do *Leonardo*, este módulo foi desenvolvido na linguagem *Python*.

A introdução do novo sistema de raciocínio provocou algumas alterações na arquitetura do sistema, como se pode observar na Figura 10. O modelo funcional geral para o sistema depende do modelo de avaliação do tutor, que é representado pelo Modelo de Raciocínio da figura. Esta dependência deve-se ao facto da geração de uma questão ocorrer apenas quando o módulo de avaliação indica o fim de uma sessão de *Quizz*.

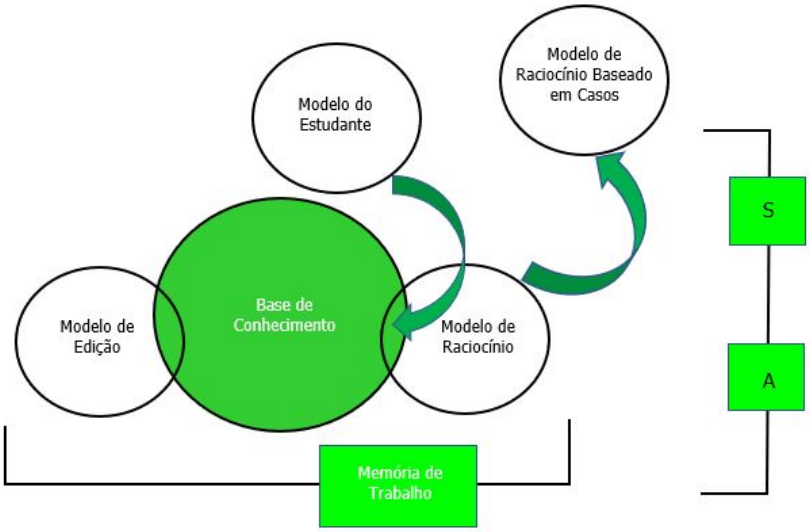

Figura 10: Arquitetura do sistema *Leonardo* incluindo o modelo de CBR

Este novo módulo tem a estrutura típica de um sistema de raciocínio, como aquela que se apresentou anteriormente, na Figura 4. O componente de 'sensorização' aguarda pelo sinal do módulo de avaliação do *Leonardo* para iniciar o processo de raciocínio. A 'reação' do sistema é representada pela geração de uma nova questão. Para este módulo do sistema decidiu-se que a base de conhecimento seria a mesma do tutor inteligente, por forma a reduzir a redundância de informação no sistema.

Uma das grandes inspirações para o desenvolvimento deste sistema de raciocínio foi o trabalho desenvolvido por McSherry (2010). Neste trabalho, o autor apresentou um mecanismo, designado por *AutoMCQ*, para a geração automática de questões de escolha múltipla, que segue uma abordagem baseada em casos. A construção do enunciado das questões baseia-se na instanciação de *templates*, ou seja, a informação dos casos da base

de dados (*case base*) são utilizadas para completar os enunciados de questões com uma estrutura previamente definida (*templates*). Através dos casos utilizados nas instanciações obtêm-se as opções de resposta corretas. As restantes opções de resposta do corpo da nova questão são designadas por *distractors* que, apesar de opções incorretas, apresentam um elevado grau de similaridade com as corretas.

Os pontos deste trabalho que mais influenciaram a modelação do novo módulo do *Leonardo* foram, nomeadamente:

- a seleção de casos interessantes para sustentar a formação da nova questão;
- a definição de medidas de interesse para a seleção destes casos;
- a seleção dos *distractors* na formação das opções de resposta;
- a definição de medidas para calcular o grau de similaridade entre os *distractors* e as opções de resposta corretas da nova questão.

4.3.2 *Estruturas de Dados Relevantes*

A definição do modelo dos casos foi precedida pela definição de outras estruturas importantes. Como indicado na Secção 2.1, um sistema de raciocínio possui tipicamente uma *working memory*, que é um componente que mantém fluidez entre as várias etapas do processo de raciocínio. De seguida, apresentam-se os constituintes da *working memory* deste sistema baseado em casos, bem como a descrição para cada um deles:

- ***username –*** Nome do utilizador.
- ***domain -*** Descrição concreta do domínio de conhecimento.
- ***study_cycle -*** Descrição do ciclo de estudos em que está inserido o domínio de conhecimento.
- ***scholarity -*** Descrição da escolaridade do domínio de conhecimento dentro do ciclo de estudos.
- ***subdomain -*** Descrição do subdomínio dentro do domínio de conhecimento.
- ***difficulty_level -*** Nível de dificuldade da última questão respondida pelo aluno.
- ***retrieved_questions -*** Questões respondidas por um aluno num certo nível de dificuldade, domínio e subdomínio de conhecimento.
- ***most_interesting_question –*** Questão, do conjunto das *retrieved_questions*, com o maior grau de interesse para ser apresentada num dado momento de uma sessão de *Quizz*.
- ***correct_answers -*** Opções de resposta corretas da *most_interesting_question*.
- ***nlp –*** Modelo de processamento de linguagem natural, disponibilizado pela ferramenta *Spacy* (Explosion, 2021).

- ***tokens_to_be_replaced_by_synonyms –*** Lista de substantivos e adjetivos presentes no *header* da *most_interesting_question*. Cada uma destas palavras é caracterizada por seis campos:
  - ***pos -*** Categorização gramatical da palavra (substantivo, adjetivo, etc.).
  - ***position_on_list -*** Posição da palavra no conjunto de *tokens* do cabeçalho da questão mais interessante (*most_interesting_question*). Um *token* pode ser entendido como um constituinte de uma frase.
  - ***description -*** Descrição concreta da palavra.
  - ***gender -*** Gênero gramatical da palavra: feminino (Fem) ou masculino (Masc).
  - ***number -*** Categorização quantitativa da palavra: singular (Sing) ou plural (Plur).
  - ***lemmas –*** Lista de palavras que possuem um sentido semelhante à palavra em questão.
- ***new_question -*** Questão formada pelo sistema de CBR.

Estando apresentada a estrutura da *working memory*, é importante explicar a inclusão que nela fizemos de certos atributos. A inclusão da informação referente ao utilizador, ao domínio de conhecimento, à escolaridade, ao ciclo de estudos, ao subdomínio e ao nível de dificuldade foi algo que foi necessário fazer, visto que esta informação é transmitida pelo módulo de avaliação para este sistema, assim que uma sessão de *Quizz* termina. Os atributos *retrieved_questions* e *most_interesting_question* representam algumas das informações das etapas do CBR (*retrieve*, *reuse*, *revise* e *retain*). As *retrieved_questions* representam os casos recolhidos na fase *retrieve*, a partir dos quais se seleciona a questão/caso mais interessante, *most_interesting_question*. Esta questão é selecionada na fase *retrieve* e irá sustentar a formação da nova questão de escolha múltipla.

Como se explicará mais adiante, a ferramenta *Spacy* assume um papel muito importante no funcionamento deste sistema de raciocínio, visto que é utilizada na comparação de expressões e na obtenção de sinónimos. As funcionalidades desta ferramenta são aplicadas através de um modelo de processamento de linguagem natural. Para além do carregamento deste modelo para memória ser demorado, a sua utilização dá-se em vários momentos do processo de raciocínio. Ou seja, este modelo deve ser apenas carregado uma vez para memória e estar constantemente disponível para utilização. Todas estas razões permitem justificar a inclusão do atributo *nlp* na estrutura da *working memory*.

A *working memory* é um objeto JSON (*JavaScript Object Notation*) não armazenado na base de dados do *Leonardo*. Na Figura 11 apresenta-se um exemplo de uma *working memory* para este sistema de raciocínio. Neste exemplo é possível verificar que já foi criada uma questão de escolha múltipla, cujo *id* é igual a 'PTEINS-BDSQL023', para completar a sessão de avaliação do utilizador 'leo_teste'. Esta sessão ocorreu no âmbito do domínio 'Sistemas de Bases de Dados', escolaridade 'Engenharia Informática', ciclo de estudos 'Ensino Superior' e subdomínio 'SQL'. Para além disto, o nível de dificuldade, no momento em que a sessão terminou, encontrava-se no valor '1'. Na Figura 11, podemos observar que os atributos *domain*, *scholarity*, *study_cycle*, *subdomain* e *difficulty_level* das questões filtradas, da questão mais interessante e da nova questão são os mencionados anteriormente. Isto permite confirmar que o raciocínio aplicado por este sistema gera uma questão de escolha múltipla tendo em consideração o estado de uma sessão de *Quizz* no momento em que esta terminou.

```
{
        'username': 'leo_teste',
        'domain': 'Sistemas de Bases de Dados',
        'study_cycle': 'Ensino Superior',
        'scholarity': 'Engenharia Informática',
        'subdomain': 'SQL',
        'difficulty_level': '1',
        'retrieved_questions': [
                {
                        '_id': ObjectId('60f1acd56ba5814bf266938c'),
                        'id': 'PTEINSBDSQL001',
                        'study_cycle': 'Ensino Superior',
                        'scholarity': 'Engenharia Informática',
                        'domain': 'Sistemas de Bases de Dados',
                        'subdomain': 'SQL',
                        'subsubdomain': '',
                        'difficulty_level': '1',
                        ...
                        'header': 'A SQL é uma linguagem utilizada em programação concebida especificamente para a descrição,
                        manipulação e controlo de dados num sistema de bases de dados relacional. Esta afirmação é:',
                        'body': [
                                {
                                        'answer': 'Verdadeira.',
                                        'correction': '1',
                                        'mandatory': '1',
                                        'eliminative': '0', '
                                        points': '1'
                                }...
                        ],
                        ...
                }...
        ],
        'most_interesting_question': {
                '_id': ObjectId('60f1acd56ba5814bf266938e'),
                'id': 'PTEINSBDSQL004',
                'study_cycle': 'Ensino Superior',
                'scholarity': 'Engenharia Informática',
                'domain': 'Sistemas de Bases de Dados',
                'subdomain': 'SQL',
                'subsubdomain': '',
                'difficulty_level': '1',
                ...
                'header': 'Genericamente , em SQL , que tipos de operações podemos realizar sobre uma base de dados :',
                'body': [
                        {
                                'answer': 'Descrição, manipulação e controlo de dados.',
                                'correction': '1',
                                'mandatory': '1',
                                'eliminative': '0',
                                'points': '1'
                        }...
                ] ...
        },
        'correct_answers': ['Descrição, manipulação e controlo de dados.'],
        'nlp': <spacy.lang.pt.Portuguese object at 0x7fb5cc38e250>,
        'new_question': {
                '_id': ObjectId('6103fadb686e4f185a94f9bf'),
                'id': 'PTEINSBDSQL023',
                'study_cycle': 'Ensino Superior',
                'scholarity': 'Engenharia Informática',
                'domain': 'Sistemas de Bases de Dados',
                'subdomain': 'SQL',
                'subsubdomain': '',
                'difficulty_level': '1',
                'header': 'Genericamente , em SQL , que tipos de operações podemos realizar sobre uma base de dados :',
                'body': [
                        {
                                'answer': 'Descrição, manipulação e controlo de dados.',
                                'correction': '1',
                                 'mandatory': '1',
                                 'eliminative': '0',
                                 'points': '0'
                        }...
                ]...
        }
}
```

Figura 11: Exemplo de um elemento da *working memory* para o sistema de CBR

4.3.3 *Processo de Raciocínio*

O funcionamento do sistema de CBR assenta em quatro fases principais, em tudo semelhantes àquelas que caracterizam um processo de CBR: recolha (*retrieve*), reutilização (*reuse*), revisão (*revise*) e retenção (*retain*).

O diagrama de atividades presente na Figura 12 exemplifica os pontos mais relevantes do funcionamento do sistema, que culmina no armazenamento de uma nova questão de escolha múltipla na base de dados do tutor inteligente. Como se pode ver nessa figura, através da ação '***Indica que não existem mais questões para lançar***', o sistema de CBR é "ativado" assim que o módulo de avaliação do *Leonardo* indica que não é possível apresentar mais questões na *interface* do sistema.

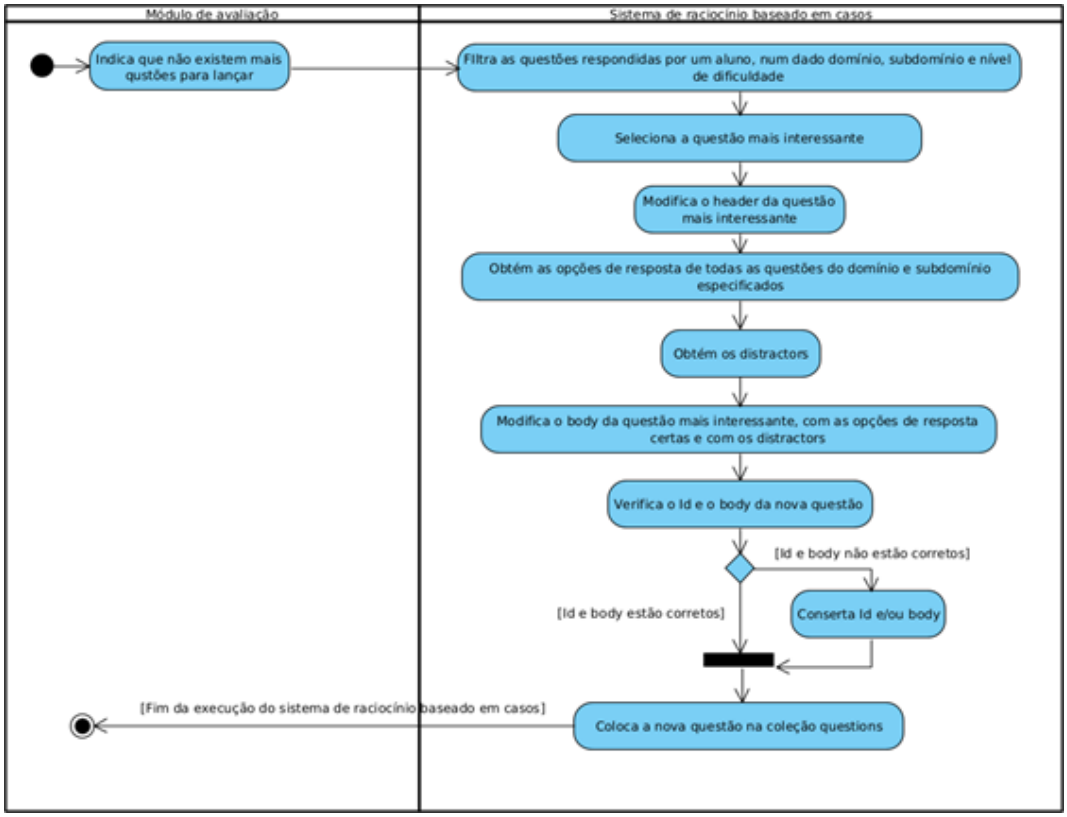

Figura 12: Diagrama de atividades do fluxo do sistema de CBR

A Fase de Retrieve

A fase retrieve representa o início do raciocínio do sistema baseado em casos e é representada por dois grandes processos: uma recolha inicial de casos (Figura 13) e a seleção, de entre estes, do mais "apto" para sustentar a formação de uma nova questão de escolha múltipla. De realçar que, como já referido anteriormente, as questões mantidas na base de dados do *Leonardo* são considerados os casos deste sistema de raciocínio.

Assim sendo, primeiramente recolhem-se as questões respondidas por um utilizador num determinado nível de dificuldade, domínio e subdomínio de conhecimento, que estão presentes na coleção *questions* da base dados do *Leonardo*. Esta recolha teve em consideração estes três pontos, pois decidiu-se que a nova questão deverá ser do mesmo domínio e subdomínio da última questão respondida pelo utilizador, para além de que deverá ter o mesmo nível de dificuldade.

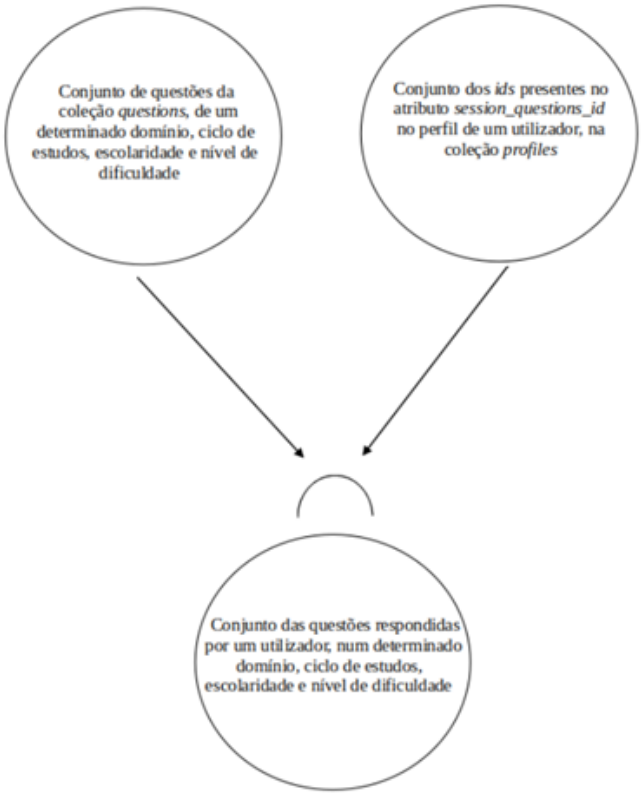

Figura 13: Esquematização da recolha de casos da fase *retrieve* do sistema de CBR

As questões obtidas nesta fase representam o conjunto *retrieved_questions* da *working memory*. De realçar que, como se pode verificar na Figura 13, nesta recolha realizam-se duas sub-filtragens, que se interligam através do atributo *session_questions_id* presente nos perfis dos utilizadores. Sem este atributo não seria possível relacionar tão facilmente as coleções *profiles* e *questions* para assim se obterem as questões respondidas pelos utilizadores.

O outro processo que se realiza na fase *retrieve* é a seleção da questão mais interessante. Esta questão, proveniente do conjunto obtido na filtragem anterior, é a que melhor sustenta a formação da nova questão de escolha múltipla num dado momento de uma sessão de *Quizz*. Para facilitar a seleção desta questão, que se representa pela ação '***Seleciona a questão mais interessante***' do diagrama de atividades da Figura 12, definiu-se a fórmula que se apresenta na Figura 14.

```
interesse = ( 0.6 * número de vezes respondida ) +
                ( 0.3 * número de opções de resposta ) +
                    ( 0.1 * número de opções de resposta corretas)
```

Figura 14: Fórmula para o cálculo de interesse de uma questão

Nesta fórmula são considerados três fatores principais: o número de vezes que os utilizadores responderam à questão, o número de opções de resposta da questão e o número de opções de resposta corretas da mesma. Estes fatores são distintos entre si e têm complexidades também distintas. Como tal, foi atribuído um peso diferente a cada um destes, pela ordem da sua relevância. De realçar que se deu maior importância ao número de vezes que uma questão foi respondida pela totalidade dos utilizadores, para permitir ao sistema *Leonardo* relacionar e compreender o estudo dos vários utilizadores.

A Fase de Reuse

A fase *reuse* é relativa à formação da nova questão, que consiste na modificação de alguns dos atributos da questão mais interessante (representada pelo atributo *most_interesting_question* da *working memory*), dos quais se destacam o *header* e o *body*. A alteração incidiu principalmente nestes atributos pois estes são, na prática, os mais visíveis aos olhos dos utilizadores. Ou seja, é através da leitura do enunciado e da seleção das opções de resposta (*body*) de uma questão que um utilizador comunica com o tutor inteligente *Leonardo*. A modificação destes atributos permite apresentar novos elementos de estudo aos utilizadores na forma de uma nova questão.

A modificação do *header* representa a substituição de algumas das suas palavras – mais concretamente os substantivos e adjetivos – por sinónimos. Esta alteração é realizada com o auxílio da biblioteca *Spacy*, que é uma ferramenta de processamento de linguagem natural, disponível na linguagem de programação *Python*. Um dos elementos mais importantes neste processo é o modelo de linguagem natural disponibilizado por esta biblioteca, representado pelo atributo *nlp* da *working memory*. Através deste modelo, é possível obter várias características gramaticais de uma palavra, como por exemplo o seu género e a categoria sintática a que pertence (substantivo, adjetivo, entre outros). Para além disto, é também possível associar as palavras que possuem o mesmo significado, os *synsets*, e os vários sentidos associados a cada uma destas, os *lemmas*. Tendo em consideração estas noções acerca da ferramenta *Spacy*, a modificação dos substantivos e adjetivos do *header* da questão consistiu em quatro pontos principais:

- Procura por substantivos e adjetivos no *header* da questão.
- Utilização do modelo de linguagem natural para obter o género e pluralidade de cada palavra (singular e plural).
- Obtenção dos *lemmas* associados a cada palavra.
- Seleção de um *lemma* cujo género e pluralidade sejam iguais aos da palavra que se pretende substituir.

De realçar que apenas se substituíram os substantivos e adjetivos contidos no atributo *header*, pois a substituição deste tipo de palavras dá uma maior fiabilidade para o funcionamento do sistema. Por vezes, outro tipo de palavras, como por exemplo os verbos compostos, não podem ser literalmente substituídos por outros. A obtenção de sinónimos para este tipo de palavras poderia gerar demasiados erros durante a execução do sistema.

A alteração do *body* representa a modificação do conjunto de opções de resposta da questão mais interessante. Esta alteração engloba a obtenção dos *distractors*, representada pela ação '***Obtém os distractors***' do diagrama de atividades. Estes elementos são opções de resposta incorretas, mas muito similares às corretas. Assim, a escolha destes elementos baseia-se na seleção das opções de resposta incorretas, referentes às questões respondidas por um utilizador, que mais se assemelham às opções de resposta corretas da questão mais interessante.

A verificação da similaridade entre as opções de resposta é realizada com o auxílio da biblioteca *Spacy*. Mais uma vez, o modelo de linguagem natural disponibilizado nesta biblioteca assume extrema relevância no processo de raciocínio, visto que tem a capacidade de determinar o grau de similaridade entre expressões textuais. Assim, para cada opção de resposta incorreta, soma-se o grau de similaridade entre esta e cada uma das opções de resposta corretas e posteriormente faz-se uma média desta soma. O *body* é então formado pelas opções de resposta corretas e as opções de resposta incorretas com melhor média (neste caso selecionam-se as quatro opções de resposta incorretas com melhor média).

A Fase de Revise

Na fase *revise*, que se inicia na ação '***Verifica o id e o body da nova questão***', analisa-se os elementos da nova questão - o *id* e o *body* - que podem introduzir redundância de informação na base de dados do *Leonardo* ou incoerências na apresentação da questão na *interface* do sistema. A correção das anomalias é representada pela ação '***Conserta id e/ou body***'.

As duas ações mencionadas anteriormente evitam que o *id* da nova questão seja igual ao de alguma questão já persistida na base de dados do *Leonardo*. Importa agora realçar os constituintes do *id* de uma questão, bem como a forma como se atribui um valor a este atributo, seja na construção inicial da nova questão como na geração de outro *id* para a mesma.

O *id* de uma questão representa várias informações, como por exemplo a linguagem da questão ou o domínio associado a esta. Por exemplo o valor de 'PTEINSBDSQL023' para o *id* indica que a questão está escrita em português, pertence à escolaridade 'Engenharia Informática', domínio 'Sistemas de Bases de Dados' e subdomínio 'SQL'. O identificador '023' indica que existem 23 questões com estes parâmetros. Na formação do *id* da nova questão mantêm-se os identificadores da linguagem, escolaridade, domínio e subdomínio presentes no *id* da *most_interesting_question* e apenas se incrementa o identificador numérico numa unidade.

Na fase *revise* eliminam-se também as opções de resposta incorretas que tenham o mesmo enunciado das opções de resposta corretas e colocam-se a 0 (valor *default*) todos os valores dos atributos *correction*, *mandatory*, *eliminative* e *points* que sejam diferentes de 0 ou de 1. As opções de resposta com o enunciado igual a 'Nenhuma das anteriores' também são removidas do *body*, visto que não acrescentam grande informação/conhecimento para o estudo dos alunos.

A Fase de Retain

A fase *retain*, representada pela ação '***Coloca a nova questão na coleção questions***', caracteriza a persistência da nova questão na base de dados do *Leonardo*. A nova questão, que é representada pelo atributo

new_question da *working memory*, é colocada juntamente com as outras questões na coleção *questions*. Esta ação realiza-se através da execução de uma query *MongoDB*, como a que se representa na Figura 15. Mais uma vez se verifica a importância da *working memory*, que interliga o módulo de raciocínio com a base de dados do sistema.

```
mongo.db.questions.insert( self.working_memory['new_question'] )
```

Figura 15: *Query MongoDB* para inserção da nova questão na base de dados do Leonardo

Para registar o resultado da atividade do sistema de raciocínio criou-se uma nova coleção na base de dados do *Leonardo*, designada por *generated_questions*. Nesta coleção armazena-se o *id* e a data de criação das novas questões. Na Figura 16 é possível observar um exemplo de um documento para esta coleção, que representa a criação da questão com o id de 'PTEINSBDSQL023'. De realçar que para se inserir estes documentos nesta coleção utiliza-se queries *MongoDB*, semelhantes à que se apresenta na Figura 15. Os elementos diferenciadores nestas queries são a coleção (*generated_questions* em substituição de *questions*) e o conteúdo do método *insert*, que neste caso contém o *id* e a data de inserção da nova questão.

```
{
    "_id": ObjectId("61a8d8d08dae954d24dfb914"),
    "id": "PTEINSBDSQL023",
    "date": "2021-12-02 14:31:44.707555"
}
```

Figura 16: Exemplo de um documento JSON da coleção *generated_questions*

4.4 MÉTODO PARA MUDANÇA DE COMPORTAMENTO

Num sistema de avaliação, é importante que todas as etapas do seu funcionamento sejam devidamente testadas ou que possuam algum tipo de verificação. No tutor inteligente *Leonardo* as questões lançadas pelo módulo de avaliação, *Evaluator*, não são verificadas antes de serem transmitidas à *interface* do tutor. Para este efeito, desenvolveu-se um mecanismo sustentado por regras que funciona como um novo módulo de raciocínio do tutor inteligente. Tal como aconteceu com o sistema de CBR, este módulo foi desenvolvido na linguagem de programação *Python*. A arquitetura funcional do sistema *Leonardo* alterou-se mais uma vez, como se pode observar na Figura 17.

De realçar as ligações do novo modelo com os modelos do estudante e de raciocínio. Como se vai verificar mais adiante nesta dissertação, a verificação das questões engloba a comunicação com os módulos de *profiling* e de avaliação, com o intuito de se obter o perfil do utilizador e as questões que podem ser lançadas depois de se executar as regras. Para além disto, não existe relação entre este novo modelo e o modelo de CBR, visto que as questões geradas pelo sistema de CBR não são verificadas por este novo componente.

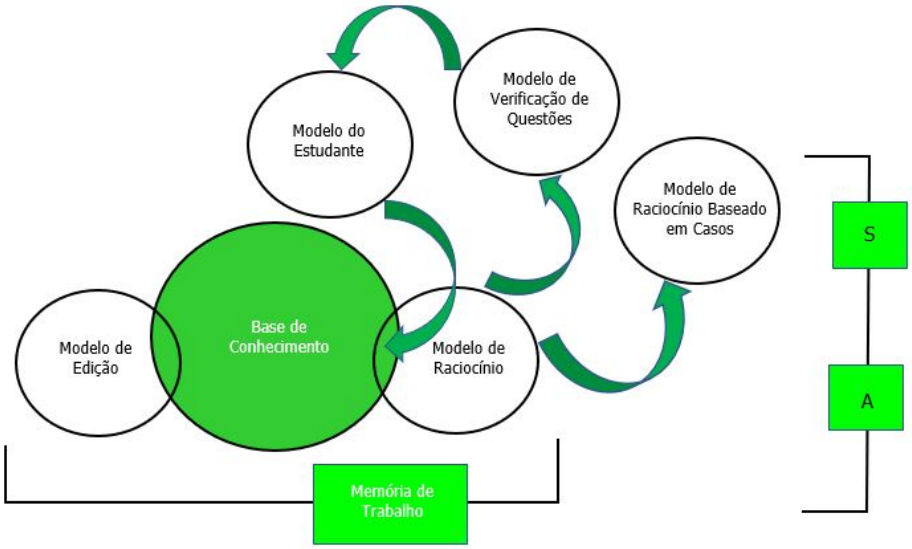

Figura 17: Arquitetura do sistema *Leonardo* incluindo o modelo de CBR e o modelo de verificação de questões

Na base da verificação das questões encontra-se a análise às propriedades de avaliação de um estudante num determinado domínio de conhecimento. Estas propriedades são calculadas e mantidas pelo módulo de *profiling* do *Leonardo*, o *Profiler*. De seguida são apresentadas as propriedades mencionadas anteriormente:

- ***user_level* -** Nível de conhecimento do aluno dentro do domínio de conhecimento.
- ***user_performance* –** *Performance* do aluno.
- ***user_skill* -** Percentagem de eficiência do aluno no domínio de conhecimento.
- ***total_questions* -** Número total de questões apresentadas ao aluno sobre o domínio de conhecimento.
- ***total_right_questions* -** Número de respostas certas dadas pelo aluno a questões do domínio de conhecimento.
- ***total_wrong_questions* -** Número de respostas erradas dadas pelo aluno a questões do domínio de conhecimento.
- ***hit_rate* -** Taxa de acerto no domínio de conhecimento.
- ***error_rate* -** Taxa de erro no domínio de conhecimento.
- ***usage_time* -** Tempo total disponível para responder ao total de questões do domínio de conhecimento.
- ***answering_time* -** Tempo total despendido pelo aluno a responder a questões do domínio de conhecimento.

- ***total_right_followed_questions -*** Número de questões certas em sequência no domínio de conhecimento.
- ***total_wrong_followed_questions -*** Número de questões erradas em sequência no domínio de conhecimento.

Através da análise aos valores destas propriedades, condiciona-se o comportamento do tutor inteligente, visto que é possível "rejeitar" uma questão que previamente tenha sido enviada pelo módulo de avaliação do mesmo.

4.4.1 *Modelação das Regras*

A análise de propriedades mencionada anteriormente é realizada através de regras, que expõem critérios de avaliação desenvolvidos por peritos, por exemplo professores, numa determinada área de conhecimento. Estes critérios são expressos através das pré-condições e pós-condições das regras. Desta forma, como se pode observar na Figura 18, consegue-se formar conjuntos de elementos condição-ação, cujos pontos centrais são as propriedades mencionadas anteriormente no início da Secção 4.4.

```
(id) se propriedade [=,>,<,<>,>=,<=] valor
       [e,ou] (propriedade operador valor)
       [e,ou] (propriedade operador valor) ...
então
       ação propriedade
       e ação propriedade
```

Figura 18: Estrutura base de uma regra para mudança de comportamento do *Leonardo*

Para além dos critérios mencionadas anteriormente, é necessário associar a uma regra as informações do domínio de conhecimento, escolaridade e ciclo de estudos. Estas informações caracterizam a área do conhecimento na qual os critérios serão aplicados. Considerando todos estes elementos, para representar uma regra na base de dados do sistema *Leonardo* devem ser considerados os seguintes atributos:

- ***id -*** Identificação do número da regra no domínio de conhecimento.
- ***study_cycle -*** Descrição do ciclo de estudos em que está inserido o domínio de conhecimento.
- ***scholarity -*** Descrição da escolaridade do domínio de conhecimento dentro do ciclo de estudos.
- ***domain -*** Descrição concreta do domínio de conhecimento.
- ***if_clauses -*** Lista das pré-condições que constituem a regra. Cada pré-condição é composta por cinco campos:
  - ***id -*** Identificação do número da pré-condição na regra.

  - ***property -*** Descrição da propriedade para mudança de comportamento.
  - ***operator -*** Descrição do operador que verifica o valor da propriedade para mudança de comportamento.
  - ***value -*** Valor da propriedade para mudança de comportamento.
  - ***condition -*** Descrição da conjunção que liga uma pré-condição à seguinte. Caso não haja mais pré-condições, esta descrição corresponde a uma *string* vazia.

- ***then_clauses -*** Lista das pós-condições que constituem a regra. Cada pós-condição é composta por três campos:
  - ***id -*** Identificação do número da pós-condição na regra.
  - ***action -*** Ação a realizar pela regra. Esta ação pode incrementar ou decrementar o valor da propriedade.
  - ***property -*** Descrição da propriedade para mudança de comportamento.

As regras são representadas por documentos JSON mantidos na coleção *rules* da base de dados (*MongoDB*) do *Leonardo*. Na Figura 19 é possível observar um exemplo de uma regra referente ao domínio 'Sistemas de Gestão Industrial', da escolaridade 'Engenharia Informática' e ciclo de estudos 'Ensino Superior'. Esta regra possui três pré-condições e uma pós-condição. Em termos textuais, esta regra expressa a seguinte expressão: "Se *total_questions* > 5 e *hit_rate* > 70 e *user_skill* < 2 então incrementa *user_skill*". Isto significa que, caso o número de questões respondidas por um aluno seja maior do que cinco, a taxa de acerto superior a setenta por cento e o *skill* menor do que dois, deve-se incrementar o *skill* do utilizador. Mais adiante neste documento se comprovará que este incremento não acontece diretamente no perfil do utilizador persistido na base de dados, mas no perfil que se encontra armazenado na *working memory*.

```
{
    "_id": ObjectId("60cfc1300abd77ee44a0457a"),
    "id": "4",
    "study_cycle": "Ensino Superior",
    "scholarity": "Engenharia Informática",
    "domain": "Sistemas de Gestão Industrial",
    "if_clauses": [
        { "id": "4.1", "property": "total_questions", "operator": ">", "value": "5", "condition": "e" },
        { "id": "4.2", "property": "hit_rate", "operator": ">", "value": "70", "condition": "e" },
        { "id": "4.3", "property": "user_skill", "operator": "<", "value": "2", "condition": ""}
    ],
    "then_clauses" : [
        { "id": "4.4", "action": "increment", "property": "user_skill" }
    ]
}
```

Figura 19: Exemplo de uma regra de verificação do comportamento do sistema *Leonardo*

4.4.2 *Funcionamento*

O funcionamento deste novo módulo é um pouco complexo, devido à quantidade de informação que trata e à sua ligação aos módulos do tutor inteligente, como é o caso dos módulos de *profiling* e de avaliação. Este módulo de raciocínio é "ativado" em certas partes de uma sessão de *Quizz*, mais concretamente nas situações em que o *Evaluator* lança uma questão, como se pode observar pela ação '***Lança uma questão***'. Caso um utilizador responda pela primeira vez a questões de um determinado domínio, o sistema não é executado. Ou seja, se não existem dados acerca da avaliação num domínio de conhecimento, o novo módulo não pode executar as regras, visto que estas dependem das propriedades de avaliação. Para além disto, e como já referido anteriormente, este módulo não verifica as questões geradas pelo módulo de CBR.

De seguida apresentam-se os aspetos mais relevantes do funcionamento deste módulo de raciocínio. A primeira parte é relativa a estruturas de dados importantes para manter o processo de raciocínio coerente e as restantes referem-se a partes específicas do funcionamento do módulo. Na Figura 20 apresenta-se um diagrama de atividades que permite ilustrar todo o processo de raciocínio realizado por este módulo, desde a filtragem das regras até ao momento em que se realiza a validação de uma questão.

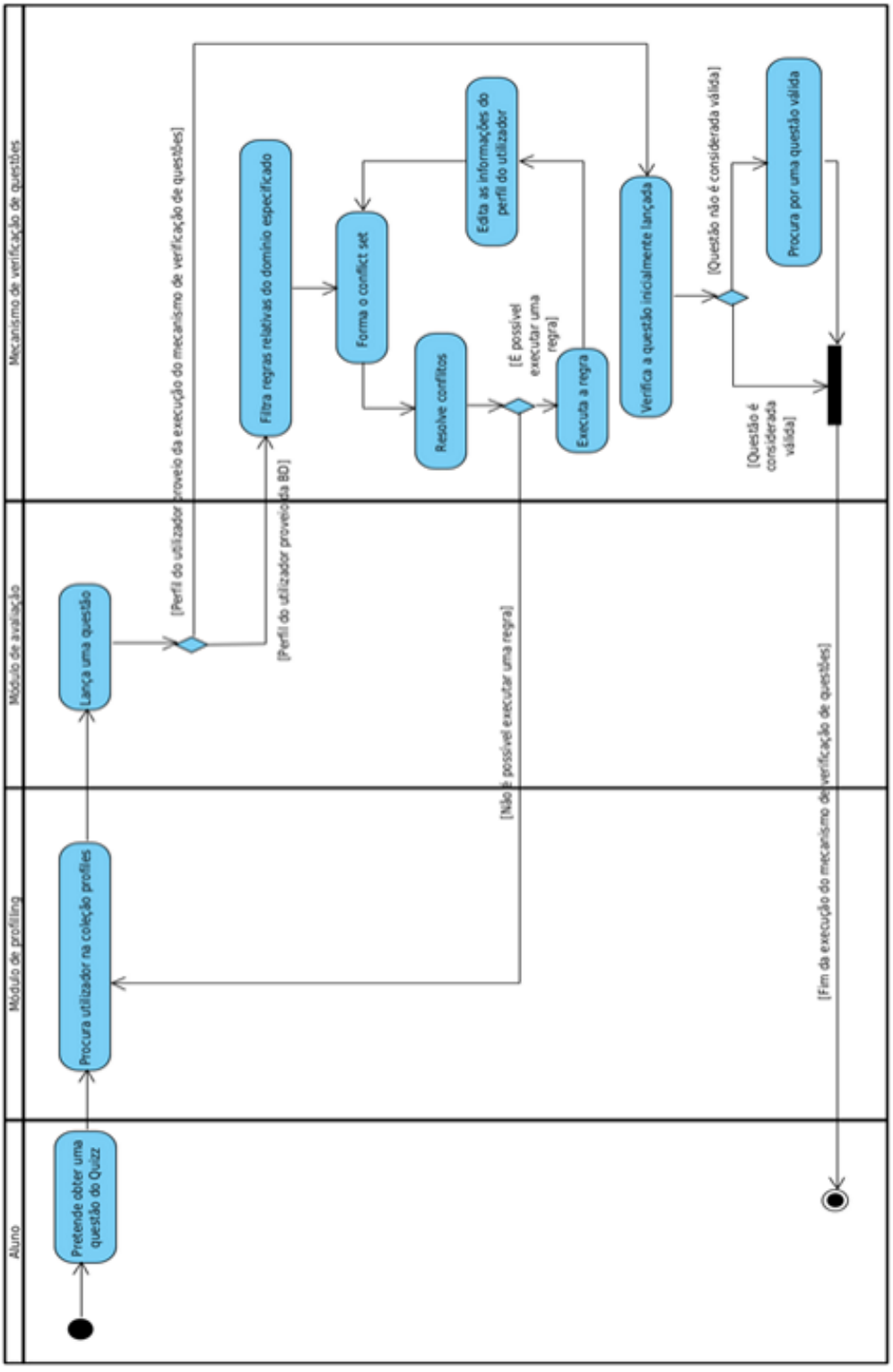

Figura 20: Diagrama de atividades do fluxo do sistema de raciocínio baseado em regras

Estruturas de Dados Relevantes

Tal como aconteceu na definição do sistema de CBR, mais concretamente na Secção 4.3.2, foi necessário definir uma *working memory* para manter o processo de raciocínio fluído. De seguida apresenta-se o conjunto dos vários atributos que constituem a estrutura deste componente:

- ***user_profile_info -*** Informação do perfil do utilizador contida na coleção *profiles* do sistema.
- ***current_user -*** Informação do utilizador contida na coleção *users* do sistema.
- ***rules_under_domain -*** Conjunto de regras pertencentes a um determinado domínio, escolaridade e ciclo de estudos.
- ***conflict_set -*** Conjunto de regras que podem ser executadas num dado momento do processo do raciocínio.
- ***rule_to_fire -*** Regra pertencente ao conjunto *conflict_set* e que foi escolhida para ser executada.
- ***fired_rules_id -*** Conjunto dos identificadores das regras disparadas durante o processo de raciocínio.
- ***domain -*** Descrição concreta do domínio.
- ***scholarity -*** Descrição da escolaridade do domínio de conhecimento dentro do ciclo de estudos.
- ***study_cycle -*** Descrição do ciclo de estudos em que está inserido o domínio de conhecimento.
- ***user_level -*** Nível do conhecimento do aluno dentro do domínio especificado.
- ***user_performance -*** *Performance* do aluno.
- ***user_skill -*** Percentagem de eficiência do aluno no domínio especificado.
- ***total_questions -*** Número total de questões apresentadas ao aluno sobre o domínio especificado.
- ***total_right_questions -*** Número de respostas certas dadas pelo aluno a questões do domínio especificado.
- ***total_wrong_questions -*** Número de respostas erradas dadas pelo aluno a questões do domínio especificado.
- ***hit_rate –*** Objeto que contém a taxa de acerto no domínio especificado antes e depois da execução das regras lógicas.
- ***error_rate –*** Objeto que contém a taxa de erro no domínio especificado antes e depois da execução das regras lógicas.
- ***usage_time -*** Tempo total disponível para responder ao total de questões do domínio especificado.

- ***answering_time –*** Tempo total despendido pelo aluno a responder a questões do domínio especificado.
- ***total_right_followed_questions -*** Número de questões certas em sequência no domínio especificado.
- ***total_wrong_followed_questions –*** Objeto que contém o número de questões erradas em sequência no domínio especificado antes e depois da execução das regras lógicas.

Estando apresentada a estrutura da *working memory*, é necessário justificar alguns dos atributos que nela estão incluídos. As propriedades mencionadas no início da Secção 4.4 são incluídas na *working memory* para facilitar a execução das regras. A execução de uma regra implica a atualização do valor destas propriedades e consequente atualização do perfil do utilizador em questão. A inclusão do perfil do utilizador na *working memory* advém de dois fatores. Primeiramente, torna-se desnecessário enviar pedidos à base de dados do *Leonardo* sempre que uma regra é executada. Para além disto, o principal objetivo deste mecanismo de raciocínio é verificar se, com as alterações de algumas propriedades dos utilizadores através da execução das regras, a questão inicialmente lançada pelo módulo de avaliação continua válida. A alteração do perfil do utilizador na base de dados é da exclusiva responsabilidade do módulo de *profiling*. A inclusão das informações do domínio de conhecimento facilita a filtragem das regras consoante o domínio da sessão de *Quizz*. Os atributos *rules_under_domain*, *conflict_set*, *rule_to_fire* e *fired_rules_id* facilitam o processo de seleção, execução e verificação das regras ao longo do processo de raciocínio.

De realçar que a *working memory* é um objeto JSON não mantido na base de dados do *Leonardo*. Na Figura 21 apresenta-se um exemplo de uma *working memory*. Neste exemplo verifica-se que existem quatro regras relativas ao domínio de 'Sistemas de Bases de Dados', escolaridade 'Engenharia Informática' e ciclo de estudos 'Ensino Superior'. Destas quatro regras apenas uma foi disparada, como se pode verificar pelo atributo *fired_rules_id*, que contém o *id* '2'. O processo de raciocínio encontra-se na fase final, visto que o atributo *rule_to_fire* se encontra vazio. Este mecanismo de raciocínio não repete a execução de regras. Como tal, apesar do *conflict set* conter a regra com o *id* '2', nenhuma regra é disparada, visto que esta já se encontra no conjunto das *fired_rules_id*. Para o raciocínio ficar concluído bastará aplicar os valores das propriedades (*user_skill*, *total_questions*, *total_right_questions*, etc.) ao perfil do utilizador contido na *working memory*. Isto permitirá obter o conjunto das questões que podem ser lançadas (tendo em conta os novos valores das propriedades) e verificar se a questão inicialmente lançada pelo módulo de avaliação se encontra neste conjunto.

```
{
        'user_profile_info': {
                'username': 'leo_teste',
                'user_level': 2,
                'total_questions': 6,
                'questions_right': 4,
                'questions_wrong': 2,
                'skill': 0.7887708027777779,
                'profile': [
                        {
                                'domain': {
                                        'study_cycle': 'Ensino Superior',
                                        'scholarity': 'Engenharia Informática',
                                        'description': 'Sistemas de Bases de Dados'
                                },
                                'user_level': 3,
                                'hitted': 4,
                                'total': 6,
                                'skill': 0.7887708027777779,
                                ...
                        }
                        ...
                ],
                ...
        },
        'current_user': {'username': 'leo_teste', 'name': 'leo_teste', ...},
        'rules_under_domain': [
                {'
                        'id': '2',
                        'study_cycle': 'Ensino Superior',
                        'scholarity': 'Engenharia Informática',
                        'domain': 'Sistemas de Bases de Dados',
                        'if_clauses': [...], 'then_clauses': [...]
                },
                {
                        'id': '3',
                        'study_cycle': 'Ensino Superior',
                        'scholarity': 'Engenharia Informática',
                        'domain': 'Sistemas de Bases de Dados',
                        'if_clauses': [...], 'then_clauses': [...]
                },
                {
                        'id': '5',
                        'study_cycle': 'Ensino Superior',
                        'scholarity': 'Engenharia Informática',
                        'domain': 'Sistemas de Bases de Dados',
                        'if_clauses': [...], 'then_clauses': [...]
                },
                {
                        'id': '1',
                        'study_cycle': 'Ensino Superior',
                        'scholarity': 'Engenharia Informática',
                        'domain': 'Sistemas de Bases de Dados',
                        'if_clauses': [...], 'then_clauses': [...]
                }
        ],
        'conflict_set': [
                {
                        'id': '2',
                        'study_cycle': 'Ensino Superior',
                        'scholarity': 'Engenharia Informática',
                        'domain': 'Sistemas de Bases de Dados',
                        'if_clauses': [...], 'then_clauses': [...]
                }
        ],
        'rule_to_fire': {},
        'fired_rules_id': ['2'],
        'domain': 'Sistemas de Bases de Dados',
        'scholarity': 'Engenharia Informática',
        'study_cycle': 'Ensino Superior',
        'user_level': 3,
        'user_performance': 1,
        'user_skill': 1,
        'total_questions': 6,
        'total_right_questions': 4,
        'total_wrong_questions': {'before_rule_execution': 2, 'after_rule_execution': -1},
        'hit_rate': {'before_rule_execution': 67, 'after_rule_execution': -1},
        'error_rate': {'before_rule_execution': 33, 'after_rule_execution': -1},
        'usage_time': 76,
        'answering_time': 76,
        'total_right_followed_questions': 4,
        'total_wrong_followed_questions': 0
}
```

Figura 21: Exemplo de um elemento da *working memory* para o mecanismo de raciocínio de verificação de questões

4.4.3 *Implementação Realizada*

Nesta secção apresentam-se os pontos mais relevantes da implementação do novo módulo baseado em regras, desde a filtragem de regras até à análise de uma questão. Apresentam-se também os tópicos que tipicamente caracterizam um mecanismo de raciocínio deste género, como por exemplo a formação do *conflict set* ou a resolução de conflitos.

Filtragem das Regras

Um dos primeiros passos do processo de raciocínio deste novo módulo é a filtragem das regras da coleção *rules* da base de dados do *Leonardo*, relativas ao domínio de conhecimento que o utilizador selecionou no início da sessão do *Quizz*. Este procedimento é representado pela ação '***Filtra regras do domínio especificado***' do diagrama da Figura 20. A filtragem das regras é realizada através da execução da query *MongoDB* da Figura 22. A *working memory* assume um papel importante, visto que efetua a ligação entre o módulo e a base de dados do *Leonardo*. Para além disto, as informações transmitidas do módulo de avaliação para este módulo baseado em regras - *domain*, *study cycle* e *scholarity* - são armazenadas na working memory.

```
db.rules.find({
    "domain": working_memory['domain'],
    "study_cycle": working_memory['study_cycle'],
    "scholarity": working_memory['scholarity']
})
```

Figura 22: *Query MongoDB* para obtenção das regras associadas a um domínio de conhecimento

Da execução desta query resulta um conjunto de regras semelhante ao conjunto *rules_under_domain* da *working_memory* da Figura 21. A filtragem destas regras sustenta as restantes etapas do processo de raciocínio, que se caracterizam por um ciclo entre a formação do *conflict set*, resolução de conflitos, seleção da "melhor" regra, execução desta e atualização das propriedades do perfil do utilizador na *working memory*.

Conflict Set e Resolução de Conflitos

Uma das fases mais relevantes no funcionamento de um mecanismo de raciocínio baseado em regras é a obtenção das regras que podem ser executadas dado o estado atual da *working memory*. Assim, forma-se um conjunto designado por *conflict set*. Para formar o *conflict set* seleciona-se, do conjunto obtido em 4.4.3, as regras cujas pré-condições tenham em conjunto o valor lógico de um e que, caso sejam executadas, mantenham os valores das propriedades dentro dos limites estabelecidos. Ou seja, se a pós-condição de uma regra indicar que o valor do *user level* deve aumentar uma unidade, por exemplo, a regra só deve ser adicionada ao *conflict set* se o valor do *user level* for menor do que cinco. Isto acontece porque o valor do *user level* tem como limites inferior e superior os valores zero e cinco, respetivamente.

Tendo o *conflict set* formado, podem existir conflitos ao decidir qual a regra deve ser disparada e, consequentemente, executada. Para se solucionarem estes problemas foram definidos alguns critérios - representativos da ação '***Resolve conflitos***' -, que se apresentam de seguida.

- A regra escolhida para disparar não pode ter sido disparada anteriormente no processo de raciocínio.
- A regra escolhida para disparar é aquela que possui mais pré-condições (ou seja, é a regra que envolve mais propriedades do perfil dos utilizadores), com exceção dos casos em que todas as regras têm o mesmo número de pré-condições. Nestes casos, é escolhida uma regra aleatoriamente do *conflict set*.

De realçar que ao selecionar-se uma regra do *conflict set*, a propriedade *fired_rules_id* da *working memory* é atualizada. Esta atualização envolve a colocação do *id* da regra que foi disparada neste conjunto. Desta forma, verifica-se mais facilmente as regras disparadas ao longo do processo de raciocínio.

Execução das Regras e Análise à Questão

As propriedades mencionadas anteriormente (Secção 4.4) são inicializadas de acordo com o perfil do utilizador, representado pela variável *user_profile_info* da *working memory*. A execução de uma regra reflete-se na atualização destas propriedades e, consequentemente, da variável *user_profile_info*. Ou seja, o perfil do utilizador, que é inicialmente carregado para a *working memory*, mantém-se sempre atualizado com o decorrer do processo de raciocínio. Este processo, que termina quando mais nenhuma regra possa ser executada, é representado pelo *loop* existente entre as ações '***Forma o conflict set***', '***Resolve conflitos***', '***Executa a regra***' e '***Edita as informações do perfil do utilizador***'.

De realçar a importância do nodo de decisão que se encontra ligado à ação '***Lança uma questão***'. O processo apresentado nas secções anteriores (Secção 4.4.3 e Secção 4.4.3) acontece sob a condição *Perfil do utilizador proveio da BD*. O outro caminho, representado pela condição *Perfil do utilizador proveio da execução do mecanismo de verificação de questões*, é percorrido quando todas as regras são disparadas e executadas. Ou seja, as informações do perfil do utilizador são atualizadas apenas na *working memory* (ação '***Edita as informações do perfil do utilizador***') e o novo módulo do *Leonardo* indica que pretende pedir uma questão. Contudo não pode comunicar diretamente com o módulo de avaliação, visto que o perfil do utilizador tem de ser transmitido ao módulo de avaliação sob a forma de um padrão. Esta ação é da responsabilidade do módulo de *profiling*, daí a ligação entre as colunas do mecanismo de verificação de questões e do módulo de profiling do diagrama de atividades da Figura 20.

A questão lançada pelo módulo avaliação, tendo em conta o "novo" perfil do utilizador, é comparada com a questão lançada antes de se ter iniciado o processo de raciocínio. Caso as questões sejam iguais, a questão inicialmente lançada é considerada válida e enviada para a *interface* do sistema. Caso contrário realiza-se a '***Procura por uma questão válida***'. Esta ação consiste em verificar se alguma das questões passíveis de ser lançadas tendo em conta o perfil original do utilizador estaria contida no conjunto das questões que podem ser lançadas tendo em conta as "modificações" ocorridas. Se alguma questão cumprir estes requisitos é apresentada na *interface* do sistema.

4.4.4 *O Editor de Regras*

Ao longo do tempo pode ser necessário alterar as regras da base de dados, adicionar ou remover outras. Para que estes processos sejam realizados de forma intuitiva e célere desenvolveu-se um editor de regras (Figura 23). A *interface* deste editor é *responsive*, permitindo assim aos utilizadores aceder ao editor nos seus computadores e noutros dispositivos, como por exemplo *smartphones* ou *tablets*. A *interface* também é reativa, o que permite aos utilizadores observar as alterações que realizaram sem ser necessário dar *reload* à página *web* do editor. Para implementar este tipo de *interface* utilizou-se a ferramenta *VueJs* (You, 2021). Este editor foi adicionado à plataforma do tutor inteligente, logo encontra-se constantemente disponível para utilização.

Os utilizadores podem visualizar as regras agrupadas por domínio de conhecimento, como se pode observar na Figura 23. Neste exemplo, verifica-se que existem duas regras associadas ao domínio 'Sistemas de Bases de Dados'. As regras são expressas na forma de expressões textuais e têm a si associadas as opções 'Editar' e 'Remover'.

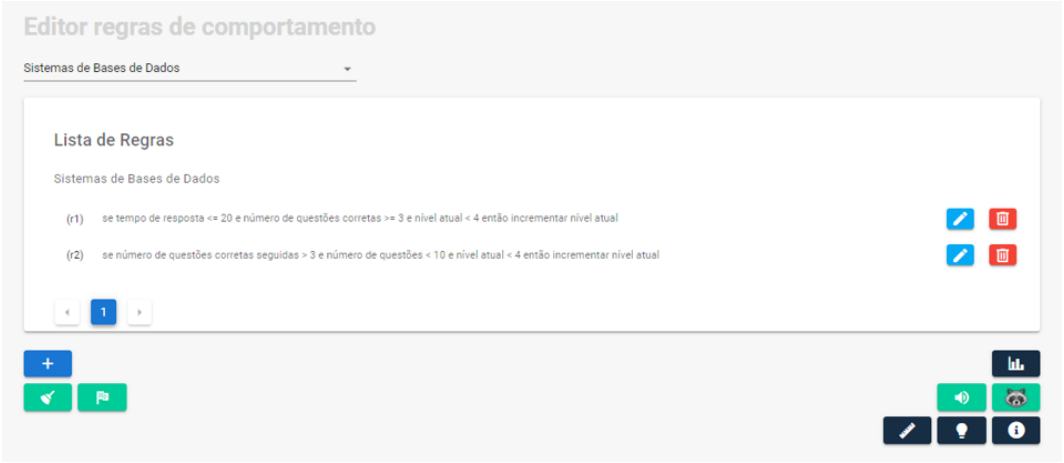

Figura 23: O ambiente do editor de regras

Na criação e edição de uma regra, os domínios de conhecimento, propriedades, operadores, valores e ações são editáveis e apresentados num componente *dropdown* (Figura 24). Desta forma, os utilizadores conseguem "manusear" muito mais facilmente os constituintes das regras. O exemplo que se apresenta na Figura 24 representa a edição de uma regra no contexto do domínio 'Sistemas de Bases de Dados'. Como se pode observar, o utilizador pretende alterar a pós-condição da regra, mais concretamente o tipo de ação - incrementar ou decrementar - a realizar sobre o valor da propriedade 'nível atual'.

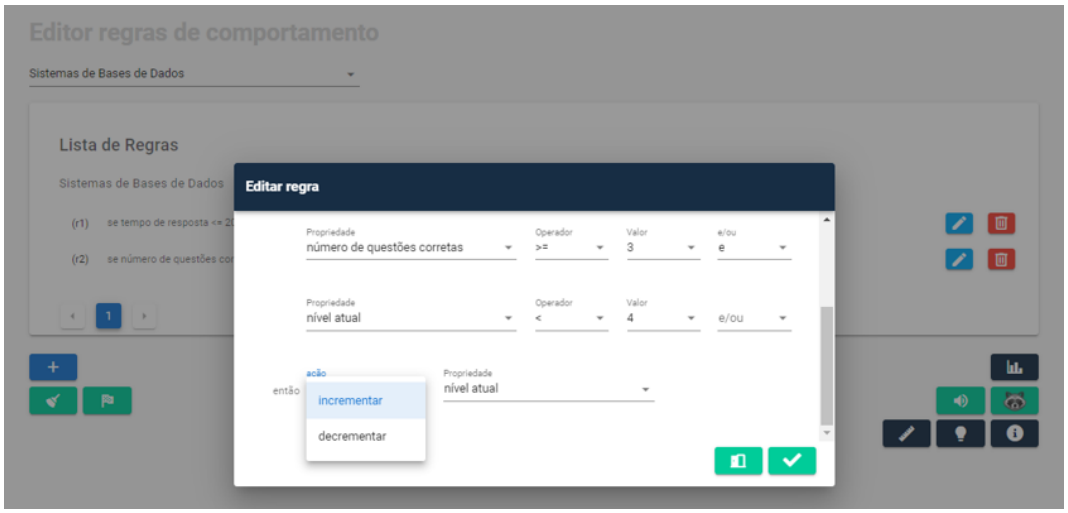

Figura 24: Edição de uma regra

Importante realçar, mais uma vez, que um dos objetivos deste editor de regras é facilitar o acesso ao conteúdo da base de dados do *Leonardo*. Mais concretamente, este editor possiblita o acesso aos utilizadores, de forma fácil, às regras - mantidas na coleção *rules* - que permitem ao tutor inteligente analisar e alterar o seu próprio comportamento.

4.5 A MELHORIA DO SISTEMA DE RACIOCÍNIO

O sistema de CBR que se desenvolveu no âmbito desta dissertação realiza a geração de questões de forma relativamente rápida. Para isto contribuiu a utilização de uma *working memory*, que armazena os elementos mais relevantes para o processo de raciocínio, evitando assim pedidos desnecessários à base de dados do *Leonardo*. Este sistema diversifica a geração das questões ao longo do tempo, através da edição dos textos dos cabeçalhos das questões. Para além disto, para formar o conjunto de opções de resposta (*body*), o sistema tem em consideração as várias questões que um aluno responde ao longo do seu processo de aprendizagem, o que reforça o caráter adaptável deste sistema de raciocínio.

De realçar que o tutor inteligente passa a entender melhor como pode relacionar o conhecimento dos vários estudantes. Isto deve-se ao facto de, na geração de uma questão, serem tidas em consideração as questões respondidas pelo conjunto dos utilizadores da base de dados do *Leonardo*.

A verificação das questões lançadas pelo módulo de avaliação do *Leonardo* também é realizada de forma concisa e rápida. Mais uma vez, a definição de uma *working memory* contribuiu para aligeirar o processo de raciocínio. Este método complementar torna-se praticamente invisível aos olhos do utilizador e uma barreira importante entre os módulos de raciocínio do *Leonardo* e a *interface* deste tutor inteligente.

5

CONCLUSÕES E TRABALHO FUTURO

5.1 CONCLUSÕES

Os sistemas de avaliação assumem um papel importante na avaliação dos estudantes. Uma oportunidade para melhorar o comportamento de um sistema de avaliação é sempre vantajoso, quer para a comunidade académica, quer para a comunidade científica. Tendo por base esta motivação realizou-se a inclusão de mecanismos de raciocínio adaptativo num sistema de avaliação, o *Leonardo*. O trabalho realizado e apresentado nesta dissertação permitiu integrar novas camadas de raciocínio neste sistema. Este trabalho foi sustentado por duas fases de análise. Primeiramente analisou-se a forma como os sistemas de raciocínio poderiam ser incluídos em sistemas de avaliação, o que representar nestes mesmos sistemas e, por fim, como comunicar com os sistemas de avaliação. Numa fase posterior analisou-se extensivamente o trabalho que já se encontrava implementado no *Leonardo*, em particular os módulos de *profiling* e de avaliação, o papel que estes módulos representam no sistema e a forma como comunicavam entre si.

O módulo de *profiling*, o *Profiler*, cria e atualiza os perfis dos utilizadores do *Leonardo* ao longo das várias sessões de avaliação. Ou seja, este módulo permite registar a assertividade dos utilizadores ao longo do seu processo de aprendizagem. O *Profiler* comunica diretamente com o módulo de avaliação do *Leonardo*, designado por *Evaluator*. Este módulo analisa as respostas dos utilizadores às questões de escolha múltipla e seleciona as questões que devem ser apresentadas na *interface* do sistema. De realçar que as questões são apresentadas na *interface* de forma adaptativa, ou seja, consoante o perfil de cada utilizador. Assim, a relação entre estes dois módulos é importante para um bom funcionamento do tutor inteligente.

O novo módulo de CBR gera automaticamente questões de escolha múltipla. Esta geração dá-se nos casos em que o *Evaluator* indica que não existem mais questões para apresentar na interface do sistema. Este novo módulo foi desenvolvido para se complementar o processo de aprendizagem dos estudantes. Comunica diretamente com o *Evaluator* para saber o momento ideal para a geração da nova questão. Desta forma, uma nova questão é criada para permitir aos estudantes melhorar o seu desempenho e assim continuar o seu estudo numa determinada área do conhecimento.

Os pontos positivos do trabalho realizado relacionam-se com dois aspetos principais. O primeiro deles relaciona-se com a nova capacidade do tutor inteligente em criar material de estudo, mais concretamente questões de escolha múltipla, de forma automática e adaptativa. Para o efeito, o sistema reutiliza questões

de escolha múltipla respondidas pelos utilizadores. Como se pode observar na Secção 4.3.3, a formação da nova questão baseia-se na seleção, do conjunto das questões recolhidas, da questão mais interessante.

A ferramenta *Spacy* revelou-se fundamental no desenvolvimento desta "habilidade". Para além de permitir verificar a similaridade de expressões, possibilita a obtenção de palavras com um mesmo sentido, algo muito importante para a obtenção dos sinónimos. Assim, evitam-se abordagens alternativas, como por exemplo a realização de *scrapping* a *websites* de sinónimos. Isto poderia tornar o sistema de raciocínio, e consequentemente todo o sistema de avaliação, vulnerável a possíveis falhas que poderiam ocorrer nesses *websites*.

O outro aspeto positivo é relativo à introdução de uma camada para "deteção de erros" e mudança de comportamento do tutor. A análise às questões permite recusar questões de escolha múltipla que tenham sido lançadas previamente pelo módulo de avaliação. Isto torna-se vantajoso porque o raciocínio do tutor torna-se mais completo, estando agora capaz de avaliar o resultado do seu próprio processo de raciocínio. Esta avaliação é feita através da execução de regras, que são relativas a propriedades dos estudantes. A alteração dos valores destas propriedades durante o funcionamento desta camada de verificação não é feita na base de dados, mas sim em memória. O módulo responsável pela criação e atualização destas propriedades na base de dados é o *Profiler*.

Em termos negativos, e tal como já aconteceu nas fases anteriores de desenvolvimento do *Leonardo*, não foi possível realizar testes em contexto real, pois a plataforma ainda está em fase de desenvolvimento. Apenas os responsáveis pelo desenvolvimento dos novos módulos realizaram testes aos mesmos. A partir do momento em que o tutor inteligente passe a estar em ambiente real, mais concretamente num contexto educativo, deverá ser possível compreender melhor o efeito destes novos mecanismos na avaliação dos alunos.

Como conclusão do trabalho realizado, é importante abordar o que neste momento temos em termos de raciocínio no *Leonardo*. De seguida apresentam-se as principais camadas de raciocínio.

- **Criação e manutenção dos dados dos utilizadores –** que são da responsabilidade do *Profiler*. Para além de criar novos documentos na coleção *profiles* da base de dados do *Leonardo*, atualiza estes documentos sempre que um aluno responder a uma questão de escolha múltipla e o módulo de avaliação indicar se a resposta está ou não correta.

- **Seleção adaptativa de questões de escolha múltipla –** que é da responsabilidade do módulo de avaliação, *Evaluator*. Esta seleção ocorre quando um aluno pretende começar a estudar numa determinada área do conhecimento ou avançar na sessão de *Quizz*.

- **Criação automática de questões de escolha múltipla –** que é da responsabilidade do novo módulo CBR do tutor inteligente. Esta nova camada é "ativada" assim que o *Evaluator* indicar que não existem mais questões para apresentar na *interface* do *Leonardo*.

- **Deteção de erros –** que é da responsabilidade da nova camada de raciocínio (baseado em regras) modelada pelo *Modelo de Verificação de Questões* da Figura 17. Para uma questão ser apresentada na *interface*, terá de ser considerada válida por esta nova camada.

5.2 TRABALHO FUTURO

Apesar de se terem desenvolvido os principais pontos para os trabalhos desta dissertação, alguns aspetos podem ser melhorados. No âmbito do sistema de CBR, algumas palavras do *header* da nova questão, mais concretamente os substantivos e os adjetivos, resultam de substituições de algumas palavras pelos seus sinónimos. Uma melhoria a implementar seria a substituição de palavras de outras categorizações gramaticais, que não sejam substantivos ou adjetivos, por sinónimos. Esta implementação não foi realizada nesta dissertação, uma vez que existem palavras, como por exemplo verbos, que não podem ser literalmente substituídas numa frase. Por forma a não se obter demasiados erros, apenas foram realizadas as substituições que, à partida, trariam maior confiança. Para além disto, na formação do *body* da nova questão, poderia ser interessante reforçar a forma como se decide quais as opções de resposta que o constituem. A introdução de novas medidas de similaridade para a obtenção dos *distractors* seria uma boa opção para melhorar o comportamento deste sistema de raciocínio.

Para além disto também poderiam ser incluídas novas medidas para calcular o interesse das questões de escolha múltipla, para além do número de constituintes das questões e do número de vezes que foram respondidas pelos alunos. Normalmente, uma questão criada no sistema *Leonardo* requer validação de um professor para que possa ser apresentada aos alunos. As questões de escolha múltipla geradas pelo sistema de CBR não são validadas antes de serem apresentadas na interface do sistema. O que se encontra implementado é a persistência da nova questão na base de dados e consequente apresentação da mesma na *interface*. Uma melhoria a aplicar no funcionamento do sistema de CBR seria a definição de um mecanismo automático para validação das questões criadas. Assim, retirava-se o trabalho aos professores de confirmar se estes elementos de estudo se encontram nas condições necessárias para serem incluídas numa sessão de *Quizz*.

REFERÊNCIAS BIBLIOGRÁFICAS

Agnar Aamodt e Enric Plaza. Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. *AI Communications*, 7(1):39–59, 1994. ISSN 09217126.

Abdelbaset Almasri, Adel Ahmed, Naser Al-Masri, Yousef Abu Sultan, Ahmed Y. Mahmoud, Ihab Zaqout, Alaa N Akkila, e Samy S. Abu-Naser. Intelligent Tutoring Systems Survey for the Period 2000-2018. *International Journal of Academic Engineering Research (IJAER)*, 3(5):21–37, 2019.

Kevin D. Ashley. Case-based Reasoning and its Implications for Legal Expert Systems. *Artificial Intelligence and Law*, 1(2-3):113–208, 1992. ISSN 09248463.

A. A. Azeta, C. K. Ayo, A. A. Atayero, e N. A. Ikhu-Omoregbe. A Case-Based Reasoning Approach for Speech-Enabled e-Learning System. *2009 2nd International Conference on Adaptive Science & Technology (ICAST)*, páginas 211–217. IEEE, 2009. ISBN 9781424435234.

Alan Baddeley. Working Memory. *Current Biology*, 20(4):R136–R140, 2010.

Edmon Begoli. Procedural Reasoning System (PRS) Architecture for Agent-mediated Behavioral Interventions. *IEEE SoutheastCon 2014*, páginas 1–8. IEEE, 2014. ISBN 9781479965854.

Orlando Belo, João Coelho, e Luis Fernandes. An Evolutionary Software Tool for Evaluating Students on Undergraduate Courses. *ICERI2019 Proceedings*, volume 1, páginas 2711–2721, Seville, Spain, 2019. IATED.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Scott Wen-tau Yih, e Yejin Choi. Abductive Commonsense Reasoning. *ICLR 2020*, 2020.

Ramesh Bharadwaj e Steve Sims. Salsa: Combining Constraint Solvers with BDDs for Automatic Invariant Checking. Susanne Graf e Michael Schwartzbach, editores, *Tools and Algorithms for the Construction and Analysis of Systems*, páginas 378–395. Springer Berlin Heidelberg, 2000. ISBN 978-3-540-46419-8.

Blackboard. Blackboard Learn - An Advanced LMS | Blackboard, 2021. URL `https://www.blackboard.com/teaching-learning/learning-management/blackboard-learn`. Acedido em: 2021-12-18.

Albert T. Corbett, Kenneth R. Koedinger, e John R. Anderson. Chapter 37 - Intelligent Tutoring Systems. M. Helander, T. K. Landauer, e P. Prabhu, editores, *Handbook of Human-Computer Interaction*, chapter 37, páginas 849–874. Elsevier, second edition, 1997.

Ramon Lopez de Mantaras. Case-based reasoning. Georgios Paliouras, Vangelis Karkaletsis, e Constantine D. Spyropoulos, editores, *Machine Learning and Its Applications: Advanced Lectures*, páginas 127–145. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. ISBN 978-3-540-44673-6.

Explosion. spaCy . Industrial-strength Natural Language Processing in Python, 2021. URL `https://spacy.io/`. Acedido em: 2021-10-26.

M. Gebser, T. Grote, R. Kaminski, P. Obermeier, O. Sabuncu, e T. Schaub. Answer Set Programming for Stream Reasoning. *Proceedings of Answer Set Programming and Other Computing Paradigms (ASPOCP 2012), 5th International Workshop*, páginas 115–129, 2012.

Dedre Gentner e Francisco Maravilla. Analogical Reasoning. L. J. Ball e V. A. Thompson, editores, *International Handbook of Thinking & Reasoning*, páginas 186–203. Psychology Press, 2018. ISBN 9780123750006.

Vinod Goel. Cognitive Neuroscience of Deductive Reasoning. Keith J. Holyoak e Robert G. Morrison, editores, *The Cambridge Handbook of Thinking and Reasoning*, chapter 20, páginas 475–492. Cambridge University Press, 2005. ISBN 0521824176.

Christan Earl Grant e Daisy Zhe Wang. A Challenge for Long-Term Knowledge Base Maintenance. *Journal of Data and Information Quality*, 6(2-3):1–3, 2015. ISSN 19361963.

Dongxiao Gu, Changyong Liang, e Huimin Zhao. A Case-based Reasoning System Based on Weighted Heterogeneous Value Distance Metric for Breast Cancer Diagnosis. *Artificial Intelligence in Medicine*, 77:31–47, 2017. ISSN 18732860.

Giovanni Guida e Carlo Tasso. Building Expert Systems: From Life Cycle to Development Methodology. Giovanni Guida e Carlo Tasso, editores, *Topics in Expert System Design*, páginas 3–24. North-Holland, 1989.

Brett K. Hayes, Evan Heit, e Haruka Swendsen. Inductive Reasoning. *Wiley interdisciplinary reviews: Cognitive science*, 1(2):278–292, 2010.

Wei Jin, Hung Hay Ho, e Rohini K. Srihari. OpinionMiner: A novel Machine Learning System for Web Opinion Mining and Extraction. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, páginas 1195–1204, Paris, France, 2009. Association for Computing Machinery. ISBN 9781605584959.

Narendra Jussien, Guillaume Rochart, e Xavier Lorca. Choco: an Open Source Java Constraint Programming Library. *CPAIOR'08 Workshop on Open-Source Software for Integer and Contraint Programming (OSSICP'08)*, páginas 1–10, 2008.

Aditya Khamparia e Babita Pandey. A Novel Method of Case Representation and Retrieval in CBR for E-learning. *Education and Information Technologies*, 22(1):337–354, 2017. ISSN 15737608.

Janet Kolodner. *Case-based Reasoning*. Morgan Kaufmann, 1993.

Janet Kolodner e William Mark. Case-Based Reasoning. *IEEE Expert*, 7(5):5–6, 1992.

Janet L Kolodner. An Introduction to Case-Based Reasoning. *Artificial Intelligence Review*, 6:3–34, 1992.

Konstantin Korovin. iProver - An instantiation-Based Theorem Prover for First-Order Logic (System Description). Alessandro Armando, Peter Baumgartner, e Gilles Dowek, editores, *Automated Reasoning*, páginas 292–298. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-71070-7.

Tim Kraska, Ameet S. Talwalkar, John C. Duchi, Rean Griffith, Michael J. Franklin, e Michael I. Jordan. MLbase: A Distributed Machine-learning System. *CIDR 2013 - 6th Biennial Conference on Innovative Data Systems Research*, 2013.

Martin Lipscomb. Abductive Reasoning and Qualitative Research. *Nursing Philosophy*, 13(4):244–256, 2012. ISSN 14667681.

Rose Luckin. Towards Artificial Intelligence-based Assessment Systems. *Nature Human Behaviour*, 1(3):1–3, 2017. ISSN 23973374.

Wei Ji Ma, Masud Husain, e Paul M. Bays. Changing Concepts of Working Memory. *Nature Neuroscience*, 17 (3):347–356, 2014. ISSN 15378276.

Julie Main, Tharam S. Dillon, e Simon C. K. Shiu. A Tutorial on Case Based Reasoning. Sankar K. Pal, Tharam S. Dillon, e Daniel S. Yeung, editores, *Soft Computing in Case Based Reasoning*, páginas 1–28. Springer London, 2001. ISBN 978-1-4471-0687-6.

Jan Malburg e Gordon Fraser. Combining Search-based and Constraint-based Testing. *2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011)*, páginas 436–439. IEEE, 2011. ISBN 9781457716393.

David McSherry. A Case-Based Reasoning Approach to Automating the Construction of Multiple Choice Questions. *International Conference on Case-Based Reasoning*, páginas 406–420, 2010. ISBN 3642142737.

Alessandra Mileo, Ahmed Abdelrahman, Sean Policarpio, e Manfred Hauswirth. StreamRule: a Nonmonotonic Stream Reasoning System for The Semantic Web. Wolfgang Faber e Domenico Lembo, editores, *Web Reasoning and Rule Systems*, páginas 247–252. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-39666-3.

MongoDB. The most popular database for modern apps. | MongoDB, 2021. URL `https://www.mongodb.com/`. Acedido em: 2021-10-26.

Moodle. Moodle LMS Features - Moodle, 2021. URL `https://moodle.com/lms/features/`. Acedido em: 2021-12-18.

Joi L. Moore, Camille Dickson-Deane, e Krista Galyen. E-learning, Online learning, and Distance Learning Environments: Are They the Same? *Internet and Higher Education*, 14(2):129–135, 2011. ISSN 10967516.

Arash Mousavi, Md Jan Nordin, e Zulaiha Ali Othman. An Ontology Driven, Procedural Reasoning System-Like Agent Model, For Multi-Agent Based Mobile Workforce Brokering Systems. *Journal of Computer Science*, 6 (5):557–565, 2010. ISSN 1549-3636.

Roger Nkambou, Jacqueline Bourdeau, e Valéry Psyché. Building intelligent tutoring systems: An Overview. R. Nkambou, J. Bourdeau, e R. Mizoguchi, editores, *Advances in Intelligent Tutoring Systems*, páginas 361–375. Springer-Verlag Berlin Heidelberg, 2010. ISBN 9783642143625.

Hyacinth S. Nwana. Intelligent Tutoring Systems: an overview. *Artificial Intelligence Review*, 4(4):251–277, 1990.

M. J. O'Connor, H. Knublauch, S. W. Tu, e M. A. Musen. Writing Rules for The Semantic Web using SWRL and Jess. *8th International Protégé Conference*, Madrid, Spain, 2005.

Jose Dutra de Oliveira Neto e Elby Vaz Nascimento. Intelligent Tutoring System for Distance Education. *JISTEM*, 9(1):109–122, 2012. ISSN 1807-1775.

Pallets. Flask — The Pallets Project, 2021. URL `https://palletsprojects.com/p/flask/`. Acedido em: 2021-10-26.

Python.org. Welcome to Python.org, 2021. URL `https://www.python.org/`. Acedido em: 2021-10-26.

Michael M. Richter e Agnar Aamodt. Case-based reasoning foundations. *Knowledge Engineering Review*, 20 (3):203–207, 2005. ISSN 02698889. doi: 10.1017/S0269888906000695.

Stephan Schulz. E-A brainiac Theorem Prover. *AI Communications*, 15(2-3):111–126, 2002. ISSN 09217126.

Tilotma Sharma, Navneet Tiwari, e Deepali Kelkar. Study of Difference Between Forward and Backward Reasoning. *International Journal of Emerging Technology and Advanced Engineering*, 2(10):271–273, 2012.

Dong-Her Shih, Hsiu-Sen Chiang, Binshan Lin, e Shih-Bin Lin. An Embedded Mobile ECG Reasoning System for Elderly Patients. *IEEE Transactions on Information Technology in Biomedicine*, 14(3):854–865, 2010. ISSN 10897771.

Jody M. Shynkaruk e Valerie A. Thompson. Confidence and Accuracy in Deductive Reasoning. *Memory and Cognition*, 34(3):619–632, 2006. ISSN 0090502X.

Adri Smaling. Inductive, Analogical, and Communicative Generalization. *International Journal of Qualitative Methods*, 2(1):52–67, 2003. ISSN 1609-4069.

Richard Socher, Danqi Chen, Christopher D. Manning, e Andrew Y. Ng. Reasoning With Neural Tensor Networks for Knowledge Base Completion. *Proceedings of the 26th International Conference on Neural Information Processing Systems*, volume 1, páginas 926–934, Lake Tahoe, Nevada, 2013. Curran Associates Inc.

Qingyun Wang, Xiaoman Pan, Lifu Huang, Boliang Zhang, Zhiying Jiang, Heng Ji, e Kevin Knight. Describing a knowledge base. *Proceedings of the 11th International Conference on Natural Language Generation*, páginas 10–21. Association for Computational Linguistics, 2018.

Patricia Wastiau, Roger Blamire, Caroline Kearney, Valerie Quittre, Eva Van de Gaer, e Christian Monseur. The Use of ICT in Education: a Survey of Schools in Europe. *European Journal of Education*, 48(1):11–27, 2013.

Ian Watson e Farhi Marir. Case-Based Reasoning : A Review. *The Knowledge Engineering Review*, 9(4): 327–354, 1994.

Evan You. Vue.js, 2021. URL `https://vuejs.org/`. Acedido em: 2021-12-29.

Hasan Zalaghi e Mahdi Khazaei. The Role of Deductive and Inductive Reasoning in Accounting Research and Standard Setting. *Asian Journal of Finance & Accounting*, 8(1):23–37, 2016. ISSN 1946-052X.