UNIVERSIDADE DO VALE DO RIO DOS SINOS — UNISINOS
UNIDADE ACADÊMICA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA
NÍVEL MESTRADO

CLEBER SCHROEDER FONSECA

AVALIAÇÃO DO CONHECIMENTO DO ESTUDANTE EM SISTEMAS TUTORES INTELIGENTES BASEADOS EM PASSOS

Uma abordagem baseada em Deep Knowledge Tracing

SÃO LEOPOLDO
2020

Cleber Schroeder Fonseca

AVALIAÇÃO DO CONHECIMENTO DO ESTUDANTE EM SISTEMAS TUTORES INTELIGENTES BASEADOS EM PASSOS

Uma abordagem baseada em Deep Knowledge Tracing

Dissertação apresentada como requisito parcial
para a obtenção do título de Mestre pelo
Programa de Pós-Graduação em Computação
Aplicada da Universidade do Vale do Rio dos
Sinos — UNISINOS

Orientador:
Prof. Dra. Patricia Jaques Maillard

São Leopoldo
2020

F676a Fonseca, Cleber Schroeder.

Avaliação do conhecimento do estudante em sistemas tutores inteligentes baseados em passos: uma abordagem baseada em Deep Knowledge Tracing / Cleber Schroeder Fonseca. – 2020.
94 f.: il.; 30 cm.

Dissertação (mestrado) – Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, São Leopoldo, 2020.
Orientadora: Profa. Dra. Patrícia Jaques Maillard.

1. Modelo de estudante. 2. Avaliação do conhecimento. 3. Sistemas tutores inteligentes. 4. Deep Knowledge Tracing. I. Maillard, Patrícia Jaques, orient. II. Título.

CDU 004.89

Catalogação na Publicação (CIP)
Bibliotecária responsável: Carolina Kautzmann – CRB10/1604

1 - "O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior Brasil (CAPES) - Código de Financiamento 001 /"*This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001*

ATA DE BANCA EXAMINADORA DE DISSERTAÇÃO DE MESTRADO Nº 21/2020

Aluno: Cleber Schroeder Fonseca

Título da Dissertação: "AVALIAÇÃO DO CONHECIMENTO DO ESTUDANTE EM SISTEMAS TUTORES INTELIGENTES BASEADOS EM PASSOS: Uma abordagem baseada em Deep Knowledge Tracing"

Banca: Profa. Dra. Patrícia Augustin Jaques Maillard – Orientadora
Prof. Dr. Eleandro Maschio Krynski - UTFPR
Prof. Dr. Gabriel de Oliveira Ramos - UNISINOS

Aos seis dias do mês de outubro do ano de 2020, às 13h, a Comissão Examinadora de Defesa de Dissertação composta pelos(a) professores(a): Profa. Dra. Patrícia Augustin Jaques Maillard, Orientadora – UNISINOS (participação por webconferência); Prof. Dr. Eleandro Maschio Krynski, Membro da Banca - UTFPR (participação por webconferência) e Prof. Dr. Gabriel de Oliveira Ramos, Membro da Banca – UNISINOS (participação por webconferência), para analisar e avaliar a Dissertação apresentada pelo aluno Cleber Schroeder Fonseca (participação por webconferência).

Considerações da Banca:

A banca aprova o trabalho de mestrado de Cléber Fonseca, pois ele apresenta de forma consistente e com qualidade todos os itens requeridos para uma dissertação de mestrado, atendendo de forma clara o objetivo inicial do trabalho. A comissão avaliadora também destaca o emprego de algoritmos de aprendizado de máquina e o cuidado metodológico na avaliação realizada.

Ocorreu alteração do título? (x)Não ( ) Sim

Indicar o novo título: .....

.....

A Banca Examinadora, em cumprimento ao requisito exigido para a obtenção do Título de Mestre em Computação Aplicada, julga esta dissertação:

(x) APROVADA ( ) REPROVADA

Conforme Artigo 67 do Regimento do Programa o texto definitivo, com aprovação da Orientadora, deverá ser entregue no prazo máximo de sessenta (60) dias após a defesa. O resultado da banca é de consenso entre os avaliadores. A emissão do Diploma está condicionada a entrega da versão final da Dissertação. A sessão da Defesa de Dissertação ocorreu integralmente por webconferência para atender às recomendações da OMS e Ministério da Saúde com relação ao Covid-19.

São Leopoldo, 06 de outubro de 2020

Patrícia J M

Profa. Dra. Patrícia Augustin Jaques Maillard - Orientadora

Aos meus pais, **Dejair** e **Clair**,
por me ensinar o sentido da vida.
Em especial a minha mãe,
por todo afeto e dedicação que destinou
para que eu alcançasse todos os meus sonhos.

Aos meus avós **Erico** (in memoriam), **Marlene**,
João (in memoriam) e **Nazi** (in memoriam),
pela inspiração e exemplo de pessoas que são/foram.

À minha amada esposa **Natalie**
pela constante motivação, zelo e carinho,
nos momentos mais difíceis dessa caminhada
a tornasse mais tranquila,
mostrando que desistir é fácil,
mas concluir requer muito esforço.
À ti meu mais profundo amor e agradecimento.

Ao meu filho **Vinícius**,
que torna meus dias mais leves e coloridos,
seu sorriso é suficiente para por fim aos dias cinzentos.
À ti todo meu amor e disposição.

AGRADECIMENTOS

Em primeiro lugar, gostaria de agradecer a minha amada, Natalie, pelo encorajamento, paciência, compreensão e disponibilidade demonstrados durante todo o processo de desenvolvimento desse trabalho. Ao meu filho, por tornar até mesmo os momentos mais difíceis, mais fáceis, simplesmente com um olhar ou um sorriso. Amo vocês incondicionalmente e serei sempre grato pela presença de vocês na minha vida, pelas palavras de incentivo, paciência e amor que me destinam.

À minha mãe, pelo incentivo, orgulho expressado a cada vez que conversamos, e até pela pressão para a conclusão desse trabalho, sem você tudo seria um pouco mais difícil.

À minha avó, agradeço pela ternura, calma, pelo exemplo de superação, por ser essa pessoa tão especial.

Minha gratidão à Profª. Dra. Patricia Jaques Maillard, seu apoio incondicional demonstrado durante todo o longo processo de desenvolvimento desse trabalho foi essencial para que fosse concluído. Sua disponibilidade, incentivos e participação foram surpreendentes, superou a todas expectativas sobre a orientação, meu agradecimento pela vontade dedicada, mesmo nos meus piores momentos caminhou ao meu lado, apontando o caminho a ser perseguido com muita paciência e dedicação.

Agradeço também ao Instituto Federal de Educação, Ciência e Tecnologia Sul-Rio-Grandense - IFSUL, pelo período de afastamento que foi importante para que esse trabalho fosse concluído. Um muito obrigado especial aos colegas do campus Gravataí, pela compreensão e auxílio, permitindo que esse trabalho tivesse uma pitada dos profissionais que somos.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

“**E o futuro é uma astronave que tentamos pilotar,**
Não tem tempo nem piedade, nem tem hora de chegar
Sem pedir licença muda nossa vida, depois convida a rir ou chorar
Nessa estrada não nos cabe conhecer ou ver o que virá
O fim dela ninguém sabe bem ao certo onde vai dar
Vamos todos numa linda passarela
De uma aquarela que um dia, enfim, descolorirá”.
(Toquinho e Vinícius de Moraes)

RESUMO

Sistemas Tutores Inteligentes (STIs) são ambientes de aprendizagem inteligentes que, por possuírem um modelo do conteúdo a ser ministrado, assim como informações do estudante, são capazes de oferecer ensino e assistência individualizada. Os STIs são quase tão eficientes quanto o aprendizado individualizado (*one-to-one tutoring*), aquele no qual o estudante fica sozinho com um professor, recebendo sua atenção exclusiva, durante um tempo previamente fixado. Essa maior eficiência se mostra ainda mais efetiva, quando os estudantes usam STIs baseados em passos, no qual recebem *feedback* do sistema a cada passo dado para a solução de um problema e não somente após a conclusão do exercício. Para fornecer instrução e assistência individualizada, os STIs se utilizam de técnicas de inteligência artificial (IA) para identificar e modelar as dificuldades e conhecimentos de cada estudante, retornando um conteúdo de qualidade, dentro das necessidades individualizadas do estudante. O componente do sistema responsável por avaliar o conhecimento e habilidades do estudante é o modelo de estudante; esse componente é responsável tanto por identificar qual o grau de proficiência dos estudantes em cada unidade de conhecimento, como por avaliar a chance de sucesso na próxima tarefa ou passo, a fim de prover assistência. Redes Bayesianas, *Bayesian Knowledge Tracing* (BKT) e *Deep Knowledge Tracing* (DKT) são as ferramentas atuais mais aplicadas para o desenvolvimento do modelo de estudante dos STIs, sendo que a técnica DKT é a que apresenta, atualmente, o melhor desempenho para modelos de estudante que buscam predizer sucesso na próxima tarefa. No entanto, essa técnica só foi empregada em STIs baseados em respostas, ou seja, STIs em que somente é analisada a resposta final do estudante. O objetivo do trabalho desenvolvido é a utilização do DKT para prever o sucesso do estudante no próximo passo em STIs baseados em passos. STIs baseados em passos são aqueles nos quais são analisados cada passo intermediário da resolução apresentada pelo estudante, fornecendo feedback e ajudas a cada nova interação. Ao utilizar os dados de um STI baseado em passos, o sistema recebe uma quantidade maior de informações, já que para concluir um exercício, em regra geral, o estudante fornece mais de um passo, tornando o modelo de estudante baseado em passos mais complexo e eficiente. Para esse estudo, foram utilizados os dados extraídos do sistema tutor PAT2Math, o qual se trata de um STI baseado em passos que assiste os estudantes na resolução de equações de primeiro grau. Os resultados demonstram que as Redes Neurais utilizando a arquitetura *Multi-layer perceptron* alcançaram um resultado geral muito próximo dos obtidos pelos trabalhos relacionados. No entanto, quando utilizados dados de STIs baseados em passos, esse resultado é ainda melhor. Enquanto os trabalhos relacionados atingiram AUC de 79,24% com redes LSTM para STIs baseados em respostas, o trabalho proposto atingiu AUC de 50% e F1-Score de 0,8506 utilizando a mesma rede LSTM com dados de STI baseado em passos, e AUC de 76,78% e F1-Score de 0,8657, quando aplicada a MLP com os mesmos dados, permitindo identificar que tal modelo de estudante tem potencial de utilização em STIs baseados em passos.

Palavras-chave: Modelo de estudante. Avaliação do conhecimento. Sistemas Tutores Inteligentes. Deep Knowledge Tracing.

ABSTRACT

Intelligent Tutoring Systems (ITS) are intelligent learning environments that, as they have a model of the content to be taught, as well as student information, are able to offer individualized teaching and assistance. ITS are almost as efficient as one-to-one tutoring, in which the student receives individualized assistance from a teacher. This greater efficiency is shown to be even more effective when students use step-based ITS, in which they receive feedback from the system at each step taken to solve a problem and not only after the end of the exercise. To provide individualized instruction and assistance, ITS use artificial intelligence (AI) techniques to identify and model the difficulties and knowledge of each student, returning quality content, adequate for the needs of the student. The component of the system responsible for assessing the student's knowledge and skills is the student model; this component is responsible both for identifying the degree of proficiency of students in each knowledge unit, and for assessing the chance of success in the next task or step, in order to provide assistance. Bayesian Networks, Bayesian Knowledge Tracing (BKT) and Deep Knowledge Tracing (DKT) are the most applied current tools for the development of the ITS student model, and the DKT technique is the one that currently presents the best performance for student models that seek to predict success in the next task. However, this technique was only used in answer-based ITS, that is, ITS in which only the student's final response is analyzed. The aim of the proposed work is to use DKT to predict student success in the next step in a step-based ITS. Step-based ITS are those in which each intermediate step of the solving process presented by the student is analyzed, providing feedback and help with each new interaction. When using data from a step-based ITS, the system receives a greater amount of information, since to complete an exercise, in general, the student provides more than one step, making the step-based student model more complex and efficient. For this study, data extracted from the PAT2Math tutor system was used, which is a step-based ITS that assists students in solving first degree equations. The results demonstrate that the Neural Networks using the multi-layer perceptron architecture achieved an overall result very close to those obtained by the related works. However, when using step-based ITS data, this result is even better. While the related work reached AUC of 79.24 % with LSTM networks for answer-based ITS, the proposed work reached AUC of 50% and of F1-Score of 0.8506 using the same LSTM network with step-based ITS , and AUC of 76.78 % and F1-Score of 0.8657, when applied to MLP with the same data, allowing to identify that such student model has potential for use in step-based ITS.

Keywords: Student Model. Knowledge Tracing. Intelligent Tutoring System. Deep Knowledge Tracing.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ADASYN	Adaptive Synthetic
AUC	Area under the ROC Curve (Área Abaixo da Curva ROC)
BCE	Binary Cross-entropy
BKT	Bayesian Knowledge Tracing
CSS	Cascading Style Sheets (Folha de Estilo em Cascata)
CSV	Comma-separated values
DKT	Deep Knowledge Tracing
GAO	Grafo Acíclico Orientado
HMM	Hidden Markov Model
HTML	Hypertext Markup Language (Linguagem de marcação de hipertexto)
IA	Inteligência Artificial
IAC	Instrução Assistida por Computador
KC	Knowledge Components (Componentes de conhecimento)
LFA	Leraning Factor Analysis (Análise de Fatores de Aprendizagem)
LSTM	Long Short Term Memory
MAE	Mean Absolute Error
MLP	Multi-layer Perceptron
MMC	Mínimo Múltiplo Comum
MSE	Mean Squared Error
PAT2Math	Personal Affective Tutor to Math
PLM	Processamento de Linguagem Natural
RB	Rede Bayesiana
RBD	Rede Bayesiana Dinâmica
RN	Rede Neural
RNA	Rede Neural Artificial
RNR	Rede Neural Recorrente
ROC	Receiver Operating Characteristic Curve (Curva Característica de Operação do Receptor)
SA	Step Analyzer
SG	Step Generator
SGBD	Sistema Gerenciador de Banco de Dados
SMOTE	Synthetic Minority Over-sampling Thechnique
STI	Sistema Tutor Inteligente

SUMÁRIO

1 INTRODUÇÃO

O ensino individualizado, segundo Bloom (1984), é tido como a forma mais efetiva de ensino. Nessa modalidade de ensino, o professor se dedica exclusivamente a um estudante por vez, permitindo ao professor maior compreensão das dificuldades do aprendizado dos estudantes e, por isso, maior individualização das estratégias de ensino e assistência. Assim, o conteúdo e as estratégias de ensino podem ser adaptados ao ritmo de aprendizagem e conhecimento de cada estudante.

Os Sistemas Tutores Inteligentes (STIs) são ambientes inteligentes de aprendizagem que buscam prover *one-to-one tutoring*. Os STIs, devido à utilização de algoritmos e técnicas de Inteligência Artificial (IA), adéquam os problemas e conteúdos ao ritmo de aprendizagem, conhecimentos e habilidades de cada estudante. Segundo VanLehn (2011), os STIs podem ser classificados em dois grande grupos: (i) os **STIs baseados em respostas**, nos quais é analisada somente a resposta final do estudante para cada um dos problemas propostos, definindo-a como certa ou errada, e, (ii) os **STIs baseados em passos**, os quais analisam cada etapa da resolução de um problema, permitindo, assim, que o estudante identifique mais facilmente o ponto onde errou, bem como o STI forneça auxílios mais específicos e adequados às lacunas de aprendizagem do estudante.

Os STIs possuem informações individualizadas sobre cada estudante, como perfil, status do conhecimento, habilidades e características afetivas. Ainda, reúne as informações dos dados da interação dos estudantes com o sistema. O modelo de estudante reúne as informações constantes no STI para avaliar o conhecimento do estudante sobre cada componente do conhecimento trabalhado pelo STI. Esse modelo é o componente que torna possível a personalização dos STIs a cada estudante. A avaliação se dá através das respostas do estudante em cada atividade, por essa ser a única evidência que o sistema é capaz de receber do estudante (VANLEHN, 2006).

Algumas técnicas têm sido mais utilizadas para a criação do modelo de estudante, entre as quais *Redes Bayesianas Dinâmicas* (RBD), a *Bayesian Knowledge Tracing* (BKT) e a *Deep Knowledge Tracing* (DKT). As Redes Bayesianas envolvem a criação de um grafo acíclico que representa a relação de dependência entre as unidades de conhecimento envolvidas.

Bayesian Knowledge Tracing (BKT) é outra técnica utilizada, assim como a anterior, baseia-se no teorema de Bayes para calcular a probabilidade de ocorrência de um acontecimento. No entanto, nessa técnica são utilizadas somente fórmulas matemáticas que permitam uma avaliação da ocorrência ou não de um evento. Outro diferencial dessa técnica está no fato de que não há avaliação da ligação entre as unidades do conhecimento, assim, a análise do conhecimento refere-se apenas à unidade do conhecimento que está sendo abordada no exercício em execução.

Atualmente, devido ao grande crescimento do poder dos computadores, vem sendo utilizada com mais frequência as técnicas de aprendizado de máquina. Na área de modelagem de conhecimento do estudante, tem sido aplicada a técnica de aprendizado de máquina chamada *Deep Knowledge Tracing*. Ela utiliza um tipo de Rede Neural Recorrente, a *Long Short Term*

Memory, que, através de dados prévios de estudantes é capaz de aprender e, posteriormente, aplicar esse conhecimento obtido para prever a próxima interação do estudante.

Visto que o modelo de estudante utilizando a técnica DKT, até o presente momento, foi aplicado somente a STIs baseados em respostas, como poderá ser constatado no Capítulo 5, o objetivo desse trabalho se dá no desenvolvimento de um modelo de estudante utilizando a técnica DKT para STIs baseados em passos. Para desenvolver e avaliar esse modelo , serão utilizados os dados do PAT2Math (*Personal Affective Tutor to Math*), o qual se trata de um STI baseado em passos para ensino de álgebra elementar, com foco em equações de 1º grau, com uma incógnita. O PAT2Math é descrito mais detalhadamente na Seção 2.2.

1.1 Problema

Os STIs se diferenciam dos demais sistemas educacionais não inteligentes pela capacidade de adaptação da instrução e assistência às características do estudante. Isso quer dizer que o conteúdo e exercícios são propostos ao estudante com a complexidade adequada ao conhecimento do estudante, respeitando o seu ritmo de aprendizagem. A assistência individualizada ao nível de conhecimento do estudante se dá através da avaliação do conhecimento, sendo esse o grande desafio de todo o STI, uma vez que essa avaliação está intimamente ligada ao domínio do conhecimento, o que se passa em decorrência de cada área do conhecimento ter sua própria estrutura, com termos e definições próprios. Além disso, cada unidade do conhecimento pode ter relação com as demais, exercendo, assim, uma influência que depende de caso para caso.

Em álgebra, essa relação entre os tópicos é mais fácil de ser observada, visto que as operações algébricas possuem vários tipos de relação, como por exemplo precedência e composição. Essa relação de precedência define quais componentes devem ser aprendidos antes dos demais e a composição é quando uma operação é resolvida através da utilização de outras operações. Um exemplo da composição é a operação de soma de frações, a qual em um primeiro momento, precisa calcular o MMC (Mínimo Múltiplo Comum) dos denominadores das frações a serem somadas, para, em seguida, dividir esse valor do MMC por cada um dos denominadores da fração e, por fim, multiplicar o resultado dessa divisão pelo numerador da mesma fração, para somente depois disso poder calcular a soma das frações.

A análise dos conceitos algébricos foi feita e analisada por Seffrin (2015), através de entrevistas com professores de matemática e levantamento de trabalhos, criando uma RBD para avaliar o conhecimento do estudante. No entanto, essa técnica apresenta um desempenho aquém do desejado para uma ferramenta WEB em tempo real. Além disso, novas técnicas mais eficientes têm sido sucessivamente empregadas, tais como as *Bayesian Knowledge Tracing*, que não levam em consideração essa ligação entre os conceitos.

Ainda, conta-se com o *Deep Knowledge Tracing*, que utiliza redes neurais para identificar o conhecimento do estudante, se valendo de dados de estudantes que já utilizaram a ferramenta para treinamento para posterior avaliação do conhecimento dos futuros estudantes. Porém, a

técnica DKT somente foi implementada para STIs baseados em respostas, obtendo um resultado expressivo, chegando a alcançar 25% de ganho sobre a técnica BKT aplicada aos mesmos dados. No entanto, ainda não foi experimentado com dados de um STI baseado em passos. Assim, surge o problema que será explorado nessa pesquisa, a técnica DKT é igualmente efetiva para prever sucessos do estudante em futuros passos em STIs baseados em passos?

1.2 Questão de Pesquisa e Objetivos

Após a constatação de que dentre as técnicas para a criação do modelo de estudante a que mais tem se destacado é o *Deep Knowledge Tracing*, bem como de que o mesmo ainda não foi utilizado em STIs baseados em passos, o objetivo se dará no desenvolvimento de um modelo de estudante para um STI baseado em passos. Além disso, é preciso identificar qual a melhor configuração da rede neural para realizar tarefa proposta.

Assim formula-se a seguinte questão de pesquisa: *"Entre as Redes Neurais Recorrentes utilizando Long-Short Term Memory e as Redes Neurais Artificiais utilizando Multi-layer Perceptron, identificar qual a melhor rede neural para a construção de um modelo de estudante para STIs baseados em passos, além disso identificar a melhor configuração da rede para essa finalidade?"*.

O objetivo se molda na construção de um modelo de estudante utilizando aprendizado de máquina em redes neurais para prever se o estudante acertaria ou não o próximo passo de uma tarefa. Para tanto, os modelos serão avaliados quanto a sua *area under the curve* (AUC), bem como utilizando a métrica *F1 Score*, que associa a precisão e a revogação.

A fim de verificar essa questão de pesquisa, foram desenvolvidas as seguintes atividades:

- Implementar os modelos de Deep Knowledge Tracing.
  - Utilizando Rede Neural Recorrente (RNR).
  - Utilizando Rede Multi-layer Perceptron (MLP).
- Realizar os testes com os modelos citados e testar as configurações dos mesmos a fim de alcançar a melhor avaliação.

1.3 Organização do documento

Este documento está organizado em capítulos, cuja organização e conteúdo são descritos a seguir. O capítulo 2 versa sobre os Sistemas Tutores Inteligentes. Nessa etapa do trabalho são apresentados os conceitos dos principais componentes, assim como o processo de funcionamento dos STIs. Ao final, apresenta-se o STI PAT2Math, cujos dados serão utilizados por essa dissertação, adentrando-o, bem como descrevendo seus principais componentes de funcionalidades.

No capítulo 3 são descritos os conceitos de modelo de estudante, em especial, serão exemplificados os principais tipos. Em seguida, são analisados os principais métodos de avaliação do conhecimento do estudante, como Redes Bayesianas, *Bayesian Knowledge Tracing* e *Deep Knowledge Tracing*, os quais serão empregados no presente trabalho.

O capítulo 4 além de apresentar os conceitos de redes neurais (RNs), adentra as configurações que podem ser utilizadas para aperfeiçoar, e colocar em funcionamento as RNs. Ao fim deste capítulo serão explicitados alguns tipos de RNs como as Redes Neurais Recorrentes (RNR), as Redes Multi-layer Perceptron (MLP), entre outras.

O capítulo 5 descreve os principais trabalhos relacionados a essa dissertação, os quais envolvem modelos de avaliação do conhecimento do estudante usando *Deep Knowledge Tracing*.

O capítulo 6 detalha a metodologia adotada neste trabalho para o desenvolvimento do modelo de estudante utilizando DKT, tais como: a etapa de pré-processamento dos dados, a criação dos modelos, além das métricas utilizadas para avaliação dos modelos.

No capítulo 7 os resultados obtidos serão explorados e discutidos, buscando identificar qual seria a melhor técnica para a construção dos modelos, bem como qual seria a melhor configuração para a RN. Ao fim, o capítulo 8 traduz as conclusões que foram alcançadas após a construção desse trabalho, além de apontar caminhos para a continuação do mesmo em trabalhos futuros.

2 SISTEMAS TUTORES INTELIGENTES

A educação formal era baseada em técnicas alheias aos sistemas computacionais, as quais se baseavam no uso praticamente exclusivo de livros e lições escritas em caderno. No entanto, com o advento da computação e sua consequente evolução, os computadores passaram a ser uma realidade introduzida, também, para fins educacionais, auxiliando estudantes e mestres a desenvolver potencialidades antes, até mesmo não exploradas.

Na sua introdução nos métodos educacionais, a área computacional foi denominada como Instrução Assistida por Computador, ou IAC. Através de evolução na pesquisa e na aplicação de técnicas de Inteligência Artificial buscando a instrução individualizada, já no início da década de 70, pesquisadores passaram a utilizar o termo Sistemas Tutores Inteligentes, ou STI, para substituir a então IAC, como uma evolução dela.

Mais precisamente, os STIs são sistemas computacionais que objetivam alcançar a instrução individualizada para uma área específica de conhecimento. Tais sistemas empregam técnicas e algoritmos de inteligência artificial para modelar o conhecimento a ser ensinado, o que significa que devem ser capazes de solucionar os problemas que serão propostos aos estudantes, até mesmo os mais complexos, para personalizar a instrução e compreender as dificuldades enfrentadas pelos estudantes.

Em virtude de o sistema ser detentor do conhecimento necessário para resolver o exercício, passo a passo, ele também deve ser capaz de retornar mensagens explicativas quando o estudante fornecer uma resposta incorreta ou requisitar o auxílio do sistema (WOOLF, 2007). Além da possibilidade de requisição pelo estudante de auxílio para desenvolver quaisquer atividades, o sistema deve ser capaz de identificar erros repetitivos e/ou sucessivos do estudante, retornando, automaticamente, mensagem para mostrar o caminho que esse deve traçar para compreender o conteúdo.

Além disso, o sistema tutor deve também mostrar ao estudante a resolução de um passo específico ou sugerir como deve proceder para resolvê-lo (VANLEHN, 2006). Como dito acima, de forma geral, os STIs são sistemas inteligentes, o que significa que se valem de um sistema especialista, o qual representa o conhecimento daquele domínio. Tal capacidade permite que os STIs auxiliem os estudantes de forma mais proveitosa, já que ao possuir a plenitude do conhecimento, são capazes de identificar as dificuldades enfrentadas pelos mesmos, fornecendo formas de aprendizado mais individualizadas a cada estudante.

No que se refere à interação entre um estudante e o STI, o sistema deve ser capaz de mapear as suas características, especialmente, buscar identificar as dificuldades e as facilidades que o estudante possui com relação à disciplina que está a estudar. Dentre as características do estudante levantadas por um STI, se incluem: (i) os conceitos aprendidos, suas habilidades; (ii) a forma como esse estudante aprende melhor, com textos ou vídeos; (iii) os estados afetivos do estudante, (JAQUES et al., 2011) entre outros. A forma como um STI realiza a avaliação do conhecimento e de outras características do estudante é obtida através de cálculos probabilísticos

(CORBETT; ANDERSON, 1992) ou através de técnicas de Inteligência Artificial, tais como as Redes Bayesianas (MILLÁN et al., 2013). A inferência é realizada por um componente chamado Modelo de estudante, o capítulo 3 explorará de forma detalhada esse componente.

Não basta o uso individualizado das características do estudante associadas ao conhecimento pleno do STI para se alcançar o sucesso na busca do conhecimento pelo estudante. Para ter-se êxito no objetivo traçado, há a necessidade de aliar as estratégias pedagógicas ao STI e as características do estudante. Aliando-se as estratégias pedagógicas, o sistema será capaz de tornar o aprendizado adaptável ao estudante, já que são essas técnicas as responsáveis por identificar as melhores formas de instruir o estudante e de identificar os recursos necessários para assimilação do conteúdo para cada perfil de estudante.

A forma como o conteúdo será enviada até o estudante é denominada de conjunto de instruções, as quais nada mais são do que todo texto, vídeo ou exercícios relacionados ao domínio do sistema que pode ser empregado para o auxílio do estudante. É nesse ponto que as técnicas pedagógicas são ainda mais importantes, já que o programador possui conhecimento para desenvolver o sistema, mas de forma geral, não conhece as técnicas capazes de desenvolver as habilidades do estudante. Uma ação em conjunto com um profissional da área pedagógica torna o sistema muito mais eficiente e capaz de alcançar o seu objetivo com mais eficiência.

Assim, pode-se afirmar que o desenvolvimento de um STI não demanda somente conhecimento de computação, devendo homogeneizar os conhecimentos advindos das áreas da Psicologia e da educação (WOOLF, 2007). Cada uma das áreas envolvidas na construção dos STIs pode ser visualizada na Figura 1.

No contexto estudado, a computação entrega a parte tecnológica necessária ao desenvolvimento do sistema. Dentro das responsabilidades atreladas à computação estão a IA que será responsável pelo conhecimento do sistema e do estudante; os recursos de multimídia que tornam o aprendizado do estudante mais dinâmico; e a internet que fornece ao sistema uma certa mobilidade. Já a Psicologia será responsável pela forma de utilização das ciências cognitivas para identificar, por exemplo, como as pessoas pensam, como assimilam melhor cada conteúdo, seus estados emocionais, entre outros (JAQUES et al., 2011). Por fim, a educação fornece todo o suporte do domínio do conhecimento ao sistema, juntamente com as questões de como ensinar, quais os conteúdos ministrar, qual a metodologia utilizar, como avaliar o estudante, além da forma de intervir para auxiliar quando determinado conteúdo não for compreendido pelo estudante. Na Figura 1 podemos observar a interdisciplinaridade envolvida para a construção de um STI.

2.1 Funcionamento de um STI

O funcionamento dos STIs pode ser identificado ao se observar seu comportamento. Enquanto os estudantes interagem com o sistema, o sistema executará uma série de tarefas para acompanhar o estudo do estudante. Vanlehn (2006) estruturou o funcionamento de um STI em

Figura 1: Interdisciplinaridade dos Sistemas Tutores Inteligentes

Fonte: Traduzido de Woolf (2007)

duas grandes “iterações”, traduzidas do inglês *loops*: ***inner loop*** e ***outer loop***.

Primeiramente, a principal função do *outer loop* é selecionar problemas para o estudante. Para um sistema bem desenvolvido, os STIs se preocuparão em selecionar o melhor problema para o nível do estudante, atrelando a problemática proposta ao nível em que o estudante se encontra. Um sistema que não relaciona o nível do estudante com os exercícios propostos será menos efetivo, pois o estudante poderá analisar conteúdos aquém ou além de seu nível.

No *outer loop*, quatro abordagens de escolha das tarefas têm sido utilizadas. A mais simples permite ao estudante selecionar o próximo problema, valendo-se de um menu que abarca todos os problemas para que o estudante possa selecionar aquele que melhor lhe convêm. A segunda atribui ao estudante uma sequência fixa de problemas, na qual o estudante deverá obedecer a ordem que lhe é apresentada. A terceira utiliza um método instrucional chamado ***mastery learning***, onde o domínio é organizado como uma sequência de níveis de dificuldade. Nessa sistemática, o sistema atribui problemas de um determinado nível até que o estudante demonstre ter obtido conhecimento suficiente para iniciar o próximo nível. Então, o sistema entregará problemas cada vez mais complexos, de acordo com os níveis alcançados pelo estudante. Por fim, a quarta é baseada em um método instrucional denominado ***macroadaption***. Tal método é ainda mais complexo que os demais; nele o sistema reconhece os conhecimentos necessários para resolver um determinado problema, além de identificar quais conhecimentos o estudante já domina. Dessa forma, o sistema poderá fornecer ao estudante problemas com conteúdos que ele já sabe associados a conteúdos que ele ainda não domina, avaliando, consecutivamente,

o quanto o estudante consegue evoluir. O sistema também tem a capacidade de selecionar problemas reconhecidos por especialistas como forma de remediar possíveis *misconceptions* (falsas concepções).

Ao contrário do *outer loop*, o mecanismo de *inner loop* funciona sobre cada um dos passos para a solução de um problema. Os serviços mais comuns executados dentro de um *inner loop*, segundo Vanlehn (2006), são:

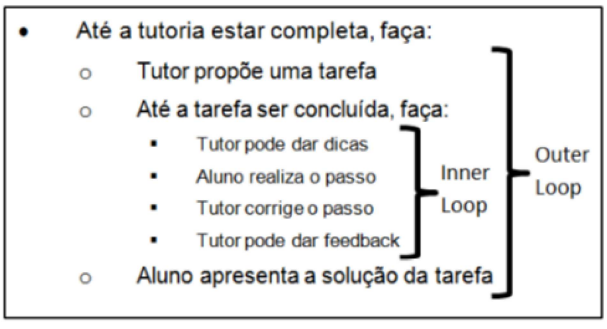
- O sistema pode oferecer **dicas** para o estudante durante a execução de um passo. Essas dicas podem ser a solução do passo ou mais genéricas, como lembrar um conceito. Essas podem ser mostradas automaticamente ao estudante ou através de solicitações do mesmo.

- O sistema pode a qualquer momento fornecer ao estudante ***feedback*** do passo em questão. Esse pode ser exibido de algumas formas como: *a*) imediato, que é mostrado ao estudante no fim de cada passo; *b*) atrasado, é mostrado após um determinado tempo de que o estudante resolveu o passo. Por exemplo, apenas no final da tarefa; *c*) sob demanda, é mostrado sempre que o estudante solicitar. Alguns outros métodos também podem ser adotados.

- A **avaliação do conhecimento** tem como base as informações obtidas na correção de cada passo e depende da granularidade desta avaliação. Ela pode ser grossa, que fornece um valor que aponta a competência do estudante no geral, ou fina, a qual trabalha com probabilidades do grau de conhecimento para cada um dos componentes do conhecimento do domínio que estão associados ao problema em questão.

As estruturas de funcionamento do *outer loop* e do *inner loop* podem ser visualizadas na Figura 2, que ilustra as ações do STI enquanto o estudante responde aos problemas propostos.

Figura 2: Iterações *outer loop* e *inner loop*



Fonte: Adaptado de Vanlehn (2006).

Em relação à presença de *outer* e *inner loops*, os STIs podem ser classificados em *step-based tutoring system* (sistema tutor baseado em passos), os quais, segundo VanLehn (2011) são sistemas que possuem tanto o *outer loop* quanto o *inner loop*, apresentando maior eficiência quando comparado com os sistemas que possuem apenas *outer loop*, por ele chamados de

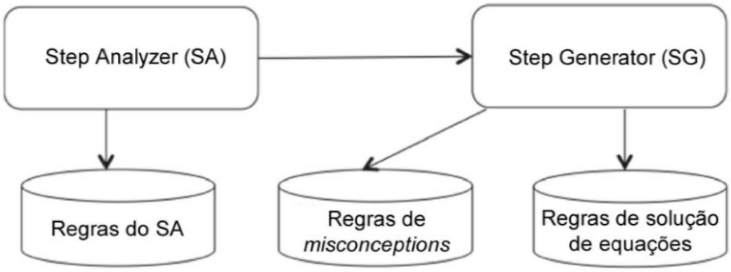
answer-based tutoring system (sistema tutor baseado em respostas). O sistema tutor baseado em passos é considerado superior em decorrência de uma maior granularidade no decorrer das interações do estudante com o sistema, permitindo assim que o mesmo retorne auxílios mais úteis ao estudante.

Assim, pode-se concluir que os STIs mais eficientes são aqueles que abrangem diversas áreas do conhecimento, tais como a psicologia, a pedagogia e a computacional. Através de tal interatividade se consegue constituir um sistema mais completo na busca por auxiliar os estudantes de determinada área. Ademais, é possível afirmar que o sistema que melhor se mostra apto para retornar respostas positivas quanto ao conhecimento do estudante é aquele que apresenta os testes corretos para o nível em que o estudante se encontra, levando em consideração uma sucessão de fatores que serão visualizados mais a frente no Capítulo 3.

2.2 PAT2Math

Adentrando no STI em desenvolvimento, observamos o PAT2Math (Personal Affective Tutor to Math), o qual é um Sistema Tutor Inteligente que auxilia os estudantes com o aprendizado de equações do 1º grau que possuem uma incógnita. O PAT2Math permite ao estudante resolver as equações passo-a-passo e a receber a correção do passo em que se encontra em tempo real. Caso o estudante solicite ajuda, o sistema tem a capacidade de oferecer uma dica, demonstrando como resolver aquele passo da equação, e em caso de erros fornecer um *feedback* de como corrigir o passo incorreto.

Figura 3: Arquitetura do sistema especialista de PAT2Math



Fonte: Traduzido de Jaques et al. (2013)

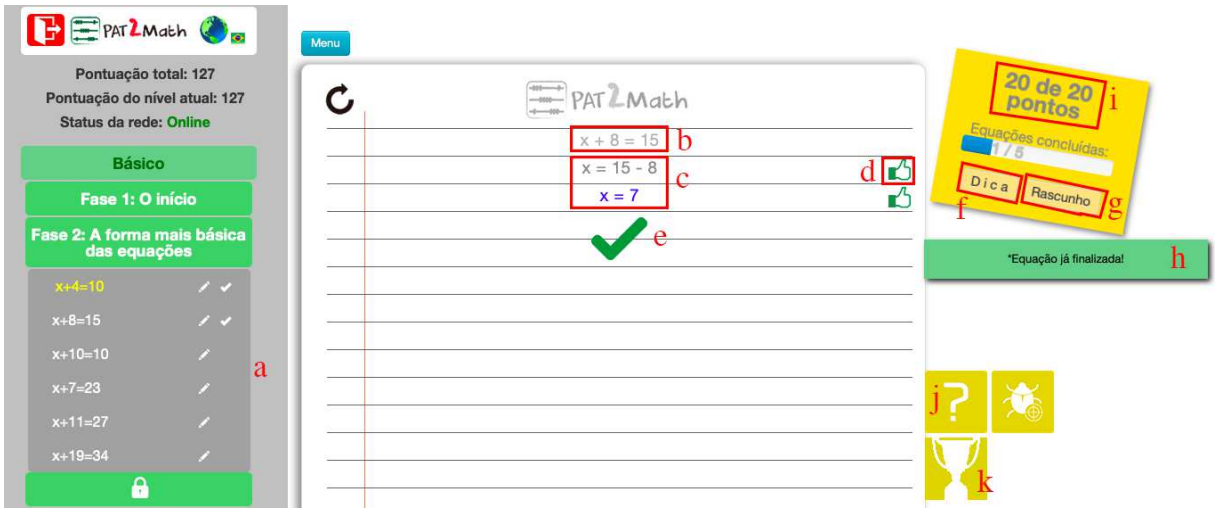
O PAT2Math possui um sistema especialista que se encarrega de corrigir as equações, além de verificar cada um dos passos do estudante. A arquitetura deste sistema especialista pode ser visualizada na figura 3. Segundo Jaques et al. (2013), o sistema especialista do PAT2Math possui os módulos *step analyser* (SA) e *step generator* (SG), nomes dados por (VANLEHN, 2006). O SG é o responsável por gerar todos os vários passos possíveis para solucionar uma equação específica. Também é ele o responsável por gerar as dicas que podem ser dadas ao estudante em cada passo. Já o SA compara a resposta do estudante com todos os passos gerados pelo SG e, além disso gera uma classificação de “correto” ou “incorreto”.

O SG do PAT2Math possui um **Módulo Resolvedor**, ou PATsolver como foi chamado inicialmente, sendo desenvolvido como um software independente do PAT2Math (SEFFRIN et al., 2009), e que foi integrado posteriormente. O módulo resolvedor possui a capacidade de solucionar uma equação completa utilizando uma *shell* de sistema especialista. Neste módulo do sistema, são implementadas regras que representam todas as soluções possíveis de uma operação de 1º grau com uma incógnita

Já o SA do PAT2Math compreende a junção de dois componentes, o **Módulo Cognitivo** que utiliza as regras do Módulo Resolvedor a fim de comparar se a solução enviada pelo estudante condiz com alguma das criadas pelo módulo, executando, assim, o *model tracing*. O módulo cognitivo, ademais, possui um conjunto próprio de regras que objetivam agilizar o processo de averiguação da resposta dada pelo estudante para o passo que está resolvendo. Assim, o sistema se torna capaz de verificar se a resposta do estudante está correta, além de retornar à operação que o estudante se valeu para solucionar cada passo da equação (SEFFRIN; RUBI; JAQUES, 2011).

Ainda em relação ao SA, um outro componente utilizado é o **Módulo de Falsas Concepções**, o qual busca identificar as *misconceptions* aplicadas pelo estudante quando a solução apresentada é incorreta. Para alcançar esse objetivo, um terceiro conjunto de regras é empregado (SEFFRIN; RUBI; JAQUES, 2011). Esse módulo se mostra relevante para identificar falhas que não são exatamente erros, mas sim concepções que estão corretas em alguns contextos, mas não em outros (MATZ, 1982 apud SEFFRIN, 2015).

Figura 4: Interface do PAT2Math + PATequation



O PAT2Math conta, ainda, com um editor de resolução de equações, denominado PATequation, que possibilita ao estudante a resolução dos passos da equação, tornando mais simples a interação do estudante com o STI (JAQUES et al., 2013). Na interface vista na Figura 4, são mostradas as seguintes áreas da interface do PAT2Math (PATequation): *a*) exibe uma lista de problemas na qual o estudante poderá escolher qual desejará solucionar. O desbloqueio dos

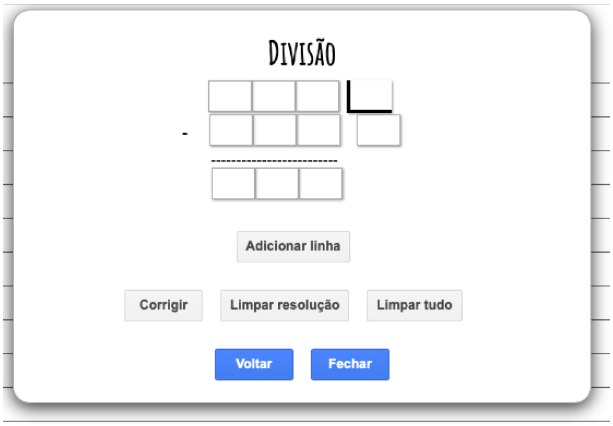
níveis acontece quando o estudante acertar todos os problemas do nível anterior, ou quando o sistema identificar através das regras previamente criadas, que o estudante já obteve o conhecimento necessário para passar ao próximo nível; *b*) exibe a equação que o estudante deve resolver; *c*) aponta os passos que o estudante respondeu para finalizar a atividade; *d*) retorna um *feedback* informando que cada um dos passos do estudante está correto ou incorreto; *e*) exibe um *feedback* informando ao estudante que ele finalizou a resolução da equação; *f*) trata de um botão para o estudante solicitar uma dica; *g*) apresenta um botão que permite ao estudante rascunhar uma operação; *h*) será o local onde serão mostradas as dicas solicitadas pelo estudante; *i*) se refere ao sistema de pontos que é usado no sistema de gameficação; *j*) trata do botão que abrirá uma ajuda do sistema; *k*) exibe o botão que abre uma janela *pop-up* que mostra o ranking dos estudantes.

Figura 5: Seleção da operação para rascunhar

Quando o estudante utiliza o botão de rascunho, abre-se a tela mostrada na figura 5, permitindo que o mesmo selecione uma das cinco operações disponíveis no sistema: Adição, Subtração, Multiplicação, Divisão ou MMC. Ao clicar em uma das operações, será mostrada a tela exibida na figura 6. No exemplo mostrado, é exibida a conta de divisão, e assim como nesta tela, as telas das demais operações tem a exibição adequada para a operação em questão. Este é o local onde o estudante pode inserir os dados e em seguida verificar a resposta da conta feita.

Por fim, essa é a forma como o PAT2Math busca representar o ensino de equações do 1º grau com uma incógnita utilizando os módulos do sistema para identificar todos os possíveis passos para a solução de um problema, levando em consideração as possíveis falsas concepções do estudante, para posteriormente comparar as respostas dadas pelo estudante com as geradas pelo sistema, além de identificar se o estudante obteve êxito em acertar o passo em questão. Ainda foi mostrada a interface gráfica do sistema com a área para solução da equação, além de exibir as telas de rascunho onde é possível escrever a resolução de uma operação de adição, subtração, multiplicação, divisão ou MMC, com a correção imediata da operação, permitindo ao estudante experimentar as contas antes de responder ao STI PAT2Math.

Figura 6: Rascunho da conta de divisão



3 MODELO DE ESTUDANTE

Quando se fala de Modelo de estudante tratasse de uma referência a um dos componentes da arquitetura tradicional dos Sistemas Tutores Inteligentes (STIs) (WENGER, 1987). Sua finalidade é o acompanhamento do progresso obtido pelo estudante durante sua interação com o material de aprendizagem, além de ser o instrumento permissor de um sistema que estabelece a preocupação com a evolução de um estudante no sistema (SELF, 1998). Para fins de exemplo, através de um sistema que se vale do modelo de estudante seria possível buscar a solução da correção de erros eventualmente cometidos por um estudante através do fornecimento de um *feedback* personalizado, sugerindo-lhe, até mesmo, a aprendizagem de um item específico adequado ao estudante individualmente, levando em conta o nível do seu conhecimento naquele momento.

3.1 Tipos de Modelo de estudante

No intuito de compreender o modelo de estudante, serão apresentados alguns de seus tipos amplamente utilizados na literatura (MILLÁN; LOBODA; CRUZ, 2010).

- Modelo de sobreposição (*overlay model*) (CARBONELL, 1970 apud MILLÁN; LOBODA; CRUZ, 2010): Nessa abordagem, o conhecimento do estudante é considerado um subconjunto adequado do conhecimento de todo o domínio. Diferenças no comportamento do estudante em relação ao comportamento de alguém com conhecimento perfeito são tratadas como uma indicação de lacunas de conhecimento. Este modelo funciona razoavelmente bem, sempre que o principal objetivo do sistema tutor é transmitir conhecimento do sistema para o estudante. A maior dificuldade do sistema é o de não considerar o fato de que o estudante pode ter crenças equivocadas (*misconpceptions*) e, portanto, dessa forma, não conseguir lidar com o erro.

- Modelo diferencial (*differential model*) (BURTON, 1982 apud MILLÁN; LOBODA; CRUZ, 2010): É uma variação no modelo de sobreposição. No modelo diferencial, o conhecimento de domínio é ramificado em necessário e desnecessário (ou opcional). Esse modelo pode ser considerado como o modelo de sobreposição, mas definido em um subconjunto do conhecimento do domínio.

- Modelo de perturbação (*perturbation model*) (BROWN; BURTON, 1978 apud MILLÁN; LOBODA; CRUZ, 2010): Nesse modelo, o conhecimento do estudante é classificado em "correto" e "incorreto". Geralmente, é construído sobre o conhecimento do domínio, aumentando com os erros mais cometidos pelos estudantes. O modelo do estudante é então o modelo de sobreposição de conhecimentos corretos e incorretos. O conhecimento incorreto é dividido em falsas concepções (misconceptions) e erros. A coleção de erros incluídos em um modelo de perturbação é geralmente chamada de biblioteca de *bugs* e

pode ser construída por análises empíricas de erros (também chamada de enumeração) ou pela geração de erros a partir de um conjunto de falsas concepções comuns (ou somente técnicas generativas). O modelo de perturbação fornece melhores explicações do comportamento do estudante, sendo mais caro de construir e manter.

- Modelo baseado em restrições (*constraint-based model*) (MAYO; MITROVIC; MITROVIC; MITROVIC; KOEDINGER; MARTIN; OHLSSON, 2001; 2003; 2003; 1994 apud MILLÁN; LOBODA; CRUZ, 2010): O tipo mais comum de conhecimento modelado é o conhecimento proposicional (por exemplo, "primatas são mamíferos"). No entanto, existem representações alternativas. Exemplificando, em modelos baseados em restrições, o conhecimento do domínio é representado por um conjunto de restrições identificadas sobre o estado do problema. Esse conjunto de restrições identifica soluções corretas e o modelo do estudante é um modelo de sobreposição sobre esse conjunto. A vantagem desse modelo é que, a menos que uma solução viole pelo menos uma restrição, ela é considerada correta. Isso permite que o estudante empregue diferentes formas de resolução de problemas, incluindo aquelas não previstas pelos engenheiros do modelo de estudante.

- Rastreamento do conhecimento (*knowledge tracing*) e rastreamento do modelo (*model tracing*) (ANDERSON et al.; BAKER; CORBETT; ALEVEN; CONATI; GERTNER; VANLEHN; CORBETT; ANDERSON; KASURINEN; NIKULA; PARDOS; HEFFERNAN, 1995; 2008; 2002; 1992; 2009; 2010 apud MILLÁN; LOBODA; CRUZ, 2010): É importante analisarmos esses dois modelos conjuntamente, já que o rastreamento do conhecimento tenta determinar o que o estudante sabe, incluindo as falsas concepções que podem ter. É uma abordagem para avaliação. O rastreamento do modelo tenta entender como o estudante resolve um determinado problema. É uma abordagem para planejar o reconhecimento. O rastreamento do modelo é especialmente útil em sistemas destinados a fornecer orientação quando o estudante atinge um impasse. O rastreamento do conhecimento também é útil como uma ferramenta de avaliação e uma ajuda de decisões pedagógicas.

3.2 Avaliação do conhecimento do estudante

Sabendo de tais formas de modelar o conhecimento, pode-se ter a falsa percepção de ser simples a criação de um sistema baseado em modelo de estudante. Entretanto, essa criação envolve inúmeros desafios, já que o conhecimento de um estudante não é binário, dependendo de inúmeras variáveis. A maior dificuldade no desenvolvimento desse sistema se encontra exatamente em determinar o nível do conhecimento, bem como as dificuldades enfrentadas daquele que se utiliza do sistema, o que é chamado de **avaliação do conhecimento**. E mais, o conhecimento não é isolado. Em várias áreas, o objeto a ser aprendido tem partes vistas anteriormente para que possa ser compreendido, chamado de conhecimento prévio. Por exemplo na mate-

mática, para saber equações, primeiro o estudante deve conhecer as operações básicas, soma, subtração, multiplicação e divisão, para só depois poder evoluir.

Além disso, nessa avaliação do conhecimento, também devem ser avaliadas as falsas concepções, que é quando o estudante erra o conteúdo por ter compreendido de forma inadequada o que lhe está sendo passado. Ainda devem ser analisados os erros em conteúdos que já foram definidos como assimilados pelo estudante, chamados *slips*, que ocorrem em sua grande maioria por desatenção do estudante. Outo conceito que deve ser levado em consideração ao modelar o conhecimento do estudante é o acerto no "chute", quando o mesmo acerta a resposta mesmo sem conhecer o conteúdo apresentado, chamado de *guess*. Para que um STI seja desenvolvido adequadamente, ele deve considerar todos esses conceitos na sua criação, sempre que o objetivo for modelar o conhecimento do estudante (VANLEHN, 2008; MILLÁN; LOBODA; CRUZ, 2010).

Talvez uma das formas mais simples de avaliar o conhecimento do estudante seja contar seus erros e acertos, aplicar uma média aritmética e então obter o resultado, descobrindo, assim, se o estudante sabe ou não determinado conteúdo. Porém, esse método pode não ser o mais adequado para avaliar o conhecimento do estudante. Segundo Vanlehn (2006), se for analisado um método hipotético no qual consideremos as sequencias "E-E-A-A-A-A" de dois erros seguidos de quatro acertos e "E-A-A-E-A-A" um erro seguido de dois acertos e outro erro seguido de mais dois acertos. Agora utilizando a forma proposta neste parágrafo somando os acertos e realizando a média aritmética temos um resultado de 66% de probabilidade do estudante saber o conteúdo. Quando de fato na primeira sequência provavelmente o estudante obteve o conhecimento, enquanto, na segunda aparentemente o mesmo ainda está com alguma dificuldade em algum ponto que pode ser reforçado.

Corbett e Anderson (1994) apresentaram a técnica para avaliar o conhecimento do estudante chamada de *Knowledge Tracing*, a qual consiste em avaliar o conhecimento do estudante sempre que for possível. Imaginemos que em uma resposta a uma atividade sobre frações um estudante demonstre conhecer soma, o sistema deve identificar tal habilidade e aumentar as estimativas do estudante sobre essa habilidade. Essa técnica tem como base quatro parâmetros: (i) o conhecimento inicial do estudante $P(L_0)$ ou p-init, (ii) a probabilidade do estudante trocar do estado "não sabe" para o estado "sabe" $P(T)$ ou p-transit, (iii) a probabilidade do estudante errar por falta de atenção (*slip*) $P(S)$ ou p-slip, (iv) a probabilidade do estudante acertar a questão por "sorte" (*guess*) $P(G)$ ou p-guess. Também ficou definido que uma vez que o estudante consiga alcançar 95% de probabilidade em uma habilidade específica é considerado que o estudante dominou a mesma.

Para criação desses modelos de estudante, algumas técnicas são mais comumente empregadas, como é o caso das Redes Bayesianas e suas variações, as quais serão exploradas na seção 3.3. Igualmente importante e comumente utilizada é a técnica *Bayesian Knowledge Tracing* (BKT), através da qual busca-se identificar, usando cálculos matemáticos, se o estudante compreendeu o conteúdo. A BKT é aprofundada na seção 3.4. Uma técnica mais atual de

criar o modelo de estudante é utilizando *Deep Knowledge Tracing*, a qual emprega técnicas de aprendizado de máquina para executar essa tarefa, como pode ser visualizado na seção 3.5.

3.3 Redes Bayesianas

As Redes Bayesianas (RBs) tiveram grande concentração de estudos na década de 90, época em que pesquisas importantes passaram a utilizá-la como ferramenta, almejando a solução de problemas. Diversos estudos podem ser citados, como é o caso do diagnostico médico (SHWE et al., 1990; SPIEGELHALTER; FRANKLIN; BULL, 1990), do aprendizado de mapas (DEAN; BASYE; LEJTER, 1990), da interpretação de linguagens (CHARNIAK; GOLDMAN, 1989) e, também, da visão computacional (LEVITT; BINFORD; ETTINGER, 1990), dentre outros. O objetivo pretendido por tais estudos foi solucionar inúmeras situações problemáticas do cotidiano comum às diversas pessoas. As RBs também foram empregadas para modelar o conhecimento do estudante. Nesse momento, serão apresentados alguns conceitos básicos acerca das RBs, tais como sua estrutura e a forma de construí-las. Após, concluindo essa parte do estudo, serão analisadas as Redes Bayesianas Dinâmicas, as quais foram aplicadas no trabalho de (SEFFRIN, 2015).

3.3.1 Visão Geral

O objetivo almejado pelas Redes Bayesianas é a representação de modelos probabilísticos com a demonstração das correspondentes ligações entre duas ou mais variáveis. Comumente encontra-se a aplicação desse modelo na verificação de dados incertos (RUSSELL; NORVIG, 2009). Ao analisar os sistemas tutores que tenham por base as RBs, conclui-se que muitas incertezas surgem em virtude de que esses dificilmente têm certeza do conhecimento do estudante quanto ao conteúdo transmitido e testado através da RB, assim como se tal conteúdo foi de fato aprendido ou se o estudante usou de subterfúgios para enganar o sistema (como colar; se no dia da avaliação no sistema o estudante estava passando por alguma situação de estresse pessoal; ou similares), o que torna praticamente impossível garantir fidedignidade total ao sistema (WOOLF, 2010).

As RBs possuem a capacidade de representar essencialmente qualquer distribuição de probabilidade conjunta completa e, em muitos casos, muito concisamente (RUSSELL; NORVIG, 2009). Tal distribuição abrange as probabilidades das variáveis $X_1, ..., X_n$ dadas todas combinações possíveis de valores que podem ser utilizadas nelas. A probabilidade de cada valor das variáveis pode ser definida pela seguinte fórmula $P(X_1, ..., X_n)$ (SEFFRIN, 2015).

A RB pode ser definida como (NEAPOLITAN, 1990):

- Um conjunto $V = X_1, ..., X_n$ de variáveis aleatórias, tal que cada variável X_i assume um conjunto de valores exaustivos e mutuamente exclusivos. Essas variáveis serão representadas pelos nodos da rede.

- Um conjunto E de relações probabilísticas entre as variáveis. Essas relações serão representadas pelos arcos (ou links) na rede.

- Uma distribuição de probabilidades conjunta (DPC) P, definida em V ($P(X_1, ..., X_n)$ denota a probabilidade de que $X_1 = x_1$, e ... e $X_n = x_n$).

de tal modo que:

- O grafo $G = (V, E)$ é um grafo acíclico dirigido (GAD).

- O conjunto (V, P) satisfaz a suposição de independência condicional.

Segundo Pearl (2014), o que torna as RBs tão Bayesianas é o fato de obedecerem aos dois atributos definidores da escola bayesiana: a disponibilidade para aceitar opiniões subjetivas como substituto para dados brutos e a aderência à condicionalização de Bayes como mecanismo primário de atualização de crenças. Assim, podemos concluir que as Redes Bayesianas possuem a capacidade de calcular probabilidades através de cálculos estatísticos previamente determinados com objetivos diretos para determinado nicho e com base em regras estabelecidas para esse fim como veremos na sequência.

3.3.2 Teorema de Bayes

Como exposto anteriormente, as RBs são aplicadas para calcular dados incertos, no sistema tutor inteligente elas são capazes de avaliar o conhecimento do estudante. O responsável por realizar os cálculos das incertezas das RBs é o Teorema de Bayes. À primeira vista, a regra de Bayes não parece muito útil, eis que para se ver calculado um único termo $P(B|A)$ se fazem necessários outros três termos: $P(A|B)$, $P(B)$ e $P(A)$. Entretanto, essa regra é muito útil na prática, como melhor veremos, já que em alguns casos possibilita o levantamento de estimativas sobre esses três termos (RUSSELL; NORVIG, 2009). Levando tais informações em consideração obtemos a equação 3.1, a qual objetiva definir a probabilidade da ocorrência de $P(B|A)$.

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \tag{3.1}$$

O Teorema de Bayes pode ser empregado de forma muito positiva na sociedade, facilitando o cotidiano das pessoas, a exemplo os casos médicos, em que para diagnosticar um paciente o sistema deve utilizar como entradas as probabilidades de cada um dos sintomas das doenças conhecidas. Por exemplo, determinado médico tem o conhecimento de que a meningite causa rigidez no pescoço em cerca de 70% dos pacientes acometidos pela doença. Esse médico também estabeleceu que a probabilidade, a *priori*, de um paciente ter meningite é $\frac{1}{50.000}$ e que a probabilidade a *priori* de qualquer paciente ter rigidez no pescoço é de 1%. Sendo s a

proposição de o paciente ter rigidez no pescoço e m a proposição de o paciente ter meningite, temos:

$$P(s|m) = 0,7$$
$$P(m) = \frac{1}{50000}$$
$$P(s) = 0,01$$

$$P(m|s) = \frac{P(s|m)P(m)}{P(s)} = \frac{0,7 * \frac{1}{50000}}{0,01} = 0,0014 \tag{3.2}$$

Assim, se conclui que apenas 1 em cada 50.000 pacientes terão meningite. Percebeu-se que a rigidez no pescoço é um sintoma que evidencia a probabilidade de o paciente ter meningite, de 70%. Não obstante, a rigidez esteja presente em um alto número de casos de meningite, é muito mais provável que as causas do sintoma de rigidez no pescoço, de forma geral, sejam por outra razão que não a meningite (RUSSELL; NORVIG, 2009). Como pode-se perceber, o teorema de Bayes permite calcular uma probabilidade desconhecida através de probabilidades previamente estabelecidas.

Entretanto, em alguns casos as probabilidades individuais dos eventos $(P(X_i))$ não estão disponíveis. Mas essas podem ser obtidas utilizando o somatório de probabilidades condicionais que sejam mutuamente excludentes (PEARL, 2014). Sendo assim, temos a equação 3.3, que é uma adaptação da equação 3.1.

$$P(B|A) = \frac{P(A|B)P(B)}{\sum_i P(A|B_i)P(B_i)} \tag{3.3}$$

A equação 3.3 representa a base das técnicas Bayesianas, na qual uma hipótese H está condicionada a uma evidência e (PEARL, 2014), como foi mostrado no exemplo acima. No exemplo foi possível verificar a possibilidade de o paciente estar doente (meningite) a depender da ocorrência de uma evidência, no caso do exemplo estudado, um sintoma específico da doença (rigidez no pescoço). Por fim, pode-se concluir que o Teorema de Bayes alcança a técnica necessária para que as RBs tenham condições de realizar os cálculos probabilísticos para que são projetadas.

3.3.2.1 Representação

As Redes Bayesianas são representadas por meio de um grafo acíclico orientado (GAO), onde cada um dos nodos (vértices) desse grafo representa uma variável aleatória e os arcos (setas) demonstram uma ligação condicional entre dois nodos. Se houver um arco saindo do nodo X até o nodo Y, é dito que X é pai de Y e cada nodo X_i tem uma distribuição de probabilidade $P(X_i|Pais(X_i))$. Além disso, pode-se ter uma variável X sem nenhum arco

chegando até ela, essa possui apenas probabilidades a *priori*, ou seja, $P(X)$ (PEARL, 2014). As variáveis que não tiverem arcos chegando nelas serão denominadas nodo-raiz (*Root Node*).

Além dessas notações, tem-se também, ligado a RB, as tabelas de probabilidades condicionais (TPC), em que cada linha da TPC mostra a probabilidade condicional de cada valor de nodos para os casos de condicionamento. Um caso de condicionamento é uma possível combinação de valores para os nodos pais (RUSSELL; NORVIG, 2009). Estas probabilidades representam a força da influência de uma variável sobre as outra (PEARL, 2014).

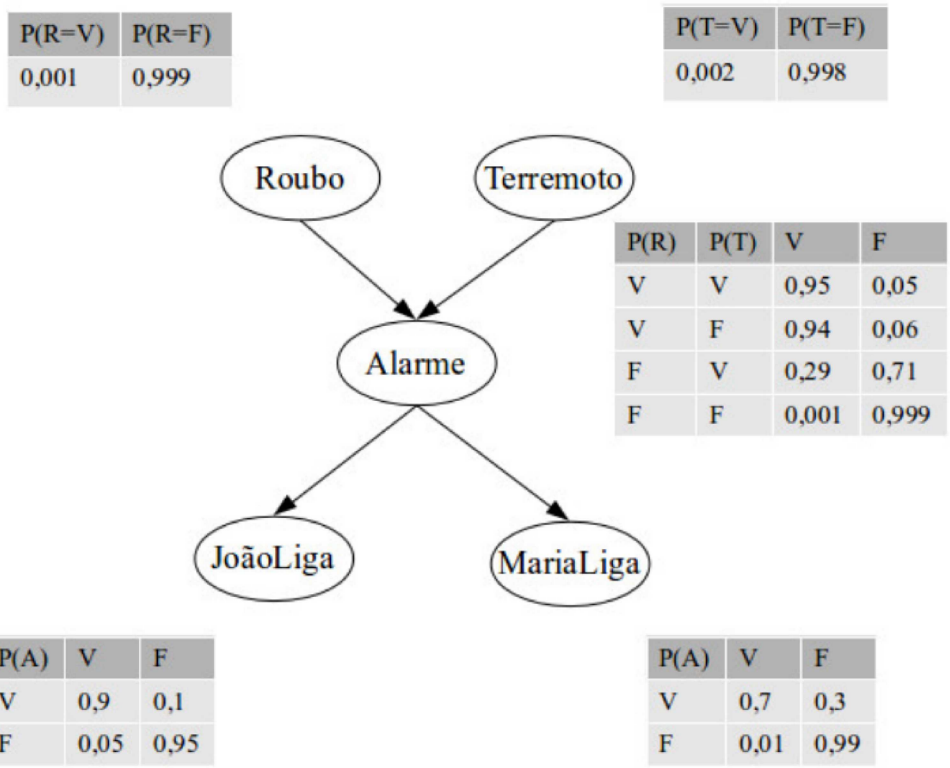
As RBs também podem definir a independência condicional entre variáveis. Este conceito refere-se à ocorrência de um evento onde a variável X tem sua probabilidade condicional definida por $P(X|Y,Z)$ e ocorre do valor da variável Z ser conhecido, tornando a informação de Y irrelevante. A independência condicional é definida por $P(X|Y,Z) = P(X|Z)$: "X é independente de Y, dada a informação de Z", ou $I(X,Z,Y)$ (PEARL, 2014). Por exemplo, sabendo o tempo em que o último ônibus passou em uma determinada parada (Y), pode-se ter uma ideia de quando o próximo virá (X). No entanto, ao tomar conhecimento da localização deste próximo ônibus (Z), a informação de Y se torna irrelevante para determinar a informação X (PEARL, 2014).

Se todos os valores da TPC $P(X_1, ..., X_n)$ são conhecidos, pode-se realizar qualquer tipo de inferência. No entanto, a elicitação da TPC é uma tarefa que requer um número de parâmetros, os quais são exponenciais ao número de variáveis (MILLÁN; LOBODA; CRUZ, 2010).

Na figura 7 é representada uma RB ilustrada em (RUSSELL; NORVIG, 2009), a qual exibe a utilização de um alarme que monitora um certo local contra roubos, sendo que o local é também suscetível a terremotos. O dono do local combinou com os vizinhos João e Maria para que ligassem para ele caso o alarme tocasse. O João quase sempre liga quando ouve o alarme, mas as vezes ele confunde o toque do telefone com o alarme e também liga ao ouvi-lo. Já Maria gosta de ouvir música com som muito alto, e em alguns casos não ouve o alarme disparar. Para modelar a RB, os acontecimentos acima citados devem ser analisados, considerando o fato de que o alarme também dispara por causa de terremotos. O conjunto dessas informações, portanto, fornece os dados que caracterizam as incertezas do modelo. Dadas as evidências de quem liga ou não liga, é possível estimar a probabilidade de um roubo.

É possível visualizar na figura 7 que não existem variáveis representando os fatos de João confundir o alarme com o telefone, tão pouco o fato de maria escutar música alta. Para utilizar a RB é necessário definir as TPC de cada um dos nodos, conforme pode ser visto na Figura 7 em que temos: o nodo Roubo com a probabilidade de 0,1% de ocorrer um roubo; e o nodo Terremoto com 0,2% de probabilidade da ocorrência de um terremoto; esses dois possuem suas probabilidade ditas a *priori*. Já o nodo alarme tem suas probabilidades baseadas nos nodos anteriores (roubo e terremoto). Então, se ocorrer um roubo e um terremoto ao mesmo tempo, existirá a probabilidade de 95% de disparar o alarme. Já caso ocorra um roubo, mas não um terremoto, a probabilidade de o alarme disparar é de 94%. Já quando não ocorre o roubo, mas ocorre o terremoto, tem-se a probabilidade de 29% de o disparo do alarme acontecer. Quando

Figura 7: Rede Bayesiana típica, mostrando a topologia e respectivas tabelas de probabilidades



não ocorre nenhum dos eventos, a probabilidade de disparo é de 0,1%.

Já o nodo JoãoLiga tem como pai o nodo alarme e responde pela probabilidade de João ligar para o proprietário quando ouve o alarme. Portanto, quando o alarme toca, João tem uma probabilidade de 90% de ligar. Ainda quando o alarme não tocar, João tem uma probabilidade de 0,5% de ligar. O nodo MariaLiga também é filho do nodo alarme, por isso também depende dos valores associados ao alarme para verificar a probabilidade de Maria ligar, caso o alarme venha a tocar, Maria tem uma probabilidade de 70% de ligar para o proprietário. Já quando o alarme não tocar, Maria tem uma probabilidade de 0,1% de ligar.

Para a construção da RB é necessário ter conhecimento do domínio do modelo, de forma a obter a melhor definição das variáveis e dos seus respectivos estados. Ainda é importante definir as relações entre as variáveis para que a construção da RB seja bem sucedida. O método de construção causal define uma relação causa-efeito entre as variáveis, no qual as causas exercem influência sobre os efeitos (PEARL, 2014). Assim, a rede da figura 7 foi desenvolvida usando este método, pois tanto "roubo" quanto "terremoto" **causam** o disparo do "alarme", e o disparo do alarme **resulta** na ligação do "João" ou da "Maria".

Definidas as relações, define-se as probabilidades das tabelas de cada nodo. Neste ponto é comum haver a participação de especialistas no domínio a ser modelado, a fim de auxiliar na definição das probabilidades a *priori* e das probabilidades condicionais (SEFFRIN, 2015). Outras formas de se obter esses valores incluem análises estatísticas (MANSKE; CONATI, 2005 apud SEFFRIN, 2015) e técnicas de aprendizado de máquina (BAKER; CORBETT; KOEDINGER;

LEE; BRUNSKILL, 2004; 2012 apud SEFFRIN, 2015). Em outros casos, certas probabilidades não podem ser estimadas através de dados disponíveis, sendo possível estimá-las através de resultados encontrados na literatura (CONATI; GERTNER; VANLEHN, 2002 apud SEFFRIN, 2015).

A fim de explicar melhor o funcionamento da RB do alarme mostrada na figura 7, a partir dos dados contidos na mesma, pode-se aferir a probabilidade de o Alarme (A) ter tocado, não havendo roubo (R), nem terremoto (T), e tanto João (J) quanto Maria (M) terem ligado (SEFFRIN, 2015).

$$\begin{aligned} P(J, M, A, \neg R, \neg T) &= P(J|A)P(M|A)P(A|\neg R, \neg T)P(\neg R)P(\neg T) \\ &= 0,90 \times 0,70 \times 0,001 \times 0,999 \times 0,998 \\ &= 0,00062 \end{aligned} \tag{3.4}$$

De mão do grafo da RB e das TPCs de cada um dos nodos dela é possível, então, verificar a probabilidade de um nodo específico tendo como evidência outros nodos da mesma rede. No exemplo utilizado para evidenciar a representação, pode-se concluir que, de posse do grafo montado, bem como das TPCs de cada nodo da RB, é possível inferir sobre os dados a chance de estar ocorrendo um roubo ou terremoto, sabendo quem foi que ligou para o proprietário, João ou Maria. Assim, resta evidenciado que a RB é capaz de retornar os resultados buscados sempre que temos conhecimento do grafo e das TPCs utilizadas.

3.3.3 Inferência

A atividade de inferência em uma Rede Bayesiana se baseia na apresentação de evidências e observação dos valores probabilísticos que as variáveis assumem. Essas evidências se tratam de variáveis onde se tem 100% de certeza da ocorrência daquele valor específico; a essa variável se dá o nome de **instanciada** (PEARL, 2014).

Sempre que uma variável é instanciada, as demais probabilidades da rede são recalculadas, tomando como base essa evidência. Sendo assim, a evidência é propagada pera o restante da rede. Em seguida, dessa propagação, pode-se concluir algumas hipóteses, tomando como base as probabilidades assumidas pelos nodos.

As RBs permitem dois tipos básicos de inferências: diagnóstico e predição. O Diagnóstico é a tarefa de, através de um conjunto de evidências, identificar as possíveis causas. As evidências nesse contexto são referidas como sintomas ou falhas. Um exemplo deste raciocínio é o diagnóstico médico, em que o foco é a identificação da doença de um paciente, com base nos resultados de exames. A predição (ou previsão), por outro lado, tenta identificar a ocorrência mais provável do evento, dada uma evidência. O diagnóstico examina o passado e o presente para raciocinar sobre o presente, enquanto a predição olha para o passado e para o presente para raciocinar sobre o futuro (MILLÁN; LOBODA; CRUZ, 2010). No caso do modelo de estudante, utiliza-se da predição para analisar qual a melhor trajetória para o desenvolvimento

do estudante.

3.3.4 Redes Bayesianas Dinâmicas

Ao longo do estudo foram visualizadas informações de probabilidades estáticas, em que cada variável aleatória possui apenas um valor específico. Por exemplo, ao consertar um carro, supõe-se que qualquer peça que esteja danificada continuará danificada durante o processo de diagnóstico. Por essa razão, a atividade é deduzir o estado do carro a partir de evidências observadas, que também permanecem fixas (RUSSELL; NORVIG, 2009).

Imaginando um paciente com diabetes, como no caso do carro, se tem como evidências doses de insulina recentes, alimentação, medição de açúcar no sangue e outros sinais físicos. Com essas evidências, é possível avaliar a quantidade de açúcar no sangue e a quantidade de insulina no organismo, podendo então tomar decisões sobre a alimentação. Só que diferentemente do caso do carro, essas evidências variam ao longo do tempo, tornando os aspectos dinâmicos do problema essenciais (RUSSELL; NORVIG, 2009). Assim como esse problema, vários outros dependem de evidências que variam. As Redes Bayesianas Dinâmicas (RBDs) são mais adequadas para tratar de situações variáveis, já que elas levam em consideração o tempo nas suas avaliações.

RBDs para (DEAN; KANAZAWA, 1989 apud CHAPELLE; ZHANG, 2009) são uma maneira de estender RBs para modelar distribuições de probabilidade sobre coleções semi-infinitas de variáveis $Z_1, Z_2, ..., Z_n$. Normalmente, essas variáveis são particionadas em $Z_t = (U_t, X_t, Y_t)$ para representar as variáveis de entrada e as de saída de um modelo estado espaço (*state-space*). São considerados apenas os processos estocásticos em tempo discreto, e neles o índice t é aumentado em 1 a cada nova observação chegada. Essa observação pode representar que algo mudou tornando-o um modelo de evento discreto. Dizer que a Rede Bayesiana é dinâmica não quer dizer que a mesma se altera ao longo do tempo, mas sim que o sistema é dinâmico.

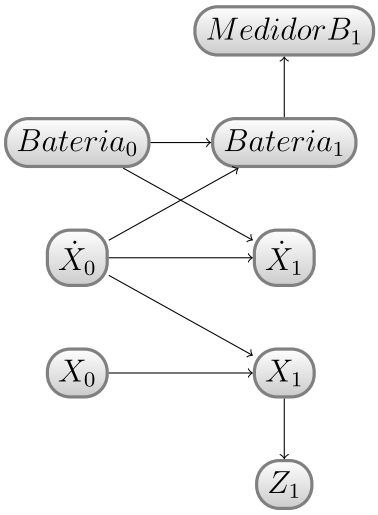
Para construção de uma RBD, três tipos de informações devem ser especificadas: a distribuição anterior sobre as variáveis de estado $P(X_0)$; o modelo de transição $P(X_{t+1}|x_t)$; e o modelo de sensores $P(E_t|X_t)$, e estes nodos receberão as evidências no instante t (RUSSELL; NORVIG, 2009).

No processo de inferência das RBDs, as evidências são apresentadas aos nodos-sensores e propagadas pela rede. Uma RBD pode ser transformada em uma RB através do processo de **desenrolamento** (*unrolling*), que consiste em inserir infinitas representações da RB, representando cada um dos instantes de tempo. Cada rede é ligada à sua sucessora através das **relações temporais** que são arcos juntando dois instantes, estabelecendo a relação condicional $P(X_{t+1}|X_t)$ para cada instante de tempo t (SEFFRIN, 2015).

Considerando o exemplo de uma RBD, retirado de (RUSSELL; NORVIG, 2009), para o monitoramento de um robô alimentado por baterias que se move no plano $X - Y$ temos que, primeiro, são necessárias as variáveis de estado, que incluirão $X_t = (X_t, Y_t)$ para represen-

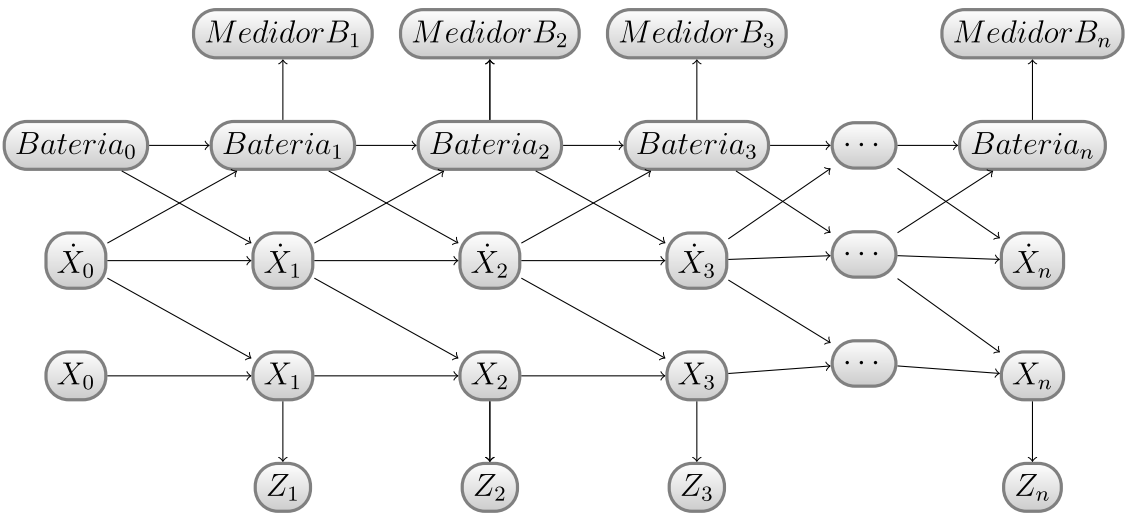
tação da posição e também $\dot{X}_t = (\dot{X}_t, \dot{Y}_t)$ que representa a velocidade. Presume-se que para medir a posição seja utilizado um GPS (*Global Positioning System*) embutido no robô, produzindo medições de Z_T. A posição no período de tempo seguinte depende da posição atual e da velocidade. A velocidade no período seguinte depende da velocidade atual e do estado atual da bateria. Foi acrescentado então $Bateria_t$ para representar o nível de carga real da bateria, que tem como pais o nível de bateria anterior e a velocidade, e também foi acrescentado um $MedidorB_t$, que mede o nível da bateria naquele momento, isso resulta na RBD mostrada na Figura 8, executando a técnica de desenrolamento na RBD citada anteriormente obtêm-se as fatias replicadas para acomodar a sequência de observações $robo_{1:3}$, conforme pode ser visto na Figura 9.

Figura 8: Rede Bayesiana Dinâmica representação de um robô alimentado por bateria



Fonte: Adaptado de Russell e Norvig (2009)

Figura 9: Rede Bayesiana Dinâmica do robô fatiada pelo processo de desenrolamento.



Fonte: Adaptado de Russell e Norvig (2009)

Como é possível perceber, a RBD se diferencia da RB por se valer da questão tempo. A RBD permite o desenvolvimento de sistemas que se modificam ao longo do tempo, permitindo, dessa forma, uma avaliação temporal da mesma. Como no exemplo do robô, pode-se avaliar o funcionamento do mesmo levando-se em consideração o nível de bateria em um instante específico, permitindo ao robô sua movimentação (ou não) sobre o plano, demonstrando o seu desempenho e consumo energético ao longo do tempo.

3.4 Bayesian Knowledge Tracing

O Bayesian Knowledge Tracing (BKT) é um método para modelar o conhecimento do estudante, o qual se dá através de um *Hidden Markov Model* (HMM), que se trata de um modelo estatístico em que o sistema modelado pode ser descrito como um processo de Markov com parâmetros desconhecidos, onde deseja-se determinar os parâmetros ocultos a partir de parâmetros observáveis. O HMM para modelar o conhecimento do estudante possui uma variável latente, utilizada através da observação exata da interação do estudante com o STI em cada uma das habilidades necessárias para se obter o domínio do conhecimento. Ele se utiliza dos quatro parâmetros citados na seção 3.2: $P(L_0)$, $P(T)$, $P(G)$ e $P(S)$ para calcular as probabilidades pretendidas. No BKT, são utilizadas as seguintes equações (YUDELSON; KOEDINGER; GORDON, 2013):

$$p(L_1)^k_u = p(L_0)^k, \tag{3.5}$$

$$p(L_{t-1}|obs_t = correto)^k_u = \frac{p(L_{t-1})^k_u.(1-p(S)^k)}{p(L_{t-1})^k_u.(1-p(S)^k) + (1-p(L_{t-1})^k_u).p(G)^k}, \tag{3.6}$$

$$p(L_{t-1}|obs_t = errado)^k_u = \frac{p(L_{t-1})^k_u.p(S)^k}{p(L_{t-1})^k_u.p(S)^k + (1-p(L_{t-1})^k_u).(1-p(G)^k)}, \tag{3.7}$$

$$p(L_t)^k_u = p(L_{t-1}|obs_t)^k_u + (1-p(L_{t-1}|obs_t)^k_u).p(T)^k, \tag{3.8}$$

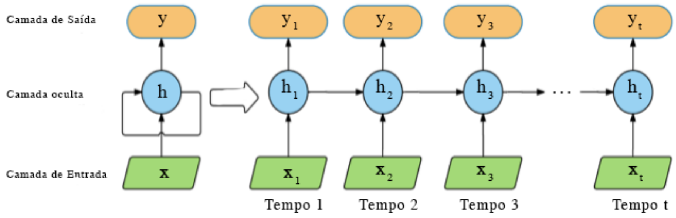
$$p(C_{t+1})^k_u = p(L_t)^k_u.(1-p(S)^k) + (1-p(L_t)^k_u).p(G)^k) \tag{3.9}$$

Sabendo que os parâmetros são definidos para todas as habilidades, as fórmulas são utilizadas para definir o conhecimento de um estudante (u) para cada uma das habilidades (k). A probabilidade inicial deste estudante para uma habilidade é dada através do parâmetro $P(L_0)$ para aquela habilidade, conforme a equação 3.5. Dependendo se o estudante aplicou a habilidade k correta ou incorretamente, a probabilidade condicional é calculada usando a equação 3.6 ou a equação 3.7 sucessivamente. Essa probabilidade condicional é aplicada para atualizar o domínio da habilidade k de acordo com a equação 3.8. E para calcular a probabilidade de o estudante acertar alguma atividade do domínio k no futuro é usada a equação 3.9.

3.5 Deep Knowledge Tracing

O *Deep Knowledge Tracing* (DKT) foi inicialmente introduzido por Piech et al. (2015) e utiliza uma Rede Neural Recorrente (RNR) para realizar a tarefa de modelar o conhecimento do estudante. Fundamentalmente, é um modelo dinâmico e flexível que conecta neurônios artificiais ao longo do tempo. A propagação das informações é recursiva, visto que os neurônios ocultos evoluem tanto com as entradas do sistema como com os dados anteriores (WILLIAMS; ZIPSER, 1989). Diferentemente da forma como o BKT é utilizado na educação, as RNRs têm apresentações dimensionais e contínuas do estado. Uma das vantagens de se utilizar essa representação mais rica das RNR é a sua capacidade de utilizar uma informação de entrada em uma previsão em um ponto muito posterior no tempo. Isso é especialmente verdadeiro ao analisar as redes *Long Short Term Memory* (LSTM), um tipo popular de RNR (HOCHREITER; SCHMIDHUBER, 1997). As RNR são muito competitivas em tarefas temporais, como por exemplo conversão de voz para texto (GRAVES; MOHAMED; HINTON, 2013), traduções (MIKOLOV et al., 2010) e legenda de imagens (KARPATHY; FEI-FEI, 2015) nas quais grandes quantidades de dados estão acessíveis para treinamento. Os resultados obtidos em outras áreas sugerem que as RNR podem ser utilizadas para modelar o conhecimento do estudante.

Figura 10: Uma RNR Desenrolada



Fonte: Traduzido de Xiong et al. (2016)

As RNRs tradicionais mapeiam uma sequência de entradas $x_1, x_2, ..., x_t$, para uma sequência de saídas $y_1, y_2, ..., y_t$. Estes dados de saída são obtidos calculando-se uma sequência de estados "ocultos" $h_1, h2, ..., h_t$, as quais podem ser vistos como codificações sucessivas de observações passadas sendo capazes de ser utilizadas para prever o futuro. A figura 10 demonstra como as variáveis se relacionam através de uma rede simples definida pelas equações:

$$h_t = \tanh(W_{hx}x_t + W_{hh}h_{t-1} + b_h), \tag{3.10}$$

$$y_t = \sigma(W_{yh}h_t + b_y), \tag{3.11}$$

onde tanto o $\tanh$ quanto a função sigmoide, $\sigma(\cdot)$, são aplicados elementarmente. Esse modelo é parametrizado por uma matriz de peso de entrada W_{hx}, uma matriz de peso recorrente W_{hh}, estado inicial h_0 e uma matriz de peso de leitura W_{yh}. Além disso, ainda possuem uma unidade latente b_h e de leitura b_y (PIECH et al., 2015).

Ainda, as redes LSTM Hochreiter e Schmidhuber (1997) são uma variante mais complexa de RNR, em diversas oportunidades, se mostram mais poderosas. Em LSTM, as unidades latentes guardam seus valores até que explicitamente sejam liberadas pela ação de um "*forget gate*". Dessa forma, enfrentam menores adversidades para reter informações, obtendo êxito em armazená-las por etapas de tempo muito mais longas, tornando-as mais fáceis de treinar. Além disso, as unidades ocultas são atualizadas usando interações multiplicativas e podem, assim, realizar transformações mais complicadas para o mesmo número de unidades latentes. As equações de atualização para um LSTM são significativamente mais complexas do que para uma RNR. A saída y_t é um vetor de comprimento igual ao número de problemas em que cada entrada representa a probabilidade prevista de que o estudante responderia a esse problema em particular corretamente.

3.5.1 Séries temporais de entrada e saída

Para melhor compreender as RNRs, se faz necessário analisar as séries temporais de entrada e saída, já que elas são as responsáveis por evidenciar as interações do estudante nos sistemas baseados na técnica de DKT. Assim, para treinar uma RNR ou LSTM é necessário converter as interações do estudante em uma sequência de valores de entrada de tamanho fixo x_t. Podemos realizar essa tarefa de duas maneiras, dependendo da natureza das interações (PIECH et al., 2015):

- Para conjuntos de dados pequenos, com um número M de exercícios únicos, define-se x_t como uma codificação simples de interação do estudante $h_t = q_t, a_t$, a qual representa a combinação de qual exercício foi respondido e se o mesmo foi respondido de forma correta ou incorreta, então $x_t \in \{0, 1\}^{2M}$, porém dessa forma com o q_t separado o desempenho não se mostra tão satisfatório.

- Já para uma grande quantidade de dados, utilizando a maneira anterior rapidamente seria impraticável devido ao tamanho muito grande de informações. Para esse conjunto de dados pode-se atribuir um vetor aleatório $n_{q,a} \sim N(0, \mathbf{I})$, sendo que a cada tupla de entrada $n_{q,a} \in \mathrm{Re}^N$, e $N \ll M$. Restou definido que cada valor de entrada x_t corresponde ao vetor $x_t = \mathbf{n}_{q_t,a_t}$, esta representação aleatória de baixa dimensionalidade de um vetor de alta dimensionalidade é motivada pela detecção comprimida. Tal detecção comprimida afirma que um sinal k-esparso em d dimensões pode ser recuperado através de projeções lineares aleatórias de $k \log d$ (até as constantes de escala e aditivas) (BARANIUK, 2007 apud PIECH et al., 2015). Assim, a previsão de a_{t+1} pode então ser lida a partir da entrada em y_t correspondente a q_{t+1}.

4 APRENDIZADO DE MÁQUINA

O aprendizado de máquina é baseado em redes neurais (RN), as quais objetivam copiar o funcionamento do cérebro humano para que as máquinas desenvolvam capacidade de executar tarefas automatizadas sem a interferência humana. As RNs são uma parte fundamental no desenvolvimento deste trabalho, já que o mesmo objetiva criar um modelo de estudante através delas, denominado de modelo *Deep Knowledge Tracing* (DKT).

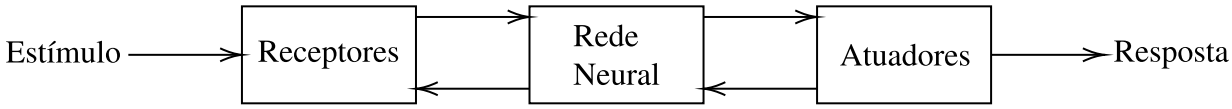
A primeira RN foi proposta por McCulloch e Pitts (1943), através de um artigo que versava sobre o funcionamento dos neurônios, resultando numa RN simples em que se utilizavam circuitos elétricos. A partir daí, vários outros trabalhos foram desenvolvidos, cada um deles contribuindo para a evolução das RNs, até que Fukushima (1975) propôs a primeira RN multicamadas.

Inicialmente, as RNs foram pensadas para a resolução de problemas diversos como se imaginava fosse o funcionamento do cérebro humano. No entanto, com o passar do tempo os cientistas constataram que mesmo o cérebro possui áreas responsáveis por processos específicos, como por exemplo o lobo frontal que tem como função a movimentação do corpo, o lobo parietal que lida com os sentimentos e assim por diante. Pelos conhecimentos adquiridos sobre o funcionamento tanto do cérebro como das RNs, os trabalhos com RNs foram sendo adaptados para as descobertas, chegando ao ponto em que elas receberam tarefas específicas, ganhando, assim, funções como visão computacional (SHINZATO, 2010; COUTO, 2012), processamento de linguagem natural (HU et al., 2014; SOCHER et al., 2011), dentre outros.

4.1 O neurônio

O sistema nervoso humano pode ser descrito como um sistema de três estágios, conforme demostrado no diagrama em blocos da Figura 11. O centro do diagrama é o cérebro, representado pela rede neural, que recebe de forma contínua as informações, tomando, ao fim, as decisões. Os dois conjuntos de setas que aparecem na Figura 11 representam impulsos, as que apontam para a esquerda indicam uma transmissão de informação para a *frente* através do sistema, enquanto as orientadas para a direita representam uma *realimentação* do sistema. Os receptores traduzem os estímulos externos ou do corpo em impulsos elétricos que são transmitidos ao cérebro. Já os atuadores convertem os impulsos elétricos oriundos do cérebro em respostas discerníveis como saída do sistema (HAYKIN, 2001).

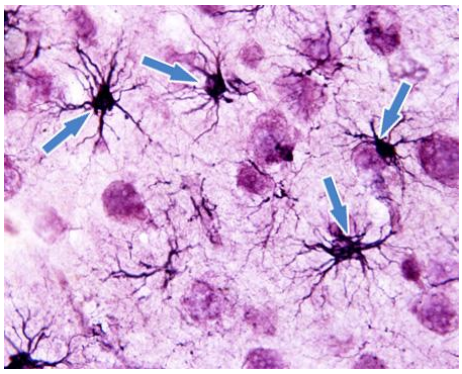
Figura 11: Representação em diagrama em blocos do sistema nervoso



Fonte: (ARBIB, 1987 apud HAYKIN, 2001)

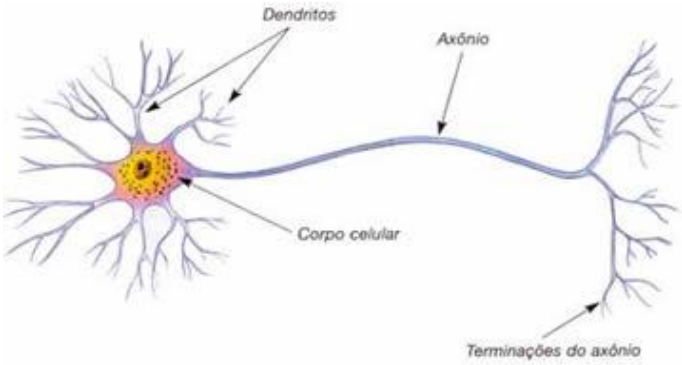
Os neurônios são estruturas biológicas que recebem uma entrada (estímulo), processa a entrada recebida e gera ou não uma saída. Caso gerada, a saída será a continuidade do estímulo, passando para o próximo neurônio. Na figura 12 pode-se observar um conjunto de neurônios em uma lâmina de microscópio; os mesmos foram impregnados com uma substância metálica, gerando assim a imagem. Cada um dos neurônios mostrados na Figura 12 é formado por três partes principais: os dendritos, o corpo celular e o axônio, como pode ser constatado na Figura 13. O **corpo celular** é a maior parte de um neurônio; nele estão contidos o núcleo e a maioria das estruturas citoplasmáticas; os **dendritos** são prolongamentos finos e geralmente ramificados, responsáveis pela condução dos estímulos captados do ambiente ou de outras células em direção ao corpo celular; já o **axônio** é um prolongamento fino, geralmente mais longo que os dendritos, apresentando, no seu final, terminações, cuja função é transmitir para outras células os impulsos oriundos do corpo celular.

Figura 12: Neurônios no microscópio



Fonte: Abrahamsohn e Freitas (2017)

Figura 13: Representação de um neurônio



Fonte: So Biologia (2008)

Quando os neurônios são interligados para formar o sistema nervoso, ligam seus axônios aos dendritos dos próximos neurônios. Essa ligação não necessariamente é física, mas simplesmente uma aproximação que permita a transmissão dos impulsos, se transformando numa espécie de cabo de transmissão de impulsos nervosos. A transmissão dos impulsos se dá entre

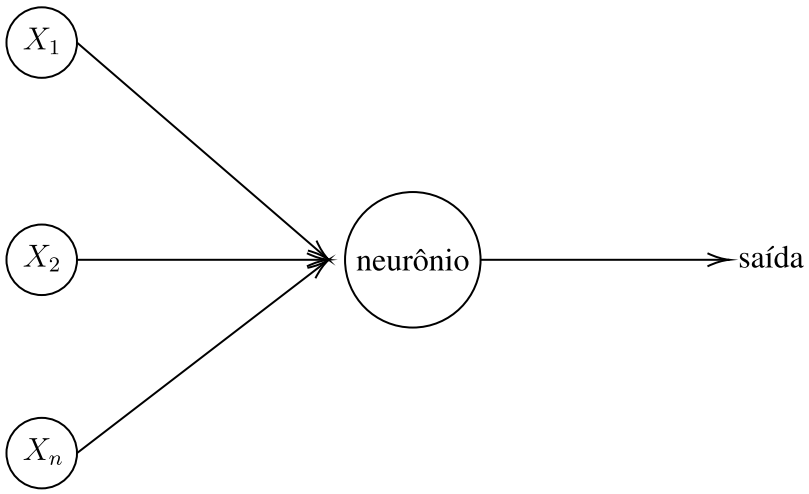
as extremidades dos neurônios vizinhos ao se encontrarem, ocorrendo a sinapse neural. Existem dois tipos de sinapse neural, as sinapses químicas e as sinapses elétricas (PURVES et al., 2005).

Na sinapse química o que é passado de um neurônio para o outro são substâncias químicas, as quais são passadas do axônio de um neurônio para o dendrito do próximo neurônio. Por outro lado, a sinapse elétrica utiliza impulsos elétricos para realizar essa comunicação.

4.1.1 Neurônio artificial

A seção 4.1 apresentou os neurônios do ponto de vista da neurociência. Esses se comparam aos neurônios artificias, já que ambos têm uma estrutura de entrada, recebendo a denominação de sinais de entrada nos neurônios artificiais. Além disso, possuem também uma estrutura de saída que nos neurônios artificiais é conhecida como sinais de saída. Essas estruturas, sinais de entrada e sinais de saída, são as responsáveis pela sinapse do neurônio artificial. Tais estruturas foram apontadas na Figura 14. Os sinais de entrada estão representados pelos círculos X_1 á X_n e os sinais de saída pela saída do neurônio. O neurônio mostrado na Figura 14 foi proposto por Rosenblatt (1957), consistindo em um sistema para reconhecimento de padrões, ficando conhecido como Perceptron de Rosenblatt.

Figura 14: Perceptron de Rosenblatt.



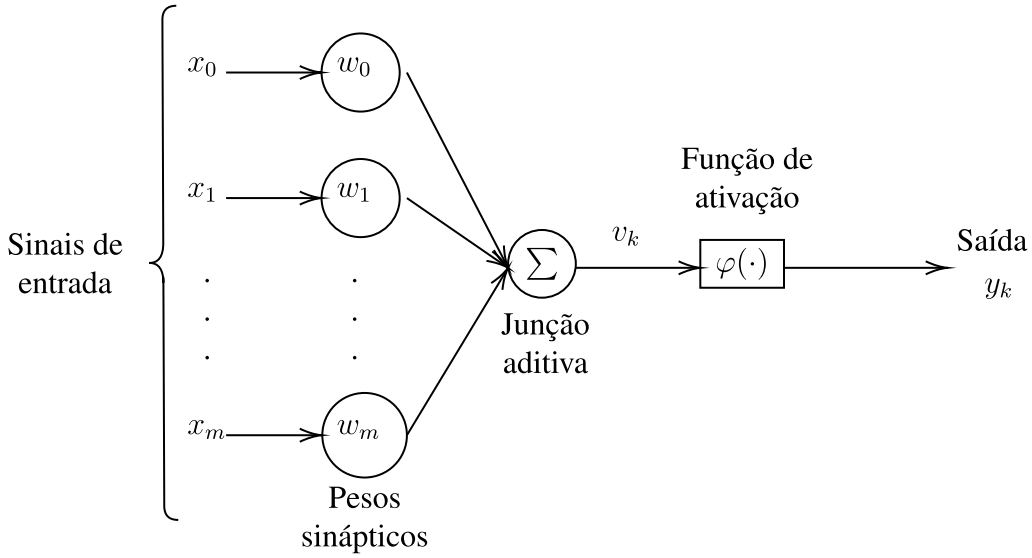
Fonte: Traduzido e adaptado de Rosenblatt (1957)

Alguns anos depois, Minsky e Papert (1969) inseriu ao neurônio o conceito dos pesos, os quais são um vetor de valores correspondendo às entradas (cada entrada é associada a um peso específico). No princípio do treinamento, os pesos são atribuídos de forma aleatória. Durante o processo de treinamento, esses pesos são ajustados para que a saída do neurônio seja a mais próxima possível da desejada. Para aplicação da regra desenvolvida por Minsky e Papert (1969), os valores das entradas são multiplicados pelos pesos associados. Na sequência, cada uma dessas entradas é entregue ao neurônio em si, no qual serão somados todos valores gerados, enviando, ao final, o resultado para a função de ativação, a qual possui a responsabilidade de

definir a saída do neurônio.

Na Figura 15 temos um exemplo de um neurônio aplicando a regra de Minsky e Papert (1969). O neurônio, chamado de Junção aditiva, recebe os valores das entradas (x_0, x_1 até x_m), os quais são multiplicados pelos pesos sinápticos (w_0, w_1 até w_m), onde cada uma das entradas é multiplicada pelo peso sináptico a ela associado, por exemplo $x_0 * w_0$, $x_1 * w_1$ e, assim, sucessivamente até a última entrada. Em seguida, é realizado o processamento (aqui representado pela junção aditiva juntamente com a Função de ativação), gerando uma saída (representada por y). Essa saída do neurônio resultará em um valor entre -1 e 1, sendo determinada pela soma ponderada $\sum_{i=1}^{m} w_i x_i$, que será menor ou maior do que um valor limiar (*threshold*). Em termos algébricos, podemos descrever na Equação 4.1.

Figura 15: Modelo não-linear de um neurônio



Fonte: Haykin (2001)

$$Output \begin{cases} 0 & if \sum_{i=1}^{m} w_i x_i \leq threshold \\ 1 & if \sum_{i=1}^{m} w_i x_i > threshold \end{cases} \tag{4.1}$$

4.1.1.1 Função de ativação

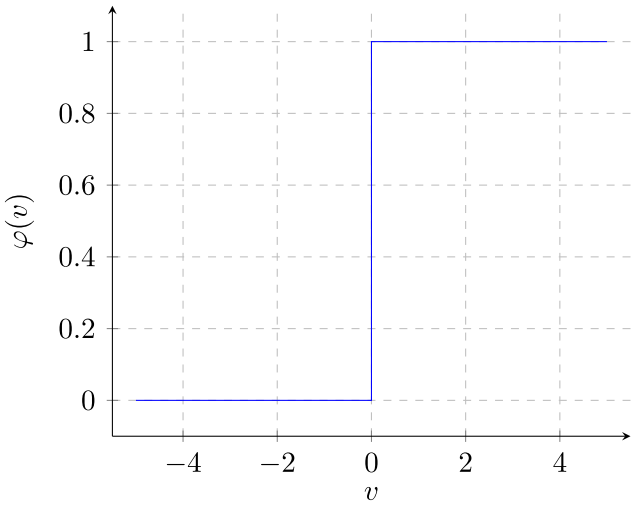
A função de ativação pode ser representada por $\varphi(v)$. Ela define como será a saída do neurônio levando em consideração o sinal v oriundo da Junção aditiva. Pode-se citar quatro tipos de função de ativação (HAYKIN, 2001).

1. **Função de limiar** - esta função é descrita pela Equação 4.2, conforme exibido na Figura 16, correspondendo a saída de um neurônio. A Função de limiar é comumente referida como *modelo de McCulloch-Pitts*, em reconhecimento ao trabalho (MCCULLOCH; PITTS, 1943). Para essa função, sempre que a saída do neurônio assume o valor 1 quer

dizer que o campo local deste neurônio é não-negativo, e 0 para o caso de ser negativo, o que descreve a *propriedade tudo-ou-nada* do modelo McCulloch-Pitts.

$$\varphi(v) = \begin{cases} 1 \text{ se } v \geq 0 \\ 0 \text{ se } v < 0 \end{cases} \tag{4.2}$$

Figura 16: Função de ativação limiar



Fonte: Adaptado de Haykin (2001)

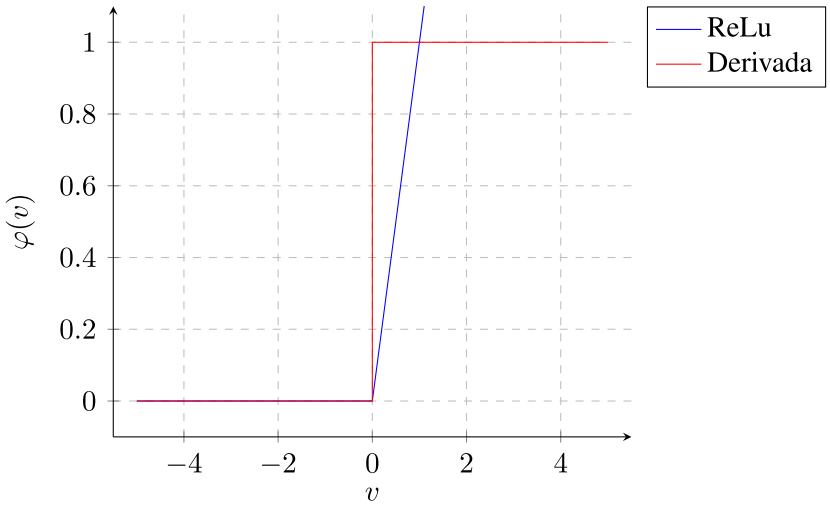
2. **Função unidade linear retificada (ReLu)** - é uma função amplamente utilizada e pode ser descrita pela Equação 4.3, e sua derivada é descrita pela Equação 4.4. Essa função simplifica consideravelmente a otimização da saída do neurônio, já que ela é muito parecida com a função identidade. A demonstração da saída dessa função de ativação pode ser visualizada na Figura 17, onde valores de v maiores que zero geram uma saída também maior que zero.

$$ReLu(x) = max0, x \tag{4.3}$$

$$ReLu'(x) = \begin{cases} 1, & \text{se } x \geq 0 \\ 0, & \text{se } x < 0 \end{cases} \tag{4.4}$$

3. **Função sigmoide** - também chamada de função logística, é dada pela Equação 4.5. Até pouco tempo, era a função mais utilizada em Redes Neurais Artificiais (RNA) por se demonstrarem biologicamente mais plausíveis. No entanto, ao verificar sua derivada na Equação 4.6, pode-se perceber que ela satura para valores maiores que 5 e menores que -5, assim tendendo a zero, e nessas regiões o gradiente desvanece. Na Figura 18 pode-se

Figura 17: Gráfico gerado pela função unidade linear retificada e sua derivada



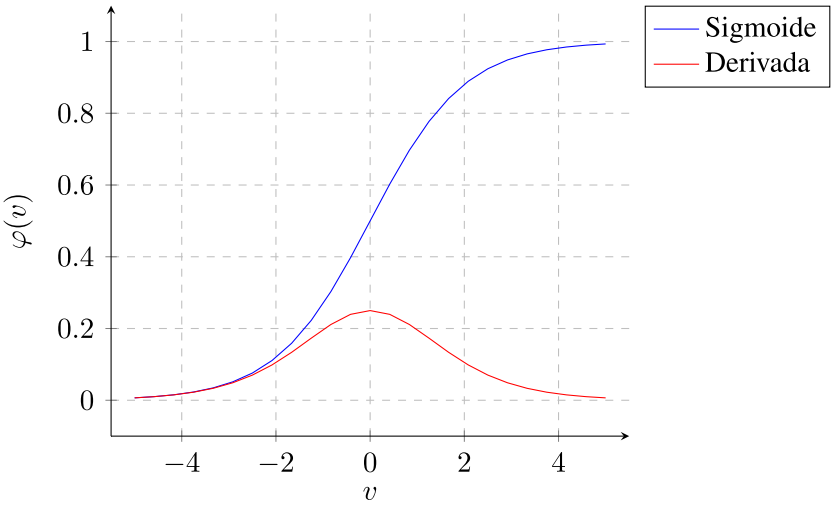
Fonte: Elaborado pelo autor

visualizar as curvas da função sigmoide e de sua derivada.

$$\sigma(x) = \frac{1}{1 + e^{x}} \tag{4.5}$$

$$\sigma^{'}(x) = \sigma(x)(1 - \sigma(x)) \tag{4.6}$$

Figura 18: Gráfico gerado pela função Sigmoide e sua derivada



Fonte: Elaborado pelo autor

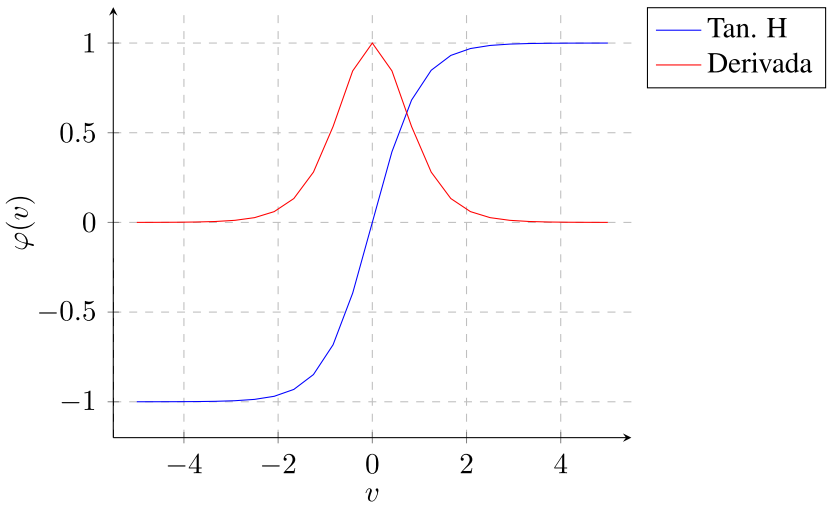
4. **Função tangente hiperbólica (TanH)** - similar à função sigmoide, essa também tem o formato de ”S”, mas ao contrário da sigmoide, ela varia de -1 a 1. A TanH se aproxima da função identidade, sendo assim uma alternativa à função sigmoide. A curva gerada pela TanH e sua derivada podem ser vistas na Figura 19. A Equação 4.7 define a função TanH,

enquanto a Equação 4.8 define a derivada da TanH.

$$\tanh(x) = 2\sigma(2x) - 1 \tag{4.7}$$

$$\tanh'(x) = 1 - \tanh^2(x) \tag{4.8}$$

Figura 19: Gráfico gerado pela função Tangente Hiperbólica e sua derivada



Fonte: Elaborado pelo autor

Outras funções poderiam ser citadas, mas a fim de exemplificação, as funções apontadas demonstram bem o que pode ser esperado das funções de ativação. Além disso, as funções vislumbradas serão as usadas no decorrer deste trabalho, como será explicitado no Capítulo 6.

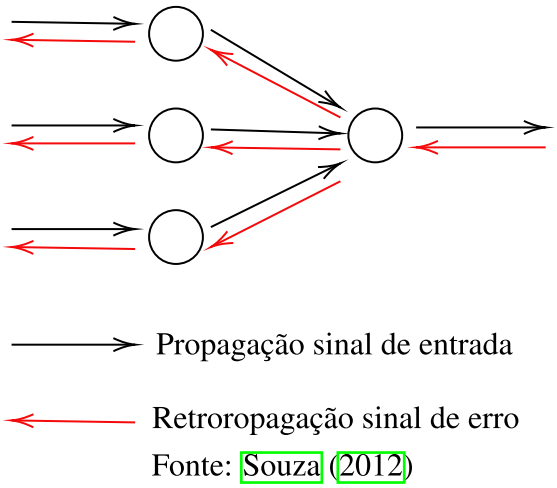
4.1.1.2 Backpropagation

O método *backpropagation* (retropropagação) foi proposto por Rumelhart, Hinton e Williams (1986), tendo por objetivo atualizar os pesos sinápticos atribuídos às entradas de cada neurônio da Rede Neural, visando minimizar a diferença entre a saída atual obtida pela RN e a saída desejada. Primeiramente, a RN é inicializada com um conjunto de pesos aleatórios. Durante o processo de aprendizagem, uma entrada é fornecida à rede e propagada até gerar uma saída. Em seguida, os valores da saída são comparados com os valores da saída desejada, resultando, assim, em um sinal de erro.

Dessa forma, o sinal de erro é retropropagado através da rede partindo da camada de saída, recalculando cada um dos pesos associados as entradas dos neurônios. Tal ocorrência pode ser visualizada na Figura 20.

A fórmula utilizada para realizar essa atualização dos pesos é a apresentada na Equação 4.9, onde o valor do peso W da próxima interação será o valor do peso da interação anterior subtraído os valores do parâmetro η, que representa a taxa de aprendizado da RN. Esse parâmetro

Figura 20: Orientação dos sinais de entrada e erro no *backpropagation*



Fonte: Souza (2012)

define o tamanho do passo dado para a correção do peso. Esse parâmetro ainda é multiplicado pelas derivadas parciais da função de erro E em relação a cada um dos pesos da rede W, culminando na geração da nova matriz de pesos atribuídas às entradas dos neurônios. Tais ocorrências irão ser repetidas por diversas vezes durante todo processo de treinamento da RN até a finalização deste processo.

$$W \leftarrow W - \eta \frac{\partial E}{\partial W} \tag{4.9}$$

4.2 Redes neurais

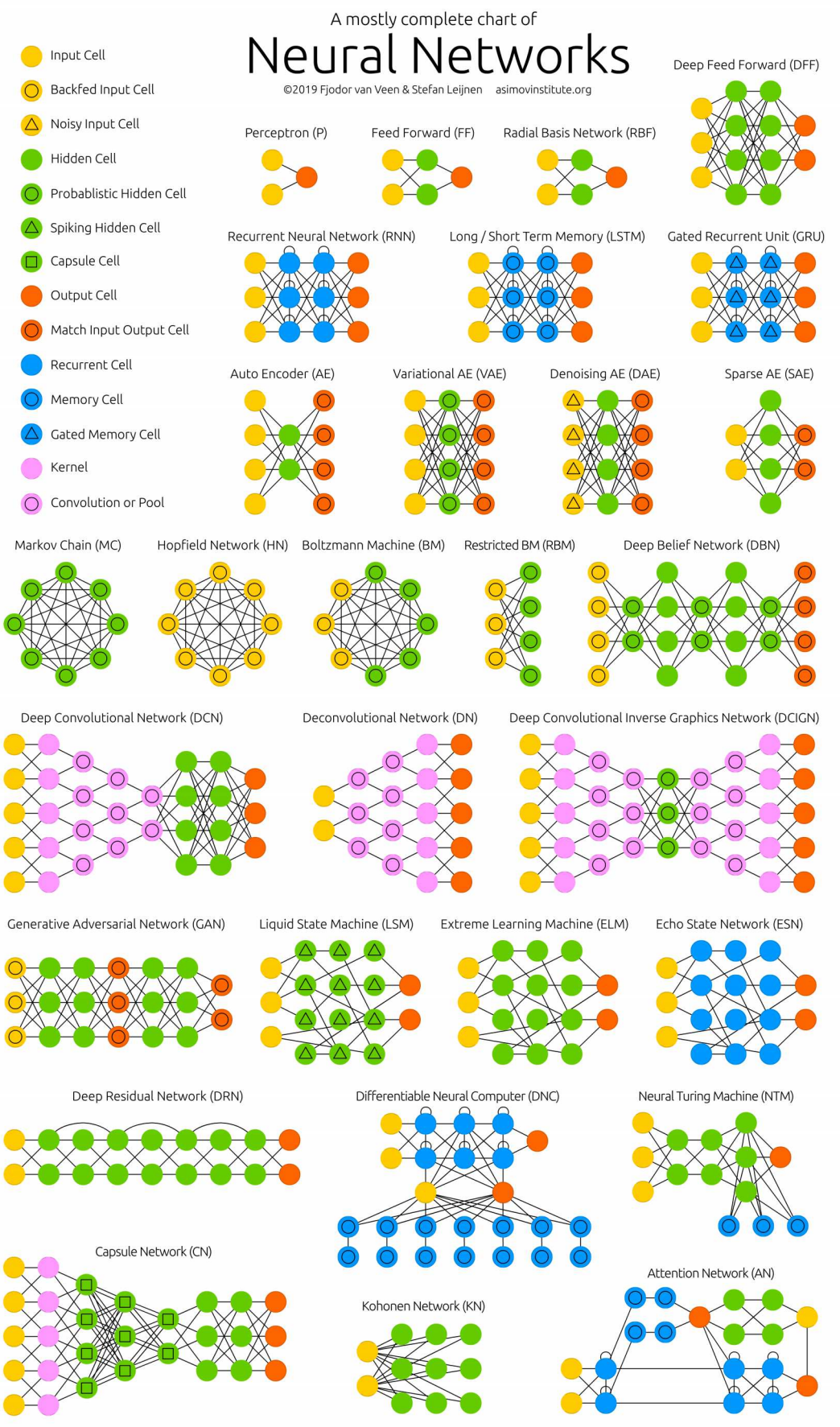
As redes neurais artificiais podem ser vistas como um grafo orientado, sem prejuízo ao que foi demostrado na Figura 15, visto que a exibição da rede neural como um grafo é somente uma simplificação do modelo gráfico visto na figura.

Os pesquisadores Leijnen e Veen (2020) reuniram em uma imagem a representação de diversas arquiteturas de redes neurais, como pode ser visto na Figura 21. Nela pode-se identificar desde a rede neural mais simples, chamada de perceptron, até as redes mais complexas, como as redes neurais recorrentes ou as redes neurais convolucionais.

Para fins deste trabalho serão mais aprofundados os seguintes tipos de redes neurais:

- Redes perceptron
- Redes multi-layer perceptron
- Redes neurais recorrentes
- Redes *long/short term memory*

Figura 21: Modelos de Redes Neurais



Fonte: Leijnen e Veen (2020)

4.2.1 Redes Perceptron

O perceptron é a forma mais simples de rede neural, utilizada para classificação de padrões linearmente separáveis. Geralmente, consiste em um único neurônio com pesos sinápticos ajustáveis. O conceito de rede neural foi inicialmente introduzido por MACCULLOCH e Pitts (1943), que inaugurou a ideia de redes neurais como máquinas computacionais, em seguida vários trabalhos surgiram, o primeiro a desenvolver uma rede neural (ROSENBLATT, 1957), a qual foi construída com apenas um neurônio.

A rede desenvolvida por Rosenblatt (1957) era capaz de realizar a tarefa de classificar informações de forma binária, buscando a identificação de objetos, por exemplo. Por possuir apenas um neurônio na sua construção, ele recebe informações de entrada, realiza as operações matemáticas necessárias, para ao fim, calcular uma saída para a rede neural.

Um exemplo do modelo Perceptron pode ser visualizado na Figura 15. Pode-se constatar que o neurônio recebe os valores de entrada $x_0, x_1, ...x_m$, esses valores são multiplicados pelos respectivos pesos associados a cada um dos neurônios $w_{k0}, w_{k1}, ...w_{km}$, resultando assim no produto escalar que será fornecido à junção aditiva. Em alguns casos, nesse ponto, pode existir uma constante chamada *bias* (viés) que em momento algum durante o treinamento sofre alterações.

Então, os dados são repassados à função aditiva ($\sum$) que realiza a soma de todo o produto escalar através da Equação $\sum_j w_j x_k$. Neste ponto pode ser adicionado o *bias* resultando na equação $\sum_j w_j x_k + b$, o que pode trazer alterações à forma como a rede é treinada. Na sequência, o resultado deste somatório é passado à função de ativação, a qual verifica o valor resultante, comparando-o com o *threshold* (limite). A função de ativação ainda analisa em que ponto deve ser mudada a classificação, conforme pode ser visto na Equação 4.1, para ser gerada, em seguida, uma saída.

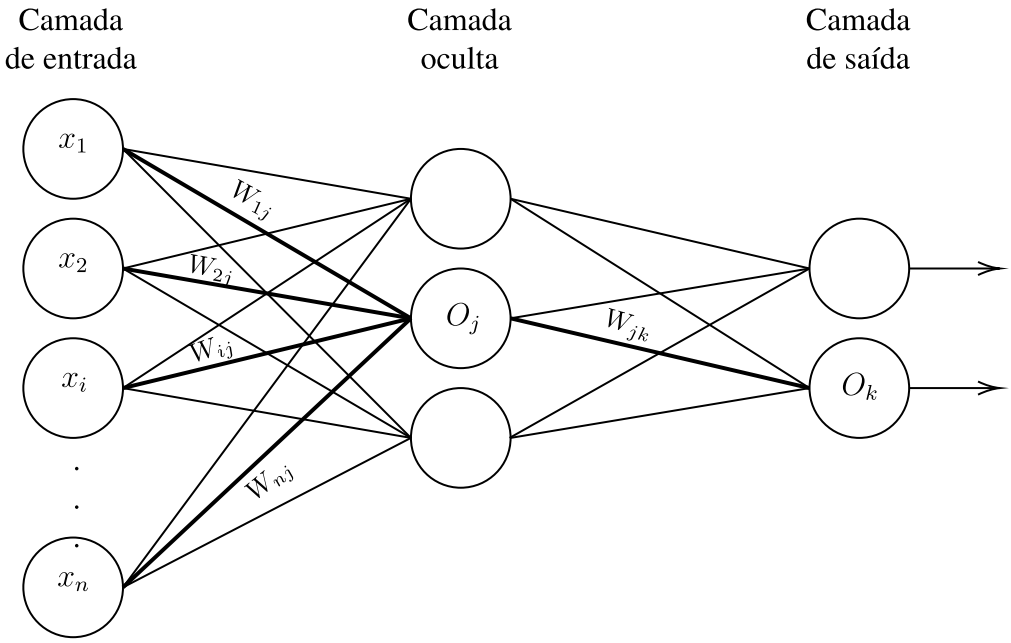
No início do processo de treinamento do modelo perceptron, valores aleatórios são atribuídos aos pesos w_i associados a cada uma das entradas. Após cada processamento durante o treinamento, esses pesos são atualizados, para realizar essa atualização é usado o algoritmo *backpropagation*.

4.2.2 Redes Multi-layer Perceptron

Essa arquitetura se baseia na Rede Perceptron, na qual os neurônios são organizados em camadas permitindo diversas arquiteturas. Sempre que se tratar de uma *Multi-layer Perceptron* está se falando de uma Rede Neural Artificial (RNA) que possui uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída Figura 22. Além disso, nenhuma das saídas de um neurônio de uma k vai estar ligada à entrada de um neurônio de uma camada menor ou igual a k. Ainda pode ser dito que esse tipo de RNA é completamente conectada, visto que cada neurônio fornece sua saída para todos os neurônios da próxima camada (HAN; PEI; KAMBER,

2011).

Figura 22: Arquitetura Rede MLP



Fonte: Han, Pei e Kamber (2011)

Ainda na Figura 22, pode-se constatar que cada um dos neurônios da camada de entrada $x_1, x_2, \cdots, x_i, \cdots, x_n$ estão ligados a todos os neurônios da camada subsequente. Pode ser visualizada ainda a camada oculta, onde tem-se um exemplo de um neurônio nominado de O_j, existindo um peso associado a cada uma dessas ligações chamados de W_{ij}. Este último está referenciado aos neurônios de onde sai a ligação, neste caso i, e ao neurônio onde essa ligação chega j. E assim sucessivamente para cada uma das ligações, inclusive naquelas que ligam a camada oculta a camada de saída, neste exemplo chamado de W_{jk}.

Em MLP uma das tarefas mais complicadas é descobrir a estrutura ideal para a rede, o que já foi fruto de diversos estudos, por exemplo (CYBENKO, 1989; HORNIK et al., 1989; FUNAHASHI, 1989). Em nenhum deles foi identificado qual a melhor formatação para a MLP, mas trouxeram alguns caminhos para orientar essa construção. Segundo os estudos mencionados, uma MLP com uma camada oculta é suficiente para aproximar qualquer função contínua, enquanto duas camadas ocultas são suficientes para aproximar qualquer função matemática. Porém, não houve consenso para se determinar a quantidade ideal de neurônios nas camadas ocultas para as MLPs. Então, Cybenko (1988, 1989); Haykin (2001) afirmam que essa quantidade deve ser definida de forma empírica.

Em alguns casos pode ser interessante a utilização de mais de duas camadas ocultas, pois assim pode ser facilitado o treinamento da RNA. No entanto, não é aconselhado visto que o erro que é propagado através da rede se torna menos útil e preciso (BRAGA; FERREIRA; LUDERMIR, 2007).

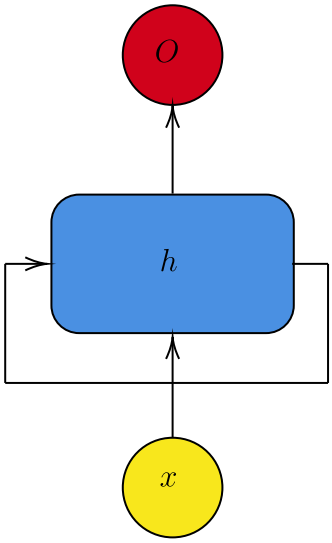
4.2.3 Redes Neurais Recorrentes

As Redes Neurais Recorrentes (RNR) são uma topologia de rede que apresentam neurônios recorrentes. Isso quer dizer que existe uma estrutura de *loop*, o qual permite a RNR armazenar a informação, além de permitir a interação desta com os novos dados de entrada. Essa capacidade de armazenamento torna essa topologia de rede ideal para atividades de processamento de séries temporais.

Uma série temporal, segundo (WOOLDRIDGE, 2016 apud CARVALHO et al., 2018), pode ser descrita como uma sequência de observações de uma variável ao longo do tempo, ou seja, uma série de informações coletadas em intervalos regulares em um período de tempo. Essas demonstram as características de um fenômeno em etapas sucessivas, e geralmente, em equidistantes períodos de tempo.

Quando se fala de RNR, é utilizada uma abstração onde um único nó representa uma camada de neurônios. Além disso, para ficar mais simples a visualização da rede, ela é representada na vertical, tendo suas camadas representadas de baixo para cima.

Figura 23: Representação RNR



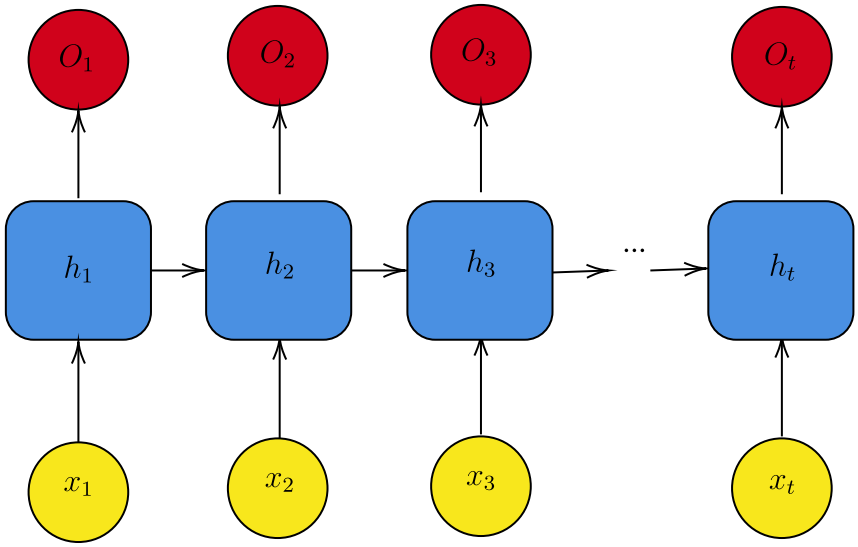
Fonte: Traduzido de Xiong et al. (2016)

Na Figura 23 está sendo ilustrada a forma de exibição de uma RNR. Nela é possível constatar a camada de entrada representada por x, assim como a camada oculta representada por h, e a camada de saída nomeada de O. Uma das características mais importantes das RNRs é o *loop* que liga os neurônios da camada oculta a eles mesmos, permitindo a este tipo de rede neural a persistência das informações.

A principal função das RNRs é o reconhecimento de padrões em sequências de dados. Esses algoritmos levam em consideração uma sequência, seja ela temporal ou não, possuindo, na sua grande maioria, um viés temporal. Isso quer dizer que a decisão tomada na etapa de tempo $t-1$ pode afetar a decisão que a rede tomar mais tarde no tempo t. Isso faz com que as RNRs

acabem tendo duas entradas, a atual e a do passado recente, que são combinadas para determinar a resposta para novos dados.

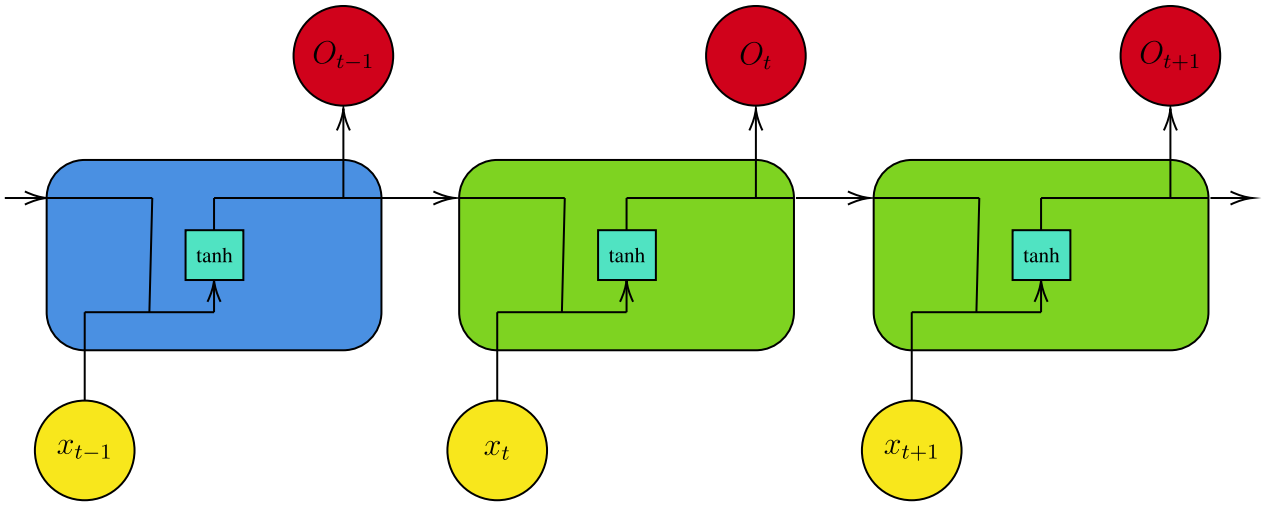
Figura 24: RNR desenrolada



Fonte: Traduzido de Xiong et al. (2016)

Visualizando mais internamente uma camada de neurônios, pode-se constatar que dentro de cada neurônio existe, assim como no perceptron, uma função de ativação que verifica se a informação é ou não passada adiante. Ser passada adiante significa que a informação da saída do neurônio irá gerar uma saída, bem como que será passada para o próximo neurônio da rede, conforme a Figura 25.

Figura 25: Visualização de uma sequência de neurônio nas RNR



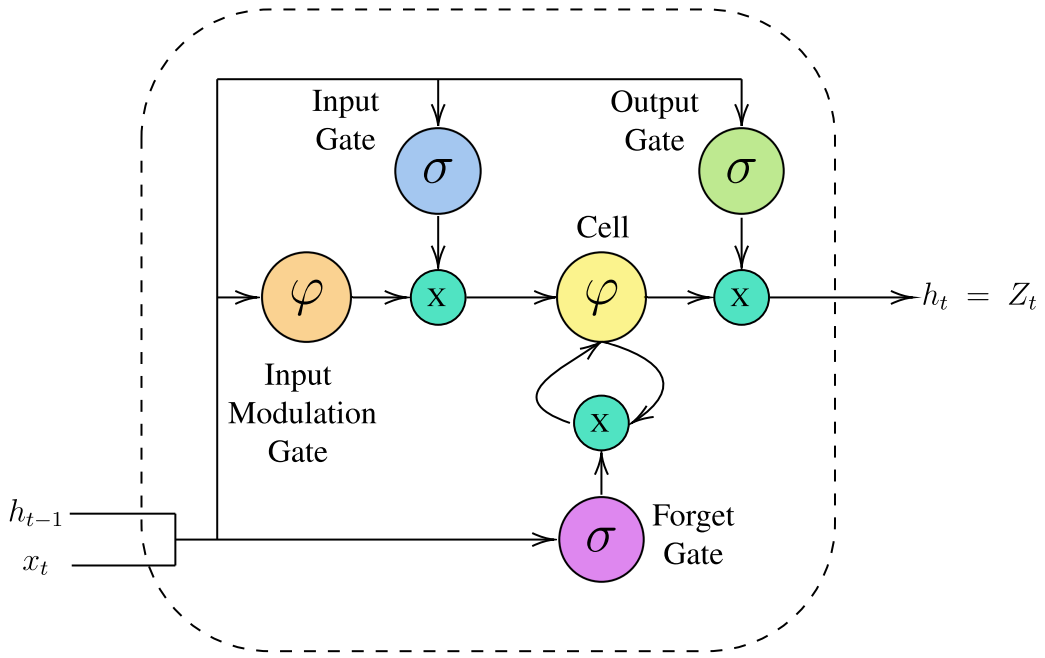
Fonte: Traduzido de Xiong et al. (2016)

4.2.4 Redes Long/Short Term Memory

As redes *Long/Short Term Memory* (LSTM) são um tipo especial de RNR, por possuírem a capacidade de armazenar informação por um tempo mais prolongado. A informação é armazenada até que um conjunto de condições seja alcançada, apagando após, a informação da célula.

Uma das principais características das LSTM em relação às RNRs convencionais é o fato de poderem armazenar a informação por muito mais tempo, o que ocorre por conta da estrutura de células desse tipo de rede. A LSTM possui uma estrutura em cadeia que se assemelha a quatro redes neurais e blocos de memória, chamados de células. Um exemplo de um neurônio pode ser visualizado na Figura 26.

Figura 26: Exemplo de Neurônio LSTM



Fonte: Kang (2017)

Toda a manipulação da memória dentro dos neurônios de uma rede LSTM é feita através dos portões (*gates*). Os três portões existentes são:

- **Forget Gate**: Neste portão, as informações que não são mais necessárias para a rede são removidas. A entrada x_t e a saída da célula anterior h_{t-1} são inseridas no neurônio, multiplicadas pelos pesos, a seguir o resultado é passado para a função de ativação (sigmoide), fornecendo uma saída binária. Se essa saída for "0", a informação armazenada na célula é esquecida; já se for "1" a informação ficará retida para uso no futuro.

- **Input Gate**: É o portão responsável por avaliar os dados que serão usados na célula do neurônio. Assim como no *Forget Gate*, os dados são regulados pela função sigmoide,

criando um vetor, além de utilizar adicionalmente a função tangente hiperbólica (TanH) que fornece uma saída de -1 a +1, armazenando todos os valores de h_{t-1} e x_t. Os valores deste vetor e os valores regulados são então multiplicados para se obter as informações úteis para a célula.

- **Output Gate**: É o portão responsável por apresentar as informações calculadas pela célula para gerar uma saída, assim como por passar os dados para o próximo neurônio.

A informação da memória de longo prazo está diretamente ligada à célula, na Figura 26 representada por *Cell*. As informações ficam armazenadas neste setor do neurônio, enquanto as setas (loop) abaixo da célula indicam a natureza recursiva da célula. Assim, a informação que ficar armazenada na célula pode ser excluída pelo *forget gate*, bem como pode ser criada/alterada pelo *input modulation gate* (portão de modulação de entrada).

4.2.5 Comparativo entre os tipos de redes neurais apresentadas

As redes neurais apresentadas se diferenciam na forma como tratam os dados, ao buscar a predição do resultado esperado. As redes Perceptron são as mais simples das redes neurais. No trabalho desenvolvido, essas não foram utilizadas, mas foram elas que permitiram o desenvolvimento de todas as demais RNs. Já as redes MLP são capazes de realizar a classificação de dados buscando prever uma classe específica.

As RNR trabalha com sequencias de dados que demonstrem o passar do tempo. Então, para empregar uma RNR, é necessário que os dados devem demonstrar um sequencia ordenada temporal. Elas permitem uma análise mais detalhada dentro de sequencias pré-definidas e sobre essas sequencias buscam prever a classificação da próxima interação. As RNRs possuem um componente que se assemelha muito a memória do cérebro humano, porém neste tipo de rede neural a informação que está armazenada nessa memória é atualizada em toda nova entrada de dados. Assim os dados da memória são lidos da memória, excluídos dessa memória, em seguida podem ser manipulados ou não e ao final são novamente armazenados na memória, e isso ocorre em cada interação da RNR. As LSTM, por sua vez, são um tipo específico de RNR, que se diferenciam pelo tempo que as informações ficam salvas nos neurônios. Nas redes LSTM essas informações ficam armazenadas por longos períodos de tempo, mais especificamente, até que uma condição seja alcançada, obrigando então a exclusão dessa.

5 TRABALHOS RELACIONADOS

5.1 Método de Seleção de Artigos

Primeiramente, foi realizada uma busca para identificar quais técnicas poderiam ser utilizadas para modelar o conhecimento do estudante. Dentre as técnicas disponíveis estão as Redes Bayesianas e sua variante Redes Bayesianas Dinâmicas apresentadas na Seção 3.3. Assim como a técnica *Bayesian Knowledge Tracing* (BKT), a qual realiza a inferência do conhecimento do estudante através de cálculos matemáticos de probabilidade, tal técnica foi explorada na Seção 3.4. Por fim, foi encontrada a técnica que se utiliza de *Deep Learning* para a construção do modelo de estudante, denominada de *Deep Knowledge Tracing* (DKT), que foi explicitada na Seção 3.5, sendo a técnica empregada nesse trabalho para a construção do modelo de estudante. Superada a escolha da técnica mais adequada para o desenvolvimento desse trabalho, foi dado início ao processo de pesquisa dos trabalhos relacionados à construção de modelo de estudante utilizando DKT.

Esse capítulo se destina a elucidar trabalhos pesquisados com relação ao assunto explorado por essa dissertação. Para realizar a seleção foi utilizado o Google Acadêmico[1] como forma de obter-se um panorama geral do tema. Refinando as buscas pelos trabalhos mais próximos aos conteúdos necessários, passou-se a consultar as principais bibliotecas disponíveis (ACM[2], Science Direct[3] e IEEE[4]). Além das bibliotecas, foram pesquisadas as conferências CSEDU, ITS, ICALT, SBIE, ITICSE, UMAP, CSEET, EDM, FIE, EC-TEL, IVA, LACLO, SIGCSE, visto que estas são as principais conferências relacionadas ao assunto deste trabalho. Por fim, os principais periódicos da área de inteligência artificial aplicada à educação, como os UMUAI, IJAIEd, Computer & Education, complementaram o rol de elementos que serviram para determinar o caminho para o desenvolvimento do presente estudo.

Para possibilitar as buscas relacionadas anteriormente, foi empregada a *string* de busca "*Deep Knowledge Tracing*", que retornou 395 resultados no Google Acadêmico. Nas demais bases de pesquisas, o retorno foi menor. Por exemplo, na IEEE foram retornados apenas seis trabalhos, na ScienceDirect o retorno foi de cinco trabalhos e na ACM foram retornados 30 registros. Realizando uma análise de duplicidade, se constatou que praticamente todos os títulos encontrados nas outras bases de pesquisas referenciadas, já estavam contemplados pela base do Google Acadêmico.

O total de trabalhos levantados totalizaram 436 artigos. Como primeira tarefa, analisou-se a duplicidade nos artigos, visto que os mesmos se originaram de fontes diferentes, para, então, começar-se a reunir os trabalhos pela lista do Google acadêmico, para, em seguida, agrupar os trabalhos das demais bases. Os trabalhos oriundos da IEEE estavam todos contidos na lista

[1]Disponível em: <https://scholar.google.com>
[2]Disponível em: <http://dl.acm.org/>
[3]Disponível em: <http://www.sciencedirect.com/>
[4]Disponível em: <http://ieeexplore.ieee.org/>

do Google, motivo pelo qual não foi acrescentado nenhum trabalho oriundo dessa conferência. Dos cinco trabalhos da ScienceDirect, dois já estavam presentes na lista do Google, enquanto da lista de 30 artigos obtida na ACM, dezenove trabalhos já se encontravam na lista do Google. Restaram, após concluir a etapa de mineração, 410 trabalhos na lista, na sequência, os trabalhos foram baixados, sendo constatado que 103 trabalhos não permitiam o download do arquivo, seja por não estarem mais disponíveis, seja por cobrarem taxas para o acesso.

Concluída a etapa anterior, foram reunidos 307 trabalhos para a próxima etapa de verificação, consistindo em averiguar a linguagem dos trabalhos, ficando constatado que 39 trabalhos dos listados estavam escritos em outra língua que não o inglês ou o português. Dessa forma, da lista inicial, apenas 268 trabalhos passaram a próxima etapa, que consistiu na leitura dos títulos dos trabalhos. Neste ponto foi constatado que 87 trabalhos não tinham nenhuma ligação com o tema abordado nessa pesquisa, motivo pelo qual foram descartados. Na continuidade, os 181 trabalhos restantes foram analisados para verificar se possuíam no mínimo quatro páginas, já que trabalhos com uma quantidade menor de páginas não costumam trazer conclusões relevantes, uma vez que, em sua grande maioria, são trabalhos em fase inicial de pesquisa, sem grandes informações relevantes. Isso resultou em 24 trabalhos sem a quantidade mínima de páginas, reduzindo para 157 trabalhos sobressalentes.

A próxima etapa consistiu na leitura do resumo dos trabalhos, na busca por verificar se os mesmos possuíam resultados que viessem a acrescentar no desenvolvimento desse trabalho. Nessa fase, 41 trabalhos foram excluídos devido à não existência de resultados, bem como foram eliminados outros 66 devido aos trabalhos não terem grande relevância para essa pesquisa, restando, assim, 50 trabalhos para a fase final deste levantamento. A fase final dessa análise consistiu na leitura completa dos trabalhos, verificando se os mesmos se encaixavam de alguma forma ao contexto procurado. Ao fim de toda mineração, foram selecionados seis trabalhos que se encaixavam ao fim almejado por essa pesquisa.

5.2 Trabalhos Selecionados

Findada a mineração de trabalhos a serem explorados para a elaboração dessa pesquisa, passou-se a explorar os trabalhos que possuíssem relação com a técnica escolhida para o desenvolvimento desse trabalho. A técnica utilizada para construção do modelo de estudante que se mostrou relevante para o estudo apresentado, foi a *Deep Knowledge Tracing* (DKT). Tal técnica pode ser considerada nova, tendo sido primeiramente apresentada por Piech et al. (2015), os quais apresentaram uma utilização de um tipo de RNRs chamado *Long Short Term Memory* (LSTM), que tinha por fim avaliar o conhecimento do estudante.

Para realizar a verificação proposta foram feitos diversos experimentos, primeiramente utilizando os dados oriundos dos STIs Khan Academy (Khan Math) e Assistments 2009-2010[5], adicionalmente, se valeram de dados simulados criados pelos próprios autores. Os dados reuni-

[5]Disponível em: <https://sites.google.com/site/assistmentsdata/home/assistment-2009-2010-data>

dos por (PIECH et al., 2015) foram testados no modelo de estudante utilizando BKT. A forma como os testes foram realizados com o BKT não foram aprofundados por (PIECH et al., 2015), sendo citado que foram utilizadas as fórmulas apresentadas nessa dissertação na Seção 3.4. Já nos testes com a técnica DKT, os autores criaram uma RNR utilizando LSTM; essa foi criada utilizando a linguagem de programação Lua[6]. O script criado por (PIECH et al., 2015) contém uma RNR utilizando a configuração formada por uma camada oculta contando com 200 neurônios, utilizando a função de ativação tangente hiperbólica (TanH). Para o agrupamento dos dados, os autores utilizaram 31 respostas ofertadas como entrada para a RNR, objetivando alcançar a previsão do próximo passo. O método DKT obteve uma melhora no desempenho quando comparado ao BKT, chegando a apresentar um patamar aproximado de 25% de melhoria. Os resultados podem ser visualizados na Tabela 1, na qual é apresentada uma visão geral, acompanhada dos resultados obtidos por cada um dos datasets nos modelos BKT e DKT.

Tabela 1: Resultados dos testes do trabalho de Piech et al. (2015)

Dataset	Visão geral			AUC	
	Estudantes	Exercícios	Respostas	BKT	DKT
Dados simulados	4.000	50	200.000	0,54	0,75
Khan Math	47.495	59	1.436.000	0,68	0.85
Assistments	15.931	124	526.000	0,67	0,86

Fonte: Traduzido de Piech et al. (2015).

Após a publicação do trabalho de (PIECH et al., 2015), Khajah, Lindsey e Mozer (2016) testaram o modelo criado, concluindo que se trata de uma técnica que traz ganhos na avaliação do conhecimento do estudante, porém não atingiram o patamar de eficiência apurada por (PIECH et al., 2015). De acordo com as conclusões apuradas por (KHAJAH; LINDSEY; MOZER, 2016), a motivação para essa diferença se deu em decorrência a uma metodologia de inferência que não seria a mais apropriada para a situação. Para a verificação dos resultados, foi implementada a mesma RNR utilizando LSTM desenvolvida no trabalho de (PIECH et al., 2015). Porém, para a implementação foi escolhida a linguagem Python[7], bem como a utilização das bibliotecas Tensorfow e Keras. Ainda, ficou constatado que nos dados do Assistments utilizados por (PIECH et al., 2015), havia muitas informações duplicadas, ocasionando uma possível especialização dos modelos aos dados utilizados. Para realizar novos testes, foram removidos os dados replicados da base do Assistments 2009-2010, além de testar-se com os dados do Assistements 2014-2015 e com os dados do KDD Cup 2010[8]. Os testes geraram os resultados apresentados na Tabela 2, na qual pode ser constatado que, de fato, houve um acréscimo do modelo DKT sobre os resultados obtidos com o modelo BKT. Fica, ainda, evidenciado que no trabalho não foi levado em consideração os exercícios respondidos pelos estudantes, mas apenas as habilidades envolvidas em cada exercício. Como cada exercício pode conter mais de uma

[6]Disponível em: <https://www.lua.org/>
[7]Disponível em: <https://www.python.org/>
[8]Disponível em:<http://pslcdatashop.web.cmu.edu/KDDCup/downloads.jsp>

habilidade envolvida, a quantidade de habilidades acabou sendo maior do que a quantidade total de exercícios.

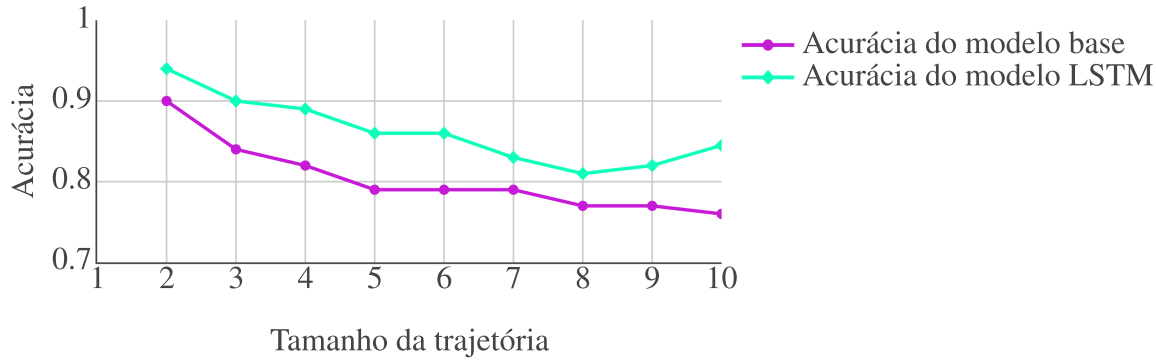
Tabela 2: Resultados dos testes do trabalho de Khajah, Lindsey e Mozer (2016)

Dataset	Visão geral			AUC	
	Estudantes	Habilidades	Respostas	BKT	DKT
Assistments 2009-2010	4.217	124	328,292	0,63	0,82
Assistments 2014-2015	19.457	100	707.944	0,64	0,70
KDD Cup 2010	574	436	607.026	0,62	0,79

Fonte: Traduzido e adaptado de Khajah, Lindsey e Mozer (2016).

Wang et al. (2017) propuseram um modelo de estudante criado para ser utilizado no intuito de inferir o nível do aprendizado do estudante em exercícios de programação do sistema WEB *Hour of Code*[9], avaliando o aprendizado recebido pelos estudantes. Para que o sistema fosse desenvolvido, uma sequência de respostas do exercício 18 da ferramenta foi avaliada pelos autores buscando verificar a assertividade no exercício 19. Dessa maneira, foi possível realizar a comparação do desempenho efetivo do estudante com o previsto pela ferramenta. Nos resultados preliminares apresentados na Figura 27 obtidos pelo trabalho, pode-se constatar uma melhoria sobre o modelo base que já vinha sendo utilizado no sistema.

Figura 27: Teste de acurácia do trabalho de Wang et al. (2017)



Fonte: Traduzido e adaptado de Wang et al. (2017)

Em Lin e Chi (2017) foi realizada uma comparação entre o modelo BKT, outro utilizando RNR simples e um terceiro usando LSTM. No modelo BKT foram implementadas variações entre o BKT original associando ao *Intervation-BKT* (iBKT), sendo testados seis modelos utilizando variações entre essas técnicas. Nos modelos RNRs, devido a sua alta flexibilidade, a qual permite entradas multivariadas não necessitando de quase nenhuma intervenção de especialistas humanos, foram realizados diversos experimentos a fim de encontrar a melhor configuração. Igualmente flexível, o modelo LSTM também foi testado em diversas configurações.

[9]Disponível em: <https://code.org>

Tanto as RNRs simples como aquelas que utilizam LSTM foram experimentadas através das configurações contando com uma ou duas camadas ocultas, contendo em cada um dos experimentos 50, 100, 150 ou 200 neurônios por camada, utilizando uma função de ativação TanH. Ademais, foi inserida uma camada de *dropout* na LSTM, a qual é responsável pela exclusão aleatória de uma parcela dos pesos para evitar o seu problema de gradiente, com valor de 20% entre as camadas da rede.

Os modelos foram desenvolvidos na linguagem Python, utilizando o framework Keras[10], experimentando todos os modelos de RNR por 100 épocas. Os dados utilizados para os experimentos foram coletados do STI de probabilidade Pyreness[11],usando as interações de 524 estudantes com o STI. Esses estudantes estavam na etapa final do curso de graduação em matemática discreta no departamento de ciência da computação da universidade da Carolina do Norte, entre os anos de 2014 e 2016. Os estudantes realizaram quatro etapas no experimento: pré-treinamento, pré-teste, treinamento e pós-teste. Todos estudantes foram avaliados com a mesma sequência de doze problemas durante o treinamento. Os pré-testes e pós-testes foram avaliados de forma duplo-cego por um único avaliador muito experiente. Analisando a Tabela 3 verifica-se que os modelos que utilizam LSTM possuem uma maior precisão e acurácia da predição, enquanto os modelos que utilizam RNR possuem pontuação melhor de F1-score. Além disso, ficou constatado que, utilizando cerca de 40% dos dados é possível alcançar um resultado bem próximo do resultado final do treinamento.

Tabela 3: Resultados dos testes realizados no trabalho de Lin e Chi (2017)

Modelo	Acurácia	Precisão	Revocação	F1 Score
BKT	0,642	0,648	0,821	0,724
RNR	0,667	0,658	**0,874**	**0,750**
LSTM	**0,670**	**0,670**	0,837	0,744

Fonte: Traduzido de Lin e Chi (2017).

Pu et al. (2020) buscaram o rastreamento do conhecimento de uma forma diferente do trabalho proposto por (PIECH et al., 2015). Para a realização dos experimentos, foram utilizadas as bases de dados do Assistments 2017[12], do KDD Cup 2010 e do Stat F2011[13], o qual é oriundo do curso de engenharia estatística da Open Learning Initiative (OLI). Pu et al. (2020) abordaram duas metodologias de transformação; a primeira delas foi a associação entre os problemas e as habilidades contidas nesses problemas. A segunda se tratou da inclusão do tempo corrente entre as respostas dadas pelos estudantes, a fim de identificar um possível esquecimento do conteúdo quando o estudante fica muito tempo sem interagir no STI. Na Tabela 4 pode-se constatar que ocorreu um ganho do modelo DKT sobre o modelo BKT. Os resultados apresentados na Tabela 4 explicitam os valores com a utilização das duas abordagens propostas.

[10]Disponível em: <https://keras.io/>
[11]Disponível em: <http://lin-res05.csc.ncsu.edu:8080/Gombaud/>
[12]Disponível em: <https://sites.google.com/view/assistmentsdatamining>
[13]Disponível em: <https://pslcdatashop.web.cmu.edu/DatasetInfo?datasetId=507>

Tabela 4: Resultados dos testes realizados no trabalho de Pu et al. (2020)

Dataset	Visão geral				AUC	
	Estudantes	Exercícios	Habilidades	Respostas	BKT	DKT
Assistments 2017	1.709	4.117	102	943.000	0,628	0,806
KDD 2010	3.287	1.379	899	4.420.000	0,744	0,784
Stat F2011	333	1.224	81	190.000	0,821	0,947

Fonte: Traduzido de Pu et al. (2020)

No trabalho de Sonkar et al. (2020) foi proposto um modelo de DKT baseado em perguntas denominado pelos autores de *question Deep Knowledge Tracing* (qDKT), o qual objetiva verificar o desempenho de todos os estudantes em uma determinada pergunta ao longo do tempo. Para testar o modelo proposto foram utilizadas as bases Assistments 2009 e 2017, Stat F2011, além de dados do Tutor, que se trata de uma plataforma de ensino on-line. Os resultados obtidos em cada uma das bases podem ser vistos na Tabela 5. Verificou-se que o modelo qDKT apresenta uma avaliação superior ao modelo DKT original em duas das quatro bases testadas.

Tabela 5: Resultados dos testes realizados no trabalho de Sonkar et al. (2020)

Dataset	Visão geral				AUC	
	Estudantes	Exercícios	Habilidades	Respostas	DKT	qDKT
Assistments 2009	4.151	16.891	111	325.637	0,740	0,678
Assistments 2017	1.709	1.183	86	249.105	0,721	0,742
Stat F2011	333	1.223	85	189.297	0,770	0,822
Tutor	895	5.981	1.592	437.524	0,856	0,875

Fonte: Traduzido de Sonkar et al. (2020)

Após comparação dos trabalhos apontados, ficou evidenciado que o principal diferencial do presente estudo, se dá pela utilização dos dados de um STI baseado em passos, já que os demais utilizam STIs baseados em respostas, conforme pode ser visualizado na Tabela 6. A utilização dos dados de STIs baseados em passos se diferencia do modelo baseado em respostas pelo primeiro apresentar uma gama maior de informações para o modelo de estudante. Isso porque, para resolver um exercício, em regra geral, o estudante fornece mais de um passo, enquanto nos baseados em resposta, somente a resposta final é dada ao sistema computacional, mesmo que o aluno empregue mentalmente vários passos. Ademais, visou-se utilizar os dados complementares oriundos do STI PAT2Math, como por exemplo, as unidades do conhecimento, que podem ser visualizadas na Tabela 7, as quais foram abordadas no passo corrente, acompanhada da informação do passo dado pelo estudante e do passo anterior por ele aplicado, almejando, com isso, uma avaliação mais precisa do conhecimento do estudante. Em um primeiro momento foi realizado um teste com a RNR LSTM proposta por (PIECH et al., 2015), porém, a mesma não alcançou os resultados esperados. Assim, realizou-se uma nova abordagem para a predição do próximo passo do estudante, a qual se vale da RNA utilizando MLP. Ressalta-se que essa abordagem não foi adotada em nenhum dos trabalhos relacionados encontrados, conforme evidencia

a Tabela 6.

Tabela 6: Comparativo entre os trabalhos relacionados

	Piech et al. (2015)	Khajah, Lindsey e Mozer (2016)	Wang et al. (2017)	Lin e Chi (2017)	Pu et al. (2020)	Sonkar et al. (2020)	Trabalho proposto
Tipo de STI usado	Baseado em respostas	Baseado em respostas	Baseado em respostas	Baseado em respostas	Baseado em respostas	Baseado em respostas	Baseado em passos
Bases utilizadas nos testes	Khan Math Assistments 2009-2010 Dados simulados	Assistments 2009-2010 Assistments 2014-2015 KDD Cup 2010	*Hour of Code*	Pyreness	Assistments 2017 KDD Cup 2010 Stat F2011	Assistments 2009-2010 Assistments 2017 Tutor	PAT2Math
Técnicas aplicadas	BKT DKT LSTM	BKT DKT LSTM	DKT LSTM	BKT DKT RNR DKT LSTM	BKT DKT LSTM	DKT LSTM qDKT	DKT LSTM DKT RNA
Features de entrada	Vários agrupamentos com três linhas: Uma contendo o ID do aluno Uma contendo uma sequência de problemas Uma contendo a sequência das respostas dos problemas	Vários agrupamentos com três linhas: Uma contendo o ID do aluno Uma contendo uma sequência de problemas Uma contendo a sequência das respostas dos problemas	Uma sequência de respostas de todos os estudantes para o problema analisado.	Vários agrupamentos com três linhas: Uma contendo o ID do aluno Uma contendo uma sequência de problemas Uma contendo a sequência das respostas dos problemas	Vários agrupamentos com três linhas: Uma contendo o ID do aluno Uma contendo uma sequência de problemas Uma contendo a sequência das respostas dos problemas	Duas abordagens foram adotadas. A primeira usou vários agrupamentos com três linhas: Uma contendo o ID do aluno Uma contendo uma sequência de problemas Uma contendo a sequência das respostas dos problemas A segunda usa uma sequência de respostas de todos os estudantes para cada um dos problemas contidos na base de dados.	Utilizada uma sequência de tuplas contendo o ID do estudante, o passo fornecido por ele, bem como o passo anterior, um contador da quantidade de passos utilizados pelo estudante desde o início de sua interação com o sistema, outro contador com os passos fornecidos até a conclusão do exercício em questão, além das unidades do conhecimento utilizadas no passo, além da correção do passo atual.
Configurações das RNs	Uma camada oculta com 200 neurônios utilizando a função de ativação TanH. E uma camada de saída com um neurônio usando a função de ativação sigmoide	Uma camada oculta com 100 neurônios utilizando a função de ativação TanH E uma camada de saída com um neurônio usando a função de ativação sigmoide	Uma camada oculta com 100 neurônios utilizando a função de ativação TanH E uma camada de saída com um neurônio usando a função de ativação sigmoide	Uma ou duas camadas ocultas com 50, 100, 150 ou 200 neurônios em cada camada utilizando a função de ativação TanH E uma camada de saída comum neurônio usando a função de ativação sigmoide	Foi utilizada a RNR LSTM utilizada no trabalho de Piech et al. (2015)	Uma camada oculta com 100 neurônios utilizando a função de ativação TanH E uma camada de saída com um neurônio usando a função de ativação sigmoide	Uma ou duas camadas ocultas com 100 ou 1000 neurônios em cada camada utilizando a função de ativação TanH ou ReLu E uma camada de saída com um neurônio usando a função de ativação sigmoide
Tipo de predição	Próxima resposta do estudante	Próxima resposta do estudante	Próxima resposta do estudante	Próxima resposta do estudante	Próxima resposta do estudante	Próxima resposta do estudante	Próximo passo do estudante
Métricas Utilizadas	AUC	AUC	Acurácia	Acurácia Precisão Revocação F1-Score	AUC	AUC	AUC F1-Score
Resultados	Comprovou que o DKT LSTM é alcançou melhor resultado quando comparado ao modelo BKT.	Comprovou que o DKT LSTM é superior ao modelo BKT, porém não alcançou os resultado obtidos anteriormente.	Se limitou a verificar a assertividade do modelo analisando apenas dois exercícios da base utilizada, deixando de compara-lo as demais técnicas.	Constatou que o modelo DKT LSTM possui uma maior acurácia, porém o modelo DKT RNR alcançou um melhor resultado de F1-Score.	Realizou as associações entre os problemas abordados e as habilidades associadas a cada problema, além da análise do tempo decorrido entre as respostas dadas pelo estudante para verificar o "esquecimento". Concluindo que, ainda que com alterações, o resultado do DKT LSTM foi superior ao melhor resultado obtido pelo modelo BKT disponível na literatura na época.	O trabalho buscou avaliar o andamento de diversos estudantes em um exercício, ao que foi denominado de qDKT, porém a melhoria almejada por essa técnica em relação ao DKT LSTM só foi obtida ao analisar uma das quatro bases utilizadas, não sendo presente nas demais	Concluiu que para a utilização em STIs baseados em passos o modelo DKT LSTM não é tão efetiva quanto o esperado. Já o modelo DKT RNA obteve um resultado consistente, quando utilizados os dados completos e pré-processados.

Fonte: Elaborada pelo autor

6 MÉTODOS

O trabalho desenvolvido objetivou a construção de um modelo para avaliar o conhecimento do estudante em sistemas tutores inteligentes baseados em passos. O sistema recebe como entrada os passos dados pelos estudantes (o passo anterior e o passo corrente), e busca prever se o estudante acertaria o próximo passo. Para esse trabalho serão utilizadas redes neurais para uma tarefa de classificação binária: classificar se o próximo passo é correto ou incorreto. Esse tipo de modelo é geralmente denominado de *Deep Knowledge Tracing*. No estudo foram avaliadas duas técnicas de aprendizado de máquina, a Rede Neural Recorrente (RNR), por se tratar do algoritmo utilizado pelos trabalhos relacionados, e o *Multi-layer Perceptron* (MLP), como uma alternativa que permitisse a utilização dos dados oriundos do STI baseado em passos.

Visando a concretização do objetivo proposto, foram utilizados os dados oriundos de um STI em funcionamento, o PAT2Math. O sistema PAT2Math, descrito na Seção 2.2, é um STI que assiste estudantes enquanto eles estão resolvendo equações passo a passo e vem sendo utilizados por escolas do Rio Grade do Sul desde 2014 e, por isso, possui uma quantidade expressiva de dados para o treinamento, a validação e a execução de testes dos modelos criados pelo estudo proposto. Os dados obtidos serão aprofundados na Seção 6.1.

6.1 Dados Utilizados

Os dados que serão utilizados para o treinamento e avaliação do modelo proposto foram extraídos do Sistema Gerenciador de Banco de dados (SGBD) MySQL, onde os dados do PAT2Math estão armazenados, no dia 06 de maio de 2019. Tais dados reúnem informações dos estudantes que usaram o sistema até aquele momento desde fevereiro de 2017. Ele inclui problemas resolvidos por estudantes, desde respostas dadas para cada passo, até os *feedbacks* retornados pelo tutor, incluindo os conhecimentos empregados pelos estudantes na resolução do problema.

Na base de dados apontada, constam 1018 estudantes cadastrados e 576 equações divididas nos níveis: Muito fácil, Básico, Fácil, Médio, Intermediário, Difícil, Avançado e *Expert*. Ainda, guarda as operações utilizadas na resolução dos passos fornecidos pelos estudantes, as quais estão descritas na Tabela 7.

Além das informações descritas, possui 163.305 passos de equações resolvidas pelos estudantes para as equações armazenadas no banco de dados. Desses passos, 119.282 são corretos e 44.023 errados.

De posse de tais dados, torna-se possível gerar o arquivo *Comma-separated values* (CSV) contendo os dados necessários para iniciar o processo de pré-processamento. Posteriormente, os dados resultantes são utilizados para execução dos testes no modelo de estudante utilizando o DKT. Tal arquivo possui:

a) a identificação do estudante, ocorrendo através do id de cadastro no PAT2Math;

Tabela 7: Operações disponíveis no STI PAT2Math

Sigla	Operação
AD	Adição
SB	Subtração
MT	Multiplicação
DV	Divisão
SP	Operações de simplificação
PA	Princípio aditivo
PM	Princípio multiplicativo
MM	Mínimo múltiplo comum
DM	Propriedade distributiva
AF	Adição e subtração de frações
MF	Multiplicação de frações
DF	Divisão de frações
OI	Operação inversa
UT	Unir termos
RE	Reescrever equações
ER	Erro
DE	Divide equações

Fonte: Seffrin (2015)

b) o passo anterior do estudante para o problema. Em se tratando do primeiro passo será apresentada a equação fornecida pelo sistema;

c) a resposta apontada pelo estudante para o passo em que se encontra;

d) informação de quais unidades do conhecimento foram utilizadas para solução do passo atual. Nesse momento, fica constatada cada uma das operações identificadas na Tabela 7;

e) um contador dos passos do estudante desde sua primeira interação com o sistema até a atual;

f) um contador dos passos do estudante que contabilize a quantidade de passos dado em cada solução de exercício proposto. Assim, esse contador será reiniciado sempre que o estudante iniciar a solução de um novo exercício;

g) a correção do passo em questão (certo ou errado);

h) a informação se o próximo passo será correto ou incorreto.

O exemplo de duas tuplas deste arquivo podem ser visualizadas na Tabela 8. Uma linha (tupla) representa um passo de uma equação resolvida por um estudante, enquanto cada operação que o estudante poderia aplicar para resolver o passo é representado como uma coluna no arquivo (por exemplo, AD, SB, MT, etc.). A utilização de uma operação pelo estudante é representada pelo valor 1, enquanto a sua não utilização é representada pelo valor 0. O estudante pode aplicar mais de uma operação na solução do passo, situação em que teremos mais

valores 1 retornados. Assim, um passo pode ser solucionado com o uso de apenas uma equação ou várias, motivo pelo qual poderemos visualizar apenas um valor 1 ou mais.

Tabela 8: Exemplo de tuplas do arquivo CSV

email	passoAnt	passo	countPasso	CountPassos	AD	SB
310	x+7=12	x=12-7	1	1	0	0
310	x=12-7	x=5	2	2	0	1

MT	SP	PA	PM	MM	DM	AF
0	0	1	0	0	0	0
0	0	0	0	0	0	0

MF	OI	UT	RE	ER	DE	correto
0	0	0	0	0	0	1
0	0	0	0	0	0	1

Fonte: Elaborado pelo autor

Além das colunas citadas na Tabela 8, adicionou-se outra contendo a informação objetivada pelo DKT, a qual visa identificar se o estudante acertaria ou não o seu próximo passo. Finalmente, após a obtenção dos dados, é possível a sua utilização nos modelos que serão objeto de estudo, desde que sejam feitas operações de pré-processamento nos dados.

6.1.1 Pré-processamento

A etapa de pré-processamento visou a adequação dos dados extraídos do sistema Pat2Math para possibilitar a sua aplicação nos modelos propostos. Sabendo-se que tais dados possuem também informações textuais (denominadas de categóricas) e de que os modelos a serem desenvolvidos trabalham com dados numéricos, houve a necessidade de realizar algumas alterações para viabilizar a utilização dos dados do Pat2Math nos modelos.

A primeira tentativa de adaptação dos dados se deu com a aplicação da função *label encoder* da biblioteca *scikit-learn*[1], a qual cria uma chave de identificação única para cada informação, como por exemplo, uma coluna contendo os passos de um estudante, conforme Tabela 9.

Na sequência da utilização da referida função se obtém a informação em um array contendo os dados a serem categorizados unicamente como:

```
1    array(["5x+3=7", "3x-4=2", "x+5=-12", "-8x+1=15", "x+4=9"])
```

[1]Disponível em: <https://scikit-learn.org>

Tabela 9: Exemplo de dados da coluna passo

1	5x+3=7
2	3x-4=2
3	x+5=-12
4	5x+3=7
5	x+5=-12
6	-8x+1=15
7	x+4=9

Fonte: Elaborado pelo autor

Associado a esse array os dados no dataset são alterados para o id correspondente a cada uma das informações textuais nele contidas, resultando, assim, nos dados da Tabela 10.

Tabela 10: Dados da coluna passo da Tabela 9 após utilização do *label encoder*

1	0
2	1
3	2
4	0
5	2
6	3
7	4

Fonte: Elaborado pelo autor

Segundo Géron (2019), a técnica realiza a associação dos dados categóricos a um número. Tal número será fornecido para o algoritmo de aprendizado de máquina, o que resulta numa possível situação problemática, já que ao analisar os dados de entrada os algoritmos podem entender, por exemplo, que uma tupla com número 3 é maior do que outra com número 1, o que nesse caso não retrata a verdade, somente representam informações diferentes. Ainda, pode trazer o dado equivocado de que 0 é mais semelhante a 1 do que a 3, não refletindo verdadeiramente os dados fornecidos. Assim, essa técnica acaba não se mostrando indicada para a utilização nessa espécie de dados.

Em alternativa a função *label encoder* foi empregada a *one-hot encoder* (OHE), igualmente da biblioteca *scikit-learn*. A função OHE primeiramente cria um array bem parecido com o utilizado na função *label encoder* e, em seguida, os dados desse array são utilizados para a criação de colunas no dataset, sendo atribuído o valor 0 para todas nesse momento. Após a criação dessas colunas, a função transforma os dados textuais em uma matriz, visualizada na Tabela 11. Na tabela exemplo podemos perceber que ao se deparar com uma informação única, a função a transforma em uma coluna.

Cada linha apresentada na tabela corresponde a um passo dado pelo estudante. Quando a função se depara com uma informação textual utilizada pelo estudante, ela buscará nas colunas criadas a qual coluna corresponde o passo apresentado. Após identificar a localização da equação nas colunas, substitui o valor 0 por 1, representando que a equação foi utilizada pelo

estudante nesse passo.

Tabela 11: Dados da coluna passo da Tabela 9 após utilização do *one-hot encoder*

	5x+3=7	3x-4=2	x+5=7	-8x+1=15	x+4=9
1	1	0	0	0	0
2	0	1	0	0	0
3	0	0	1	0	0
4	1	0	0	0	0
5	0	0	1	0	0
6	0	0	0	1	0
7	0	0	0	0	1

Fonte: Elaborado pelo autor

Assim, a rede neural não executa comparações com o valor contido na coluna, apenas atribuindo pesos a cada coluna como sendo uma entrada para a rede neural. Essa técnica é denominada de *one-hot encoding*, uma vez que apenas um atributo será igual a 1 (*hot*), enquanto os demais serão 0 (*cold*).

A técnica *one-hot encoding* se mostrou eficaz para uso dos dados do STI PAT2Math nas redes neurais, porém neste momento outro problema surgiu, a quantidade de colunas no dataset para entrada na rede neural chegou a 9.928, tornando a execução da rede neural muito complexa para rodar. Para possibilitar a utilização dos dados do STI PAT2Math, uma nova etapa foi adicionada ao processo, a qual visou a transformação das equações em seu template, assim, por exemplo, a equação $5x + 3 = 15$ se tornaria $Ax + B = B$.

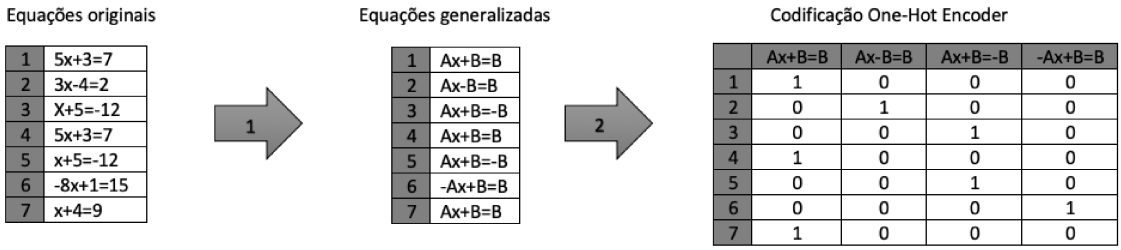
A inserção dessa etapa permitiu que a quantidade de dados na entrada da rede diminuísse, o que só foi possível através da substituição dos valores numéricos da equação por letras, ocorrendo, então, uma generalização das equações em seu template. Os valores que acompanhavam a variável x na expressão foram substituídos pela letra "A", já os valores independentes, aqueles que não se ligam à variável, foram substituídos pela letra "B".

Essa etapa foi realizada em todos os dados das duas colunas (passo atual e passo anterior), generalizando assim as equações contidas nessas colunas. A Figura 28 exemplifica esse processo. Na primeira tabela da Figura 28 encontram-se os dados originais, iguais à Tabela 9. Na segunda tabela, constam os dados após a generalização realizada. Por fim, na terceira tabela, constam os dados resultantes da codificação após utilização da técnica OHE. Assim, esse processo resultou no quantitativo de 5.293 colunas, as quais exigiram muito menos poder de processamento para o treinamento e teste dos modelos que serão propostos neste trabalho.

O gráfico exibido na Figura 29 ilustram alguns exemplos de passos que foram gerados após a etapa de generalização, ou seja, transformação da equação em template. O gráfico ilustra que 25,2% dos passos extraídos resultaram no template $Ax = +B$, outros 16,5% exibem o template $Ax = -B$, entre outros.

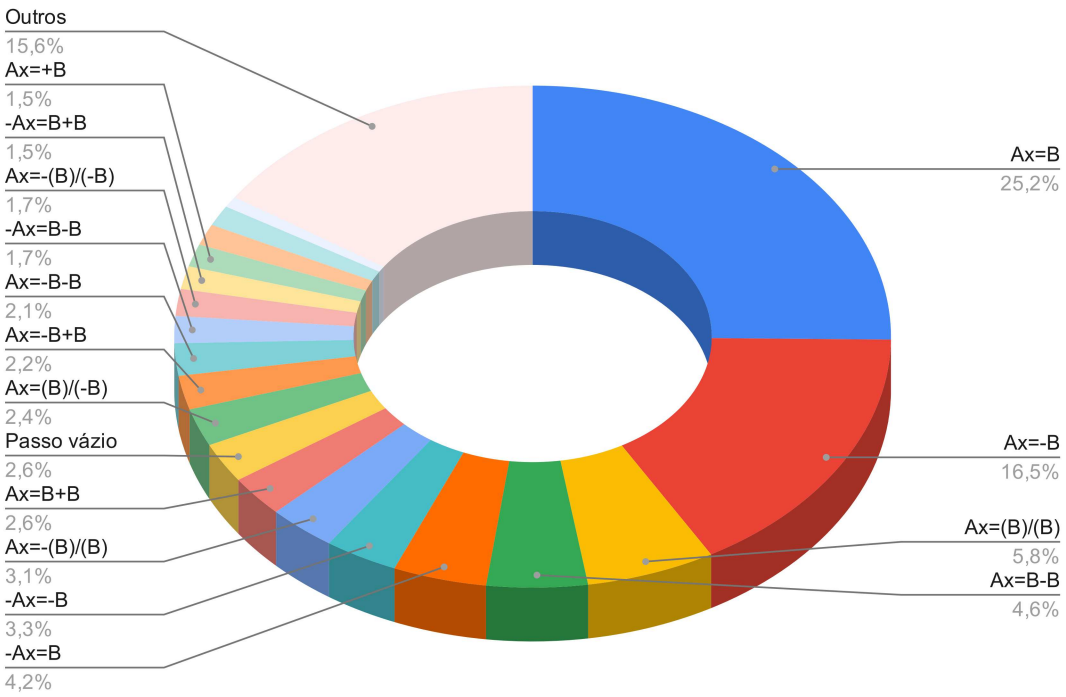
Concluída a fase de pré-processamento, os dados foram submetidos a uma análise junto

Figura 28: Exemplo dos dados da coluna passo da Tabela 9 após a generalização e codificação OHE



Fonte: Elaborado pelo autor

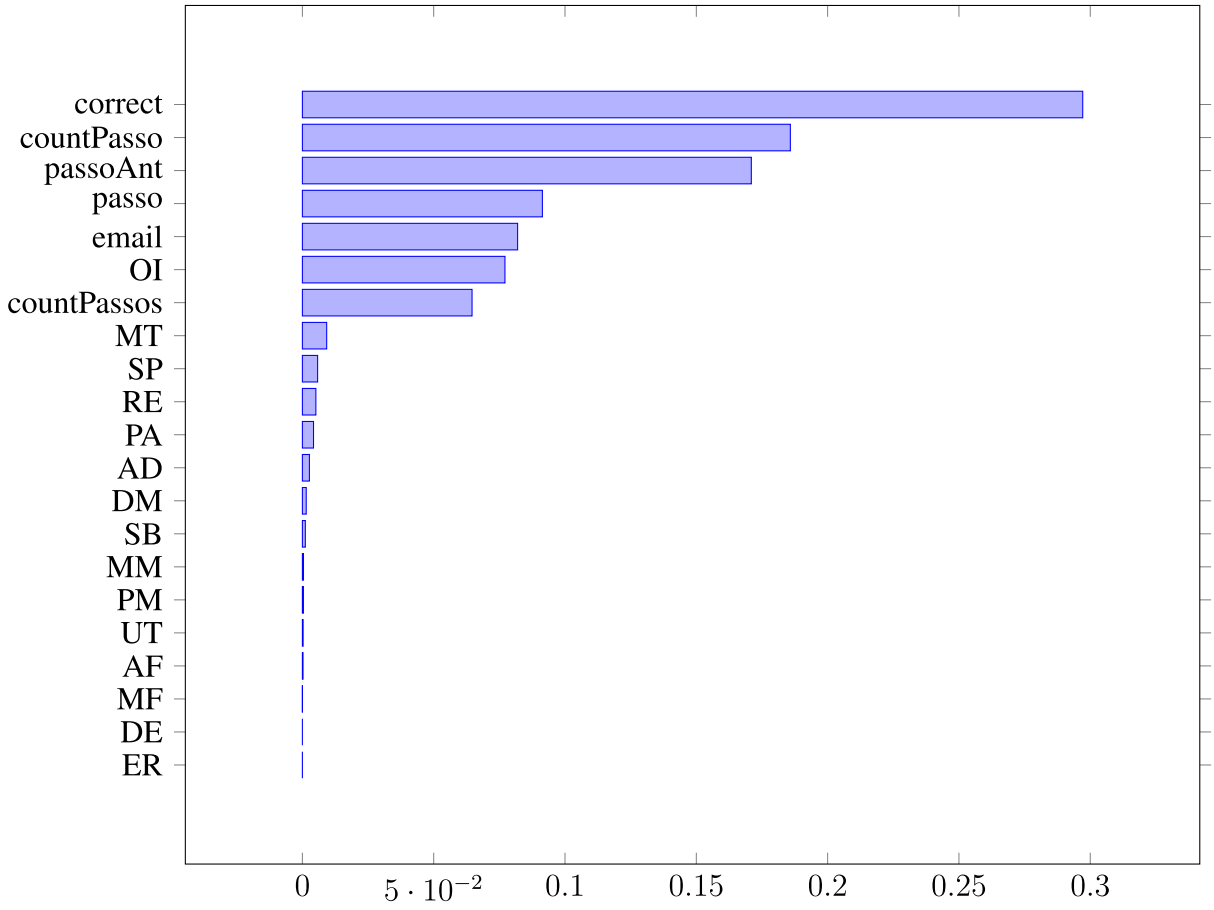
Figura 29: Gráfico de distribuição dos passos dados pelos estudantes após a generalização



Fonte: Elaborado pelo autor

ao sistema *AutoML Tables*[2] do Google Cloud Platform. Essa ferramenta executa uma análise dos dados para identificar quais colunas possuem maior influência sobre o resultado esperado, identificando se o estudante acertaria ou não o passo seguinte. Essa análise resultou no gráfico da Figura 30, onde pode-se constatar que a coluna **correct** equivale à resposta se o estudante acertou ou não o passo corrente, tendo uma influência de 29,71% no resultado esperado. Em segundo ficou a coluna **countPasso**, a qual traz a quantidade de passos que foram dados na solução do problema em questão, correspondendo a uma influência de 18,58%, e assim sucessivamente. Também pode-se ver que os dados das colunas referentes as operações algébricas aplicadas (MM, PM, UT, AF, MF, DE, ER) possuem uma influência muito baixa sobre o resultado, se aproximando muito de 0%

Figura 30: Gráfico demonstrativo da importância dos recursos



Segundo Ng (2017), até alguns anos atrás os dados eram divididos em duas partes de forma randômica, uma delas com 70% dos dados para treinamento e outra com 30% para teste. Atualmente, os dados podem ser divididos em três partes comumente definidas como:

- Conjunto de treinamento - Aquele que é utilizado para realizar o treinamento do algoritmo de aprendizagem.

[2]Disponível em: <https://cloud.google.com/automl>

- Conjunto de desenvolvimento - Aquele usado para ajustar os parâmetros, selecionar recursos e tomar decisões a respeito de qual algoritmo de aprendizagem escolher. Também pode ser chamado de conjunto de validação.

- Conjunto de teste - Usado para avaliar o desempenho do algoritmo escolhido, mas não para tomar decisões sobre qual algoritmo ou quais parâmetros de aprendizagem usar.

Sendo assim, se baseando em Ng (2017), para possibilitar o treinamento, a validação e os testes dos modelos desenvolvidos, os dados extraídos do PAT2Math foram pré-processados e divididos em três partes. Uma delas, contendo a maior quantidade de tuplas foi destinada ao treinamento, uma segunda parte foi destinada à validação do modelo criado, enquanto uma terceira parte foi reservada para a realização dos testes.

Visando a realização dos testes, do conjunto total de 1018 estudantes, 50 deles foram sorteados de forma aleatória. Em seguida, foram extraídos todos os passos fornecidos por esses estudantes durante a utilização do STI. Esses dados foram então adicionados a um dataset específico para os testes. Essa ação, conhecida como validação a nível do estudante (*validation at student-level*), se fez necessária para evitar que os modelos se especializem nos dados utilizados no treinamento. Os passos desses estudantes entraram no modelo como se fossem estudantes novos, permitindo, então, a testagem como se o sistema estivesse em produção. Esse conjunto de teste resultou em 11.753 tuplas, equivalendo a todos passos fornecidos pelos 50 estudantes sorteados anteriormente, correspondendo a uma média de 235 passos por estudante.

Assim, restaram 151.552 tuplas para os conjuntos de treinamento e de desenvolvimento, que foram divididos na proporção de 70% das tuplas para o conjunto de treinamento, enquanto o conjunto de desenvolvimento ficaria com os 30% restantes. Para que tal divisão fosse alcançada, foi utilizada a biblioteca *scikit-learn* para separar os dados entre esses dois conjuntos. Essa biblioteca possui uma função chamada **train_test_split**, que executa a divisão dos dados de forma randômica sem a necessidade de interação humana para ser executada. Sendo assim, tem-se os três conjuntos de dados salvos para posterior utilização nos algoritmos de aprendizagem que serão criados. O resultado final da quantidade de tuplas por conjunto está sendo mostrado na Tabela 12.

Tabela 12: Tabela com a quantidade de tuplas em cada conjunto de dados

Conjunto de dados	Contagem de duplas
Conjunto de treinamento	107.245
Conjunto de desenvolvimento	45.963
Conjunto de teste	10.097

Fonte: Elaborada pelo autor

6.1.2 Balanceamento dos dados

Um conjunto de dados é dito desbalanceado se as classes (por exemplo, acertou ou não acertou o passo corrente de resolução, da base do STI) não estiverem representadas de forma equilibrada, ou seja, entende-se que as tuplas devem apresentar quantificações mais aproximadas. O desbalanceamento na ordem de 100 para 1 é pertinente na detecção de fraudes e o desequilíbrio de até 100.000 para 1 foi relatado em outras aplicações (PROVOST; FAWCETT, 2001 apud CHAWLA et al., 2002).

Esse desbalanceamento foi constatado nos dados de treinamento, ou seja, o conjunto de treinamento continha mais dados classificados como acertou "1" do que errou "0". Mais especificamente, a base apresentava 78.785 classificados como certo e 28.460 classificados como incorretos. Esse desbalanceamento pode resultar em certos problemas para a rede, podendo se especializar mais na classificação de um valor em detrimento do outro.

Para solucionar esse problema, existem algumas técnicas que visam realizar o balanceamento dos dados, são elas: *Over-sampling* (sobre amostragem), *Under-sampling* (sub amostragem), bem como técnicas que combinam ambas. Para realizar os experimentos propostos, foi utilizada a biblioteca **imbalanced-learn**, desenvolvido por Lemaître, Nogueira e Aridas (2017), a qual se vale da linguagem Python, fornecendo uma série de métodos que realizam as transformações necessárias para cada uma das técnicas.

A biblioteca **imbalanced-learn** é uma *toolbox* de código aberto com diversos métodos voltados para a resolução do problema de dados desbalanceados, o que é frequentemente encontrado no aprendizado de máquina. Para seu funcionamento faz uso das seguintes bibliotecas **numpy**[3], **scipy**[4] e **scikit-learn**[5].

6.1.2.1 Técnicas de *Over-sampling*

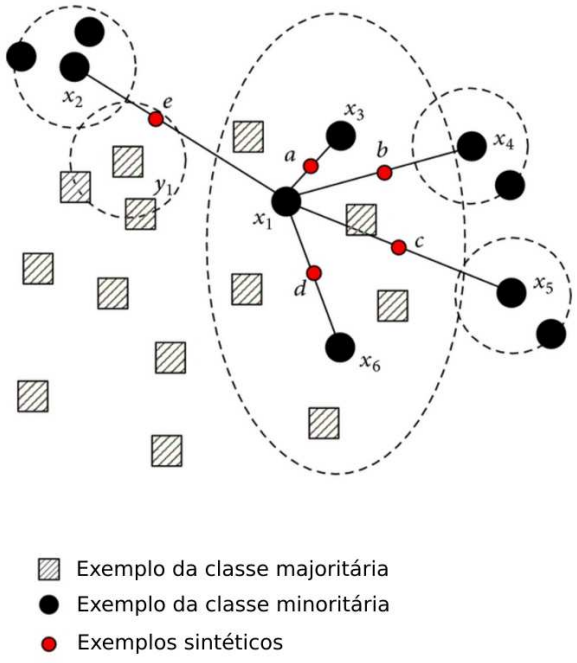
As técnicas de balanceamento chamadas de *over-sampling* tratam os dados da classe que possui uma quantidade menor de amostras, buscando a criação de dados para aproximar ou igualar as quantidades em cada uma das classes existentes no dataset. Alguns métodos são capazes de realizar o balanceamento dos dados utilizando a técnica do *over-sampling*, como por exemplo *Synthetic Minority Over-sampling Tecnique* (SMOTE) e *Adaptive Synthetic* (ADASYN).

O método SMOTE foi proposto por Chawla et al. (2002), buscando a criação de dados sintéticos (simulados) para a classe minoritária, baseando-se no trabalho de (HA; BUNKE, 1997 apud CHAWLA et al., 2002). (HA; BUNKE, 1997 apud CHAWLA et al., 2002) criou dados simulados devido à falta de dados para treinamento de uma rede neural em seu trabalho, o qual visava a identificação ou não de câncer, onde a simples rotação e inclinação das imagens já gerava os dados, ainda que utilizasse repetidamente as mesmas imagens.

[3]Disponível em: <https://numpy.org/>
[4]Disponível em: <https://www.scipy.org/>
[5]Disponível em: <https://scikit-learn.org/>

Já no método SMOTE, os dados da classe minoritária são criados através da coleta de amostras e inserção dos dados sintéticos ao longo dos segmentos de linha que ligam qualquer ou todas as k classes minoritárias dos vizinhos mais próximos. Essas amostras são geradas considerando a diferença entre o vetor de característica (amostra) e seu vizinho mais próximo; multiplica-se essa diferença por um número aleatório entre 0 e 1 e adiciona-o ao vetor de recursos. Isso faz com que a seleção seja um ponto aleatório ao longo do segmento de linha entre dois recursos específicos. Então, os dados são criados dentro desses segmentos. Tal funcionamento do SMOTE pode ser visualizado na figura 31. Após a execução do método SMOTE, os dados ficaram da seguinte forma, amostras negativas, que significa que o estudante erraria a próxima resposta, contendo 78.785 registros e amostras positivas, significando que o estudante acertaria a próxima resposta, reunindo 78.785 registros. Assim, concluída a execução, a quantidade de tuplas com amostras positivas ficou com a mesma quantidade de tuplas com amostras negativas, tornando os dados balanceados.

Figura 31: Exemplo do funcionamento do SMOTE



Fonte: Hu e Li (2013)

O método ADASYN foi proposto por He et al. (2008) e também trabalha na criação de dados sintéticos, sendo, porém, criadas algumas regras que permitiram um ganho sobre o método SMOTE em alguns casos. A ideia do método ADASYN tem por base a utilização da densidade $\hat{r}_i$ como critério para decidir automaticamente o número i de amostras sintéticas ao invés de utilizar um número fixo. Mais especificamente, $\hat{r}$ é uma medida da distribuição de pesos para diferentes exemplos de classes minoritárias, levando em consideração o seu nível de dificuldade na aprendizagem. Ao fim da execução do método ADASYN, as quantidades de dados em cada uma das classes contabilizaram 77.742 amostras negativas e 78.785 amostras positivas.

Nos dados do trabalho em desenvolvimento, a técnica que obteve uma melhor avaliação foi a SMOTE, visto que a mesma conseguiu um treinamento melhor da rede neural. Esses resultados serão apresentados no Capítulo 7.

6.1.2.2 Técnicas de *Under-sampling*

As técnicas de balanceamento dos dados de treinamento chamada *under-sampling* atuam sobre os dados das classes mais abundantes, buscando diminuir a quantidade de dados nas classes mais abundantes para igualar ou equilibrar com as classes com menor quantidade de dados.

Diversos métodos podem ser utilizados e praticamente todos se utilizam da técnica *k-Nearest Neighbors* (KNN), podendo ser traduzida como K vizinhos mais próximos. Assim, os métodos envolvem o agrupamento e a exclusão dos dados que podem estar repetidos ou muito próximos com a técnica do KNN. A biblioteca **imbalanced-learn** fornece diversas variações da técnica KNN, trazendo vantagem para alguns tipos de dados Lemaître, Nogueira e Aridas (2017). Dentre as técnicas está o método *NearMiss*, o qual se baseia no artigo Mani e Zhang (2003) que versa sobre a utilização de KNN no balanceamento de dados em um trabalho em que o objetivo era a nomeação de proteínas. Após a execução do método *NearMiss*, o sistema resultou em 28.460 registros de amostras positivas e 28.460 registros de amostras negativas.

Outro método disponível é *AllKNN*, o qual se baseia no artigo Tomek et al. (1976), no entanto, ao contrário dos demais que utilizam uma quantidade específica de vizinhos, esse algoritmo aumenta o número de vizinhos a cada nova interação. Com o emprego desse método, foram obtidos os seguintes resultados: 28.460 registros negativos e 37.990 registros positivos.

Na aplicação da técnica *under-sampling*, o algoritmo que obteve o melhor resultado foi o *AllKNN* trazendo um ganho sobre os demais métodos dessa técnica. Os resultados destes testes serão mais bem demonstrados no Capítulo 7.

6.1.2.3 Técnicas combinadas

O uso de técnicas combinadas utiliza as técnicas de *over-sampling* e *under-sampling* associadas, buscando, assim, aumentar a quantidade de tuplas da classe que possui menor quantidade de registros e diminuir a quantidade da classe que possui uma maior contagem de registros. Essa técnica tem por fim equilibrar a quantidade de dados e consequentemente tentar melhorar a classificação da rede.

A biblioteca **imbalanced-learn** fornece duas abordagens para trabalhar com a combinação das técnicas anteriores. A primeira delas é chamada de *SMOTETomek*, se baseando no artigo de (BATISTA; PRATI; MONARD, 2004). Essa abordagem utiliza o método SMOTE para realizar o *over-sampling* e o método de Tomek para fazer o *under-sampling*. Ao fim da execução restaram 76.470 registros negativos e 76.470 registros positivos. Por outro lado, a abordagem cha-

mada *SMOTEENN*, baseia-se no artigo (BATISTA; BAZZAN; MONARD, 2003), utilizando-se, igualmente, do método SMOTE para realizar o *over-sampling* e o método AllKNN para realizar o *under-sampling*. Após a execução do método *SMOTEENN* obteve-se 60.248 registros negativos e 36.722 registros positivos.

Percebe-se que o método *SMOTEENN* obteve o melhor resultado quando utilizado nos dados do STI PAT2Math. Os resultados apurados serão demonstrados no Capítulo 7.

6.2 Construção dos modelos

Ao encerrar o trabalho com o pré-processamento dos dados, avança-se à etapa da construção dos modelos propriamente ditos. Dois modelos foram adotados, sendo o primeiro deles utilizando a técnica de Rede Neural Recorrente (RNR) que insere alguns conceitos como memória, recorrência, dentre outros, enquanto o segundo vale-se da técnica *Multi-layer Perceptron* (MLP), a qual fornece o poder do *Deep Learn*, sendo mais simples de implementar, treinar e testar.

Para a construção dos modelos de estudante foram utilizadas as bibliotecas *Tensorflow*[6] e *keras*[7], permitindo uma facilitação na construção dos mesmos. Esses modelos serão mostrados a seguir.

6.2.1 Modelo de estudante usando RNR

O modelo utilizando Rede Neural Recorrente (RNR) utiliza um tipo específico de rede neural chamada de *Long-Short Term Memory* (LSTM), a qual possui uma capacidade maior de armazenar informações e essas somente são apagadas através de comando explícito. Para a criação deste modelo de estudante foram utilizadas as bibliotecas *Tensoflow* na versão 2.1 e *keras* que se encontra na versão 2.2.4. Neste modelo também serão utilizados os dados oriundos do STI PAT2Math, em um primeiro momento de forma simplificada, para testes iniciais, como será explicado a seguir, sendo após aplicados os dados oriundos do pré-processamento explorados na seção 6.1.

Buscou-se identificar a possibilidade de utilização desse tipo de Rede Neural (RN) com os dados oriundos do STI PAT2Math, visto que os trabalhos desenvolvidos até então utilizaram a técnica LSTM, mas em um formato diferente do extraído do STI. Os dados utilizados nos trabalhos relacionados são várias sequências com três linhas cada, na primeira linha consta o ID do estudante, a segunda conta com uma sequência de IDs dos problemas respondidos pelo estudante, enquanto a terceira linha apresenta a sequência das respostas se o estudante acertou ou errou cada um dos exercício da segunda linha, como no exemplo mostrado no trecho do arquivo CSV exibido na Tabela 13.

[6]Disponível em: <https://www.tensorflow.org>
[7]Disponível em: <https://keras.io>

Tabela 13: Trecho do CSV gerado para RNR dos trabalhos relacionados

513
3,5,9,12,56,37,37,42,25
1,1,1,1,0,0,1,1,0
Fonte: Elaborado pelo autor

Então, dentro do código é feita uma formatação para que um certo número de dados seja utilizado como entrada, por exemplo, as últimas 10 respostas do estudante, buscando identificar se no próximo exercício o estudante acertaria ou erraria a questão. Dessa forma, deixa-se de realizar uma análise detalhada sobre a interação do estudante com o STI, como por exemplo, qual foi a resposta dada, quais os conceitos aplicados, etc. Com os dados do STI PAT2Math é possível fornecer mais informações à RN, pois ele é um STI baseado em passos e, por isso, registra o desenvolvimento da solução do problema do estudante (passos), se eles estão corretos ou não, unidades de conhecimento aplicadas.

Os trabalhos relacionados envolveram dados de STIs baseados em respostas, ou seja, em que apenas a resposta final do estudante era corrigida. Na tentativa de utilizar os dados do STI PAT2Math, foi feita uma extração simplificada dos dados contendo apenas as informações do exemplo do CSV exibido na Tabela 13. Foram agrupadas as correções do STI em sequências de 31 passos, buscando prever a 32^a. Essa quantidade de passos (ou respostas) foi escolhida, seguindo os trabalhos relacionados, que a definiram empiricamente (PIECH et al., 2015; KHAJAH; LINDSEY; MOZER, 2016; WANG et al., 2017; SONKAR et al., 2020). Um exemplo dessa formatação final a ser fornecida à RNR, pode ser visualizado na Tabela 14, na qual os primeiros trinta e um valores apresentados serão fornecidos como entrada da RNN, enquanto o trigésimo segundo será o valor que a RNR deverá prever. A cada nova linha da entrada os dados são deslocados em uma posição, o que significa que o dado fornecido ao segundo neurônio na interação anterior agora será fornecido ao primeiro, e assim consecutivamente. Dessa forma, o dado que era o objetivo na linha anterior passa a ser fornecido como entrada para o trigésimo primeiro neurônio de entrada. Tal formatação foi nomeada de **Formatação 1**.

Tabela 14: Exemplo dos dados passados a RNR

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	predição
1	1	0	1	0	0	1	1	0	1	0	1	1	1	0	0	1	0	1	0	1	0	1	0	1	0	1	1	1	0	0	1
1	0	1	0	0	1	1	0	1	0	1	1	1	0	0	1	0	1	0	1	0	1	0	1	0	1	1	1	0	0	1	1
0	1	0	0	1	1	0	1	0	1	1	1	0	0	1	0	1	0	1	0	1	0	1	0	1	1	1	0	0	1	1	0
1	0	0	1	1	0	1	0	1	1	1	0	0	1	0	1	0	1	0	1	0	1	0	1	1	1	0	0	1	1	0	1
0	0	1	1	0	1	0	1	1	1	0	0	1	0	1	0	1	0	1	0	1	0	1	1	1	0	0	1	1	0	1	1

Fonte: Elaborada pelo autor

Em seguida foi criada uma RNR utilizando LSTM com a seguinte configuração: uma camada oculta com 100 neurônios utilizando a função de ativação Tangente Hiperbólica (TanH) e uma camada de saída com 1 neurônio utilizando a função de ativação sigmoide (PIECH et al., 2015). Adicionalmente, foram passados os dados simplificados extraídos do STI PAT2Math como entrada para a rede. Porém, não foi obtido um resultado satisfatório visto que os dados

que são utilizados nos trabalhos relacionados são oriundos de um STI baseado em respostas. Nesses STIs, quando o estudante acerta a solução na primeira tentativa somente uma interação é armazenada no banco de dados. Já nos STIs baseados em passos, para o estudante resolver um exercício, na grande maioria das vezes, mais de um passo é armazenado no banco de dados. Assim, serão armazenadas no banco de dados tantos acertos quanto os passos certos dado pelo estudante para a solução do exercício.

Na continuidade, buscou-se outros modos de testar a RNR, inserindo nela dados mais próximos daqueles fornecidos pelo STI PAT2Math. Uma das tentativas se deu através do fornecimento de informações mais detalhadas do último passo do estudante, adicionando-se a informação se o estudante acertou ou errou os 30 passos anteriores a este, objetivando prever qual seria o resultado do próximo passo por ele dado, a esse formato foi dado o nome de **Formatação 2**. Porém, novamente, não foi obtido um bom resultado. Por fim, foi testado o fornecimento à RNR os dados conforme resultantes do pré-processamento, essa tentativa foi nomeada de **Formatação 3**.

Tabela 15: Resumo das Formatações fornecidas a RNR

	Dados	RNR
Formatação 1	Dados com sequencias de respostas fornecidas pelo estudante, 31 respostas para identificar a 32º.	Uma camada oculta com 100 neurônios utilizando a função de ativação TanH Uma camada de saída com 1 neurônio utilizando a função de ativação Sigmoide.
Formatação 2	Dados detalhados do último passo do estudante adicionado aos 30 passos anteriores buscando identificar o próximo passo.	
Formatação 3	Foram passados os dados pré-processados conforme o explicado na Seção 6.1	

Um resumo das formatações que serão utilizadas na RNR criada para rastrear o conhecimento do estudante pode ser visto na Tabela 15. Os resultados da avaliação desses experimentos serão explicitados no Capítulo 7.

6.2.2 Modelo de estudante usando MLP

Visando uma alternativa a utilização da RNR abordada na seção 6.2.1, foi desenvolvida uma abordagem que buscasse alcançar o objetivo desse trabalho. Para tanto foi desenvolvida uma Rede Neural Artificial (RNA) utilizando *Multi-layer Perceptron* (MLP) como uma possibilidade de construção de um modelo de estudante. Para a construção do modelo foram utilizadas as bibliotecas *Tensorflow* na versão 2.1 e a *Keras* na versão 2.2.4.

Segundo Silva e Oliveira (2001), um dos grandes problemas na criação de RNAs é a definição dos parâmetros da rede como quantidade de camadas ocultas, quantidade de neurônios em cada camada, qual a melhor função de ativação, entre outros. Algumas vezes, pequenas alterações nesses parâmetros levam a uma grande diferença no resultado do treinamento. Existem

diversas pesquisas que tentam encontrar uma possível fórmula “mágica”, porém até o momento nenhum conseguiu definir tais valores, somente sugestões levando em consideração experimentos realizados.

Assim, a primeira RNA criada levou em consideração alguns desses estudos. Em seguida, foram criados experimentos a fim de verificar a possibilidade de melhoria nos resultados obtidos pela RNA através de alterações nos parâmetros. Nesse sentido foi criada uma RNA com uma camada oculta com 100 neurônios, a escolha dessa quantidade de neurônios foi empírica, utilizando a função de ativação ReLu e uma camada de saída com um único neurônio que utiliza como função de ativação sigmoide, visto que a saída é binária. Devido a esta RNA estar sendo utilizada para a classificação binária dos dados, foi utilizada a função de perda *binary cross-entropy*, tendo por função atualizar os pesos dos neurônios da RNA para obter a melhor generalização dos mesmos. A esta RNA inicial foi dado o nome de **Modelo 1**.

Este modelo permitiu, em um primeiro momento constatar que a formatação inicial dos dados utilizados não estavam corretos, pois ao passar os dados como eram retornados do pré-processamento para a RNA, a mesma tentava identificar se o estudante teria ou não acertado o passo em questão, porém essa não era a resposta esperada. Para corrigir o problema adicionou-se uma nova coluna contendo a informação se o estudante acertou ou errou o próximo passo, tornando-se essa coluna o objetivo da RNA. Após a correção do problema, buscou-se a melhor formatação para a RNA em questão.

Já que nos trabalhos relacionados nenhum deles abordou a criação de uma RNA, foi realizada uma experimentação exploratória, alterando-se os parâmetros do Modelo 1 a fim de se obter os melhores resultados nas métricas utilizadas. O próximo passo foi realizar uma alteração de cada vez nos parâmetros da RNA, após cada alteração o processo de treinamento, validação e teste da rede foi executado, estando os resultados obtidos demonstrados no Capítulo 7.

No segundo modelo criado foi alterada a quantidade de neurônios na camada oculta do Modelo 1, aumentando-se o quantitativo para 1.000 neurônios nessa camada. Os demais parâmetros da RNA permaneceram os mesmos. Assim, qualquer diferença nos resultados dos testes se daria somente em função desta alteração. A este modelo foi dado o nome **Modelo 2**.

A terceira abordagem se deu na adição de mais uma camada oculta de neurônios, com 100 neurônios, igualmente ao Modelo 1, também utilizando a função de ativação ReLu. E, novamente, os demais parâmetros da rede permaneceram inalterados, resultando uma RNA com duas camadas ocultas com 100 neurônios cada, utilizando a função de ativação ReLu, e uma camada de saída utilizando a função de ativação sigmoide. A essa RNA foi dada o nome de **Modelo 3**.

A próxima RNA que foi desenvolvida foi a utilização do Modelo 3 aumentando a quantidade de neurônios em cada camada para 1.000 neurônios e mantendo as demais configurações. A essa RNA foi dado o nome de **Modelo 4**. Até o momento, foram feitas modificações na quantidade de neurônios e de camadas ocultas, assim como outras alterações foram feitas como, por exemplo, a alteração do otimizador do treinamento.

Para a criação do próximo modelo foram desenvolvidos diversos treinamentos, porém, para tornar o texto mais conciso, apenas um será apresentado. Assim, a RNA desenvolvida foi a utilizada no Modelo 1, sendo que foi alterado o otimizador do treinamento para *RMSProp*, que divide a taxa de aprendizado de um peso por uma média constante das magnitudes dos gradientes recentes desse peso, sendo nomeado de **Modelo 5**.

Tabela 16: Resumo dos modelos de RNA testados

Modelo	Dados	RNA
Modelo 1	Foram passados os dados pré-processados conforme explicados na Seção 6.1	Uma camada oculta com 100 neurônios, usando a função de ativação ReLu e uma camada de saída com 1 neurônio usando a função de ativação Sigmoide. Utilizando o otimizador Adam.
Modelo 2		Uma camada oculta com 1000 neurônios, usando a função de ativação ReLu e uma camada de saída com 1 neurônio usando a função de ativação Sigmoide. Utilizando o otimizador Adam.
Modelo 3		Duas camadas ocultas com 100 neurônios cada, usando a função de ativação ReLu e uma camada de saída com 1 neurônio usando a função de ativação Sigmoide. Utilizando o otimizador Adam.
Modelo 4		Duas camadas ocultas com 1000 neurônios cada, usando a função de ativação ReLu e uma camada de saída com 1 neurônio usando a função de ativação Sigmoide. Utilizando o otimizador Adam.
Modelo 5		Uma camada oculta com 100 neurônios, usando a função de ativação ReLu e uma camada de saída com 1 neurônio usando a função de ativação Sigmoide. Utilizando o otimizador RMSProp.

Fonte: Elaborada pelo autor.

Um resumo dos modelos criados pode ser visualizado na Tabela 16, e os resultados obtidos por cada um desses modelos será demonstrado no Capítulo 7. Além desses modelos, foram experimentados um maior aumento de camadas ocultas e/ou aumento do número de neurônios, porém, segundo Silva e Oliveira (2001), esse aumento na quantidade de camadas ocultas, além de não trazer um ganho no resultado da rede, ainda acaba resultando no aumento exacerbado do tempo para treinamento.

6.3 Métricas

Para a avaliação do presente trabalho serão utilizadas duas métricas principais para avaliação dos modelos aqui criados. A primeira métrica será a *Area Under The ROC Curve* (AUC), consistindo em um valor numérico que representa o grau de separabilidade dos acertos de um algoritmo. Utiliza como base a *Receiver Operating Characteristics* (ROC), que é criada traçando a taxa de verdadeiro positivo (VP) contra a taxa de falso positivo (FP), ou seja, a quantidade de vezes que o classificador acertou a predição contra a quantidade de vezes que ele errou. Essa métrica foi adotada por ser a mesma utilizada em todos os trabalhos relacionados.

A segunda métrica utilizada é a *F1 Score*, sendo uma medida muito utilizada em modelos de classificação binária, objetivo traçado por este trabalho. Tal métrica leva em consideração a precisão (*precision*) e a revocação (*recall*). A precisão pode ser definida como a capacidade do algoritmo de evitar falsos positivos. Para identificar esse valor, pode-se utilizar a equação 6.1, onde VP significa verdadeiro positivo e FP significa falso positivo. Já a revocação é a frequência com que o classificador encontra os exemplos de um valor como pode ser visualizado na equação 6.2, onde FN representa falso negativo. Assim, para obter a métrica *F1 Score* é utilizada a equação 6.3.

$$Precision = \frac{VP}{VP + FP} \tag{6.1}$$

$$Recall = \frac{VP}{VP + FN} \tag{6.2}$$

$$F_1 = 2.\frac{precision.recall}{precision + recall} \tag{6.3}$$

Além dessas métricas ainda será, em alguns casos, analisada a matriz de confusão que quantifica os resultados de verdadeiro positivo (VP), falso positivo (FP), verdadeiro negativo (VN) e falso negativo (FN) em um quadro como o exibido na Tabela 17 para analisar a quantidade exata de tuplas que o classificador conseguiu identificar de forma correta.

Tabela 17: Matriz de Confusão

		Predito	
		Classe A	Classe B
Real	Classe A	VP	FN
	Classe B	FP	VN

Fonte: Traduzido e adaptado de Luque et al. (2019)

7 RESULTADOS E DISCUSSÕES

Esse capítulo se destina a explicitar os resultados obtidos durante todo o processo de pré-processamento, validação e dos testes em cada um dos modelos desenvolvidos. Os primeiros resultados a serem exibidos serão os obtidos com a RNR explorada na seção 6.2.1, em virtude desse modelo ter sido o primeiro experimentado no desenvolvimento deste trabalho, bem como por se tratar do modelo original proposto por (PIECH et al., 2015).

Os primeiros experimentos que foram executados reuniram a extração simplificada dos dados por se tratar do formato original dos trabalhos relacionados, assim como por essa configuração ter obtido um resultado satisfatório e dentro do alcançado pelos trabalhos relacionados. Porém, o que se buscou na construção desse trabalho foi a utilização dos dados mais completos conforme explicado na Seção 6.1, de forma a permitir que a rede pudesse prever o próximo passo do estudante com mais propriedade.

Os resultados obtidos pelas formatações exibidas na Seção 6.2.1 podem ser visualizados na Tabela 18, na qual é possível verificar, por exemplo, que a **Formatação 1** foi a que obteve o melhor resultado, visto que os dados estavam dispostos da mesma maneira que aquela utilizada nos trabalhos relacionados.

Tabela 18: Resultados dos testes com RNR

Formatação dos dados	AUC	F1 Score
Formatação 1	**0,7885**	**0,8735**
Formatação 2	0,5875	0,7427
Formatação 3	0,5000	0,8506

Fonte: Elaborada pelo autor

Ainda analisando a Tabela 18 verifica-se que a **Formatação 3** obteve um resultado de *F1 Score* bem próximo ao melhor resultado obtido, porém, ao analisar a métrica AUC essa formatação obteve uma avaliação de 0,5, apontando um forte indício de que a RNR "chutou" os valores ao invés de tentar prevê-los. Tal afirmação se confirma verdadeira ao analisar a matriz de confusão na Tabela 19. Através dela se verifica que a RNR simplesmente "chutou" todos os valores no acertar, tendo assim acertado 7.472 registros, e, consequentemente, errando 2.625 registros.

Tabela 19: Matriz de confusão formatação 3

		Classe Prevista	
		Acerta	Erra
Classe Esperada	Acerta	7.472	0
	Erra	2.625	0

Fonte: Elaborada pelo autor

Já a Formatação 2 obteve uma classificação intermediária, não obtendo uma avaliação muito

boa em nenhuma das métricas analisadas, alcançando uma nota para *f1 score* de 0,7427, resultando em um valor mais baixo do que na Formatação 3, onde a RNR simplesmente "chutou" na classe que continha mais registros. Assim, se encerram os testes realizados sobre as RNR, não sendo alcançado o resultado esperado quando utilizada a formatação dos dados conforme o desejado. No entanto, outra etapa foi iniciada, buscando uma alternativa à RNR para a modelagem do conhecimento do estudante, utilizando os dados conforme aqueles explicados na Seção 6.1.

A alternativa encontrada se deu através da criação das RNAs, dando início, então, aos testes utilizando os modelos criados na seção 6.2.2. Foram desenvolvidos cinco modelos, conforme demonstrado. Na primeira tentativa foram realizados testes com os cinco modelos utilizando os dados pré-processados, porém desbalanceados, visto que até esse momento não havia sido identificado que esse desbalanceamento dos dados poderia interferir na classificação final obtida.

Durante o processo de treinamento dos modelos foram realizados testes utilizando os dados do conjunto de validação, ao fim do treinamento foram realizados os testes com os dados do conjunto de teste. Os resultados obtidos em ambos os testes podem ser visualizados na Tabela 20.

Tabela 20: Resultados dos testes com RNA com os dados desbalanceados

Modelos	Conjunto de Validação		Conjunto de Teste	
	AUC	F1 Score	AUC	F1 Score
Modelo 1	0,6815	0,8701	0,7621	0,8705
Modelo 2	0,6629	0,8781	0,7567	**0,8781**
Modelo 3	**0,6883**	0,8696	**0,7655**	0,8689
Modelo 4	0,6567	**0,8801**	0,7465	0,8759
Modelo 5	0,6724	0,8786	0,7593	0,8775

Fonte: Elaborada pelo autor

Analisando a Tabela 20 constata-se que o Modelo 3, o qual possui duas camadas ocultas com 100 neurônios cada, obteve a melhor avaliação quando levada em consideração a métrica AUC, que foi a métrica utilizada nos trabalhos relacionados. Noutro sentido, quando verificada a métrica *F1 Score*, deixou de obter o melhor resultado, mas se aproximou bastante dele.

Então, se encaminhava a finalização dos testes, identificou-se que o desbalanceamento dos dados do PAT2Math estava interferindo no treinamento realizado nos modelos. Contatou-se que dos dados separados para o treinamento dos modelos, 78.785 classificações eram positivas, significando que o estudante acertaria a próxima interação, contra 28.460 negativas, ou seja, que o estudante erraria a próxima interação.

Constada a problemática envolvendo o desbalanceamento, foi necessário ser iniciado o processo de balanceamento dos dados de treinamento. Primeiramente, buscou-se identificar as técnicas de balanceamento dos dados. Há três principais técnicas de balanceamento, a *over-sampling*, a *under-sampling*, bem como uma terceira que se trata de uma combinação das duas.

Como visto, nesse trabalho os dados estão divididos em classe positiva, ou seja, o estudante acertaria a próxima interação, contando com 78.785 registros, e na classe negativa, na qual constam 28.460 registros. A técnica de *over-sampling* consiste no incremento dos dados na classe que possui menos dados. Assim, buscando alcançar o balanceamento dos dados, aplicou-se os métodos apresentados na Seção 6.1.2.1, resultando em dois datasets próprios, cada um contendo as quantidades de dados balanceados em cada classe. Após, os datasets resultantes foram utilizados para treinar a RNA do **Modelo 3**; esse foi o que obteve a melhor avaliação com os dados desbalanceados na métrica AUC. Os resultados obtidos com cada um dos datasets utilizados para treinar a RNA estão sendo exibidos na Tabela 21.

Tabela 21: Resultados dos treinamentos com dados balanceados com *over-sampling*

Método	AUC	F1 Score
SMOTE	**0,7687**	**0,8657**
ADASYN	0,7558	0,8231

Fonte: Elaborada pelo autor

Analisando a Tabela 21, constata-se que a técnica de *over-sampling* que utiliza o método SMOTE obteve uma melhor avaliação nas duas métricas utilizadas. Assim, esse método foi o eleito para o treinamento com os demais modelos que foram criados na Seção 6.2.2. Para tanto, foram treinados todos os demais modelos utilizando os dados de treinamento gerados com o método SMOTE. Os resultados desses treinamentos podem ser visualizados na Tabela 22, na qual se pode constatar que o Modelo 3 obteve o melhor desempenho nesta técnica de balanceamento de dados.

Tabela 22: Resultados do treinamento dos demais modelos com dados balanceados com *over-sampling*

Modelo	Conjunto de Validação		Conjunto de Teste	
	AUC	F1 Score	AUC	F1 Score
Modelo 1	**0,6993**	0,8288	0,7620	0,8292
Modelo 2	0,6964	0,8252	0,7605	0,8294
Modelo 3	0,6949	**0,8653**	**0,7687**	**0,8657**
Modelo 4	0,6958	0,8560	0,7642	0,8530
Modelo 5	0,6992	0,8209	0,7560	0,8231

Fonte: Elaborada pelo autor

Na sequência, a mesma metodologia foi adotada para a técnica de *under-sampling*, em que os dados da classe mais abundante são reduzidos a uma quantidade aproximada da classe que possui menos registros. Assim, foram executados os treinamentos do Modelo 3 com os dados oriundos de cada um dos métodos da técnica de *under-sampling*, obtendo os resultados constantes da Tabela 23.

Após execução dos primeiros testes valendo-se dos dados balanceados com a técnica *under-sampling*, pode-se verificar, analisando a Tabela 23, que o método que obteve o melhor desem-

Tabela 23: Resultados dos treinamentos com dados balanceados com *under-sampling*

Método	AUC	F1 Score
NearMiss	0,7595	0,8101
AllKNN	0,7605	0,8176

Fonte: Elaborada pelo autor

penho foi o AllKNN. Assim, então, seguiu-se o mesmo princípio utilizado no balanceamento *over-sampling*, no qual usou-se os dados gerados pelo melhor método para realizar o treinamento dos demais modelos, obtendo-se os resultados da Tabela 24.

Tabela 24: Resultados do treinamento dos demais modelos com dados balanceados com *under-sampling*

Modelo	Conjunto de Validação		Conjunto de Teste	
	AUC	F1 Score	AUC	F1 Score
Modelo 1	0,7113	**0,8295**	**0,7620**	**0,8240**
Modelo 2	0,7091	0,8134	0,7609	0,8165
Modelo 3	**0,7123**	0,8247	0,7606	0,8176
Modelo 4	0,7103	0,8169	0,7604	0,8140
Modelo 5	0,7105	0,8164	0,7587	0,8121

Fonte: Elaborada pelo autor

Nos resultados da Tabela 24 é possível verificar que o Modelo 1 obteve o melhor desempenho. Salienta-se que esse é o Modelo mais simples, contando com uma camada oculta de 100 neurônios. Pode-se defender, ainda, que a menor quantidade de dados utilizados no treinamento desses modelos, exigiu menos camadas ocultas e também uma menor quantidade de neurônios por camada para atingir o seu melhor desempenho, concluindo que, ao aumentar a quantidade de camadas e/ou neurônios, os resultados são menos expressivos do que quando se trata de redes menos complexas.

Por fim, foi aplicada a técnica de balanceamento dos dados que combina as técnicas de *over-sampling* e *under-sampling*. Primeiramente, foram realizados treinamentos no Modelo 3 utilizando os dados balanceados com os dois métodos dessa técnica. Após esse treinamento, foram realizados os testes, resultando nos dados constantes da Tabela 25.

Tabela 25: Resultados dos treinamentos com dados balanceados com técnicas combinadas

Método	AUC	F1 Score
SMOTETomek	0,7458	0,8038
SMOTEENN	**0,7509**	**0,8064**

Fonte: Elaborada pelo autor

Analisando a Tabela 25, constata-se que o método SMOTEENN obteve um melhor desempenho, ainda que pouco mais expressivo que o método SMOTETomek. Então, igualmente ao apresentado nos demais casos, foi realizado o treinamento dos demais modelos utilizando os

dados balanceados com as técnicas combinadas que utilizaram o método SMOTEENN. Após os treinamentos foram obtidos os resultados que são ilustrados na Tabela 26.

Tabela 26: Resultados do treinamento dos demais modelos com dados balanceados com técnicas combinadas

Modelo	Conjunto de Validação		Conjunto de Teste	
	AUC	F1 Score	AUC	F1 Score
Modelo 1	0,7064	0,8150	0,7569	0,8120
Modelo 2	0,7039	0,8174	0,7610	0,8189
Modelo 3	0,7062	0,8078	0,7509	0,8064
Modelo 4	**0,7100**	**0,8403**	**0,7674**	**0,8386**
Modelo 5	0,7098	0,8254	0,7644	0,8242

Fonte: Elaborada pelo autor

Ao analisar a Tabela 26, pode-se constatar que o Modelo 4, o qual possui duas camadas com 1000 neurônios cada, foi o que obteve o melhor desempenho com essa técnica. Porém, essa análise individualizada não reproduz o significado almejado por esse trabalho. Portanto, se faz necessária uma análise de todos os resultados em paralelo para identificar qual dos modelos obteve, de fato, o melhor desempenho.

Para possibilitar a análise paralela, serão utilizados somente os resultados obtidos pelos conjuntos de teste, visto que em todos os experimentos os resultados dos testes com o conjunto de validação estão muito próximos dos obtidos com o conjunto de teste.

Tabela 27: Resultados de todos os treinamentos utilizando RNA

	Dados desbalanceados		Técnica de *over-sampling*	
	AUC	F1 Score	AUC	F1 Score
Modelo 1	0,7621	0,8705	0,7620	0,8292
Modelo 2	0,7567	**0,8781**	0,7605	0,8294
Modelo 3	**0,7655**	0,8689	**0,7678**	**0,8657**
Modelo 4	0,7465	0,8759	0,7642	0,8530
Modelo 5	0,7593	0,8775	0,7560	0,8231

	Técnica de *under-sampling*		Técnicas combinadas	
	AUC	F1 Score	AUC	F1 Score
Modelo 1	**0,7620**	**0,8240**	0,7569	0,8120
Modelo 2	0,7609	0,8165	0,7610	0,8189
Modelo 3	0,7606	0,8176	0,7509	0,8064
Modelo 4	0,7604	0,8140	**0,7674**	**0,8386**
Modelo 5	0,7587	0,8121	0,7644	0,8242

Fonte: Elaborada pelo autor

Ao se analisar a Tabela 27 pode-se visualizar que os dados balanceados com as técnicas *under-sampling* e com as técnicas combinadas obtiveram os piores resultados, o que ocorreu por uma significativa redução na quantidade de dados para o treinamento em decorrência da exclusão de dados da classe mais abundante. Nas técnicas combinadas, foi necessária a utilização

do modelo com maior número de camadas e neurônios por camada para se obter o melhor resultado com esses dados. Acredita-se que devido a inversão das quantidades de dados em cada classe pode ter ocorrido a perda de algumas características importantes para o desempenho da RNA.

Segundo Chen e Wasikowski (2008), quando constatado um desequilíbrio entre as amostras positivas e negativas dos conjuntos de treinamento e/ou de teste, é recomendado desconsiderar as medidas da curva ROC, já que para se obter um valor de "F" alto, tanto os valores de precisão quanto os de revogação devem, igualmente, serem altos. Assim, quando os dados a serem utilizados estiverem desbalanceados, deve-se realizar a avaliação utilizando outras métricas, como por exemplo, *F1-Score*. Ainda que desaconselhada a aplicação de medidas da curva ROC, em virtude de ter sido a métrica utilizada nos trabalhos relacionados, optou-se por valer-se dela nos experimentos realizados nesse trabalho.

Dessa forma, pode-se constatar que a técnica de *over-sampling* obteve um resultado de AUC melhor do que os dados originais, ou seja, quando ainda estavam desbalanceados, já que essa métrica se comporta melhor após o balanceamento dos dados. No entanto, quando analisada a métrica *F1 Score*, ficou constatado uma pequena vantagem alcançada pelos dados desbalanceados, quando comparados aos dados balanceados usando a técnica de *over-sampling*. O mesmo ocorre em virtude de que quando utilizados os dados desbalanceados, a RNA pode ter uma certa especialização de seus pesos para a classe mais abundante, ocasionando, assim, uma melhor avaliação com essa métrica. Porém, constata-se, ainda, que o resultado obtido com os dados balanceados com a técnica *over-sampling* apresentou uma diferença muito pequena, mostrando que mesmo não ocorrendo a especialização da RNA, essa se mostrou muito competente ao prever o próximo passo do estudante.

Por fim, tem-se que a técnica de *over-sampling* se mostrou a que melhor se adaptou aos dados extraídos do STI PAT2Math após a realização do balanceamento, ficando, na conclusão, constatado que a RNA do Modelo 3, aquela contendo duas camadas ocultas de neurônios com 100 neurônios cada, foi a que obteve o melhor desempenho na predição do próximo passo do estudante.

8 CONCLUSÃO

O presente trabalho objetivou a criação de um modelo de estudante para ser utilizado em STIs baseados em passos. Mais especificamente, o modelo proposto visa prever se o estudante irá acertar o próximo passo. Para alcançar o objetivo pretendido, foram utilizadas técnicas de *Deep Learning* (DL); esse modelo de estudante utilizando DL é denominado de *Deep Knowledge Tracing* (DKT).

No trabalho foram aplicadas duas técnicas de DL. Primeiramente, testes foram executados utilizando a RNR, a qual foi escolhida por ter sido utilizada pelos trabalhos relacionados. Essa abordagem alcançou um resultado satisfatório, porém, a forma como os dados são entregues para a RNR não se encontravam da forma como era o desejado para esse trabalho. Conforme esperado, ao passar os dados da forma como se objetivava, compreenda-se, dados desbalanceados e pré-processados, a RNR não obteve um resultado satisfatório, já que ao simplesmente "chutar" na classe que possuía mais registros, gerou um resultado muito aquém do desejado. Então, uma nova abordagem foi testada, valendo-se da segunda técnica de DL: RNA. Salienta-se, que o algoritmo RNA não foi utilizado em nenhum dos trabalhos encontrados até então, representando, dessa forma, uma inovação alcançada pelo trabalho desenvolvido.

Buscando a construção do modelo de estudante utilizando a técnica de RNA, foi aplicada uma abordagem simplificada. Para tanto, foi criada uma RNA básica com uma camada oculta contando com 100 neurônios, utilizando a função de ativação ReLu e uma camada de saída com um neurônio, utilizando a função de ativação sigmoide. Esse modelo foi criado por se almejar a verificação da possibilidade de utilização dos dados no formato desejado, bem como se seria capaz de alcançar o objetivo de avaliar o desempenho do estudante.

Ao fim do primeiro teste foi constatado que seria possível a utilização dos dados no formato desejado, desde que os mesmos passassem por uma etapa de pré-processamento, a qual foi explorada na Seção 6.1. Tal etapa se fez necessária em virtude da impossibilidade encontrada pelas redes neurais em trabalhar com dados categóricos, além de buscar reduzir a quantidade de dados advindos do OHE, reduzindo as equações aos seus formatos genéricos, onde valores numéricos são substituídos por variáveis. Os dados resultantes desse pré-processamento foram, então, enviados à RNA, a qual conseguiu inferir se o estudante acertaria ou não a próxima interação.

Na sequência, foi realizada uma busca para identificar qual seria a melhor configuração para a construção da RNA com o fim de alcançar o objetivo de classificação dos dados. Os resultados encontrados não eram exatos ao informar qual seria a melhor formatação, apenas deixando indícios sobre qual configuração poderia ser utilizada para o fim de classificar os dados. Ao se estudar os trabalhos relacionados, verificou-se que nenhuma configuração foi apontada como a melhor nos trabalhos pesquisados. Por tal motivo, uma busca empírica foi realizada a fim de identificar qual formatação alcançaria o melhor resultado.

Os dados pré-processados foram, assim, fornecidos para o treinamento em vários modelos,

cada qual possuindo uma possível configuração para se obter o melhor desempenho na predição do próximo passo do estudante. Inicialmente, o modelo contava com uma camada de neurônios ocultos com 100 neurônios, o qual foi recebendo alterações nas suas configurações em um parâmetro por vez. Como segunda abordagem foi aumentado o número de neurônios na camada oculta de 100 para 1.000 neurônios. O próximo passo foi a inclusão de mais uma camada oculta de neurônios, resultando, assim, em duas camadas ocultas de neurônios com 100 neurônios cada. Em seguida, foi criada outra RNA com o aumento na quantidade de neurônios em cada camada, chegando a um modelo com duas camadas ocultas de neurônios com 1.000 neurônios cada.

Ainda foram realizados diversos testes com alterações nos demais parâmetros da RNA, as quais não alcançaram melhorias para a predição, como foi o caso da RNA com uma camada oculta de neurônios com 100 neurônios utilizando o otimizador "RMSProp". Após a criação dos modelos, passou-se a execução dos treinamentos e testes.

Na conclusão desses testes, foi identificado que os dados utilizados estavam desbalanceados. Para balancear os dados e evitar o problema da especialização dos modelos a uma classe em detrimento da outra, buscou-se alternativas para o balanceamento. Assim, foram identificadas as três técnicas de balanceamento dos dados, a técnica de *over-sampling*, a técnica de *under-sampling* e uma terceira técnica, que se trata da combinação das duas anteriores. Após o balanceamento dos dados, foi realizado um novo treinamento dos modelos contando com os dados balanceados, ficando constatado que a técnica de *over-sampling* obteve os melhores resultados.

Conclui-se que o objetivo de utilizar DL para modelar o conhecimento do estudante é viável, ainda que em STIs baseados em passos. Os trabalhados relacionados se utilizavam de uma organização de dados simplificada, a qual não se mostrou efetiva quando aplicada aos dados do STI baseado em passos, como é o caso do STI PAT2Math. Já a RNA se mostrou mais efetiva com esse tipo de dados, visto que recebe informações mais completas para estruturar a predição. Assim, como resultado desse trabalho ficou identificado que a RNA utilizando duas camadas ocultas de neurônios com 100 neurônios cada, obteve o melhor resultado quando se vale dos dados balanceados através da técnica de *over-sampling* para utilização dos dados mais completos do STI baseado em passos PAT2Math.

Como trabalhos futuros se objetiva a integração da RNA desenvolvida ao STI PAT2Math, visto que essa se mostrou muito eficiente para o fim almejado. Essa integração permitirá ao STI personalizar de forma mais precisa os conteúdos trabalhados ao conhecimento atual do estudante, permitindo assim que o sistema se adapte ao conhecimento e ritmo de aprendizado do estudante. Outra ação indicada é o estudo de outras arquiteturas de redes neurais que visem a obtenção de resultados mais satisfatórios utilizando dados de um STI baseado em passos.

REFERÊNCIAS

ABRAHAMSOHN, P.; FREITAS, V. **Tecido nervoso**. Acessado em: 14/05/2020, <http://mol.icb.usp.br/index.php/9-13-tecido-nervoso/>.

ANDERSON, J. R.; CORBETT, A. T.; KOEDINGER, K. R.; PELLETIER, R. Cognitive tutors: lessons learned. **The journal of the learning sciences**, [S.l.], v. 4, n. 2, p. 167–207, 1995.

ARBIB, M. A. **Brains, machines, and mathematics**. 2. ed. New York: Springer-Verlag, 1987.

BAKER, R. S.; CORBETT, A. T.; KOEDINGER, K. R. Detecting student misuse of intelligent tutoring systems. In: INTERNATIONAL CONFERENCE ON INTELLIGENT TUTORING SYSTEMS, 2004. **Anais...** [S.l.: s.n.], 2004. p. 531–540.

BAKER, R. S. d; CORBETT, A. T.; ALEVEN, V. More accurate student modeling through contextual estimation of slip and guess probabilities in bayesian knowledge tracing. In: INTERNATIONAL CONFERENCE ON INTELLIGENT TUTORING SYSTEMS, 2008. **Anais...** [S.l.: s.n.], 2008. p. 406–415.

BARANIUK, R. G. Compressive sensing. **IEEE signal processing magazine**, [S.l.], v. 24, n. 4, 2007.

BATISTA, G. E.; BAZZAN, A. L.; MONARD, M. C. Balancing Training Data for Automated Annotation of Keywords: a case study. In: WOB, 2003. **Anais...** [S.l.: s.n.], 2003. p. 10–18.

BATISTA, G. E.; PRATI, R. C.; MONARD, M. C. A study of the behavior of several methods for balancing machine learning training data. **ACM SIGKDD explorations newsletter**, [S.l.], v. 6, n. 1, p. 20–29, 2004.

BLOOM, B. S. The 2 sigma problem: the search for methods of group instruction as effective as one-to-one tutoring. **Educational researcher**, [S.l.], v. 13, n. 6, p. 4–16, 1984.

BRAGA, A. d. P.; FERREIRA, A. C. P. d. L.; LUDERMIR, T. B. **Redes neurais artificiais**: teoria e aplicações. [S.l.]: LTC Editora Rio de Janeiro, Brazil:, 2007.

BROWN, J. S.; BURTON, R. R. Diagnostic models for procedural bugs in basic mathematical skills. **Cognitive science**, [S.l.], v. 2, n. 2, p. 155–192, 1978.

BURTON, R. An Inverstigation of Computer Coaching for Informal Learning Activities. In: INTELLIGENT TUTORING SYSTEMS, 1982. **Anais...** [S.l.: s.n.], 1982. p. 79–98.

CARBONELL, J. R. AI in CAI: an artificial-intelligence approach to computer-assisted instruction. **IEEE transactions on man-machine systems**, [S.l.], v. 11, n. 4, p. 190–202, 1970.

CARVALHO, H. V. d.; CARVALHO, E. C.; ARRUDA, H.; IMPERATRIZ-FONSECA, V.; SOUZA, P. de; PESSIN, G. Detecção de anomalias em comportamento de abelhas utilizando redes neurais recorrentes. In: IX WORKSHOP DE COMPUTAÇÃO APLICADA A GESTÃO DO MEIO AMBIENTE E RECURSOS NATURAIS, 2018. **Anais...** [S.l.: s.n.], 2018.

CHAPELLE, O.; ZHANG, Y. A dynamic bayesian network click model for web search ranking. In: WORLD WIDE WEB, 18., 2009. **Proceedings...** [S.l.: s.n.], 2009. p. 1–10.

CHARNIAK, E.; GOLDMAN, R. P. A Semantics for Probabilistic Quantifier-Free First-Order Languages, with Particular Application to Story Understanding. In: IJCAI, 1989. **Anais...** [S.l.: s.n.], 1989. v. 89, p. 1074–1079.

CHAWLA, N. V.; BOWYER, K. W.; HALL, L. O.; KEGELMEYER, W. P. SMOTE: synthetic minority over-sampling technique. **Journal of artificial intelligence research**, [S.l.], v. 16, p. 321–357, 2002.

CHEN, X.-w.; WASIKOWSKI, M. Fast: a roc-based feature selection metric for small samples and imbalanced data classification problems. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 14., 2008. **Proceedings...** [S.l.: s.n.], 2008. p. 124–132.

CONATI, C.; GERTNER, A.; VANLEHN, K. Using Bayesian networks to manage uncertainty in student modeling. **User modeling and user-adapted interaction**, [S.l.], v. 12, n. 4, p. 371–417, 2002.

CORBETT, A. T.; ANDERSON, J. R. Student modeling and mastery learning in a computer-based programming tutor. In: INTERNATIONAL CONFERENCE ON INTELLIGENT TUTORING SYSTEMS, 1992. **Anais...** [S.l.: s.n.], 1992. p. 413–420.

CORBETT, A. T.; ANDERSON, J. R. Knowledge tracing: modeling the acquisition of procedural knowledge. **User modeling and user-adapted interaction**, [S.l.], v. 4, n. 4, p. 253–278, 1994.

COUTO, L. N. **Sistema para localização robótica de veículos autônomos baseado em visão computacional por pontos de referência**. 2012. Tese (Doutorado em Ciência da Computação) — Universidade de São Paulo, 2012.

CYBENKO, G. Continuous valued neural networks with two hidden layer are suficient. Medford: tufts university. , [S.l.], 1988.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. **Mathematics of control, signals and systems**, [S.l.], v. 2, n. 4, p. 303–314, 1989.

DEAN, T.; BASYE, K.; LEJTER, M. Planning and active perception. In: DARPA WORKSHOP ON INNOVATIVE APPROACHES TO PLANNING, SCHEDULING, AND CONTROL, 1990. **Proceedings...** [S.l.: s.n.], 1990. p. 271–276.

DEAN, T.; KANAZAWA, K. A model for reasoning about persistence and causation. **Computational intelligence**, [S.l.], v. 5, n. 2, p. 142–150, 1989.

FUKUSHIMA, K. Cognitron: a self-organizing multilayered neural network. **Biological cybernetics**, [S.l.], v. 20, n. 3-4, p. 121–136, 1975.

FUNAHASHI, K.-I. On the approximate realization of continuous mappings by neural networks. **Neural networks**, [S.l.], v. 2, n. 3, p. 183–192, 1989.

GÉRON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow**: concepts, tools, and techniques to build intelligent systems. [S.l.]: O'Reilly Media, 2019.

GRAVES, A.; MOHAMED, A.-r.; HINTON, G. Speech recognition with deep recurrent neural networks. In: IEEE INTERNATIONAL CONFERENCE ON ACOUSTICS, SPEECH AND SIGNAL PROCESSING, 2013., 2013. **Anais...** [S.l.: s.n.], 2013. p. 6645–6649.

HA, T. M.; BUNKE, H. Off-line, handwritten numeral recognition by perturbation method. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, [S.l.], v. 19, n. 5, p. 535–539, 1997.

HAN, J.; PEI, J.; KAMBER, M. **Data mining**: concepts and techniques. [S.l.]: Elsevier, 2011.

HAYKIN, S. **Redes neurais**: princípios e prática. 2. ed. Porto Alegre: Bookman, 2001.

HE, H.; BAI, Y.; GARCIA, E. A.; LI, S. ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: IEEE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IEEE WORLD CONGRESS ON COMPUTATIONAL INTELLIGENCE), 2008., 2008. **Anais...** [S.l.: s.n.], 2008. p. 1322–1328.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural computation**, [S.l.], v. 9, n. 8, p. 1735–1780, 1997.

HORNIK, K.; STINCHCOMBE, M.; WHITE, H. et al. Multilayer feedforward networks are universal approximators. **Neural networks**, [S.l.], v. 2, n. 5, p. 359–366, 1989.

HU, B.; LU, Z.; LI, H.; CHEN, Q. Convolutional neural network architectures for matching natural language sentences. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 2014. **Anais...** [S.l.: s.n.], 2014. p. 2042–2050.

HU, F.; LI, H. A novel boundary oversampling algorithm based on neighborhood rough set model: nrsboundary-smote. **Mathematical Problems in Engineering**, [S.l.], v. 2013, 2013.

JAQUES, P. A.; SEFFRIN, H.; RUBI, G.; MORAIS, F. de; GHILARDI, C.; BITTENCOURT, I. I.; ISOTANI, S. Rule-based expert systems to support step-by-step guidance in algebraic problem solving: the case of the tutor pat2math. **Expert Systems with Applications**, [S.l.], v. 40, n. 14, p. 5456–5465, 2013.

JAQUES, P. A.; VICARI, R.; PESTY, S.; MARTIN, J.-C. Evaluating a cognitive-based affective student model. In: INTERNATIONAL CONFERENCE ON AFFECTIVE COMPUTING AND INTELLIGENT INTERACTION, 2011. **Anais...** [S.l.: s.n.], 2011. p. 599–608.

KANG, E. Long Short-Term Memory (LSTM): concept. **Accessed: Dec**, [S.l.], v. 10, p. 2019, 2017.

KARPATHY, A.; FEI-FEI, L. Deep visual-semantic alignments for generating image descriptions. In: IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION, 2015. **Proceedings...** [S.l.: s.n.], 2015. p. 3128–3137.

KASURINEN, J.; NIKULA, U. Estimating programming knowledge with Bayesian knowledge tracing. In: ACM SIGCSE BULLETIN, 2009. **Anais...** [S.l.: s.n.], 2009. v. 41, n. 3, p. 313–317.

KHAJAH, M.; LINDSEY, R. V.; MOZER, M. C. How deep is knowledge tracing? **arXiv preprint arXiv:1604.02416**, [S.l.], 2016.

LEE, J. I.; BRUNSKILL, E. The Impact on Individualizing Student Models on Necessary Practice Opportunities. **International Educational Data Mining Society**, [S.l.], 2012.

LEIJNEN, S.; VEEN, F. v. The Neural Network Zoo. In: MULTIDISCIPLINARY DIGITAL PUBLISHING INSTITUTE PROCEEDINGS, 2020. **Anais...** [S.l.: s.n.], 2020. v. 47, n. 1, p. 9.

LEMAîTRE, G.; NOGUEIRA, F.; ARIDAS, C. K. Imbalanced-learn: a python toolbox to tackle the curse of imbalanced datasets in machine learning. **Journal of Machine Learning Research**, [S.l.], v. 18, n. 17, p. 1–5, 2017.

LEVITT, T. S.; BINFORD, T. O.; ETTINGER, G. J. Utility-based control for computer vision. In: **Machine Intelligence and Pattern Recognition**. [S.l.]: Elsevier, 1990. v. 9, p. 407–422.

LIN, C.; CHI, M. A comparisons of BKT, RNN and LSTM for learning gain prediction. In: LECTURE NOTES IN COMPUTER SCIENCE (INCLUDING SUBSERIES LECTURE NOTES IN ARTIFICIAL INTELLIGENCE AND LECTURE NOTES IN BIOINFORMATICS), 2017. **Anais...** [S.l.: s.n.], 2017. v. 10331 LNAI, p. 536–539.

LUQUE, A.; CARRASCO, A.; MARTÍN, A.; HERAS, A. de las. The impact of class imbalance in classification performance metrics based on the binary confusion matrix. **Pattern Recognition**, [S.l.], v. 91, p. 216–231, 2019.

MACCULLOCH, W.; PITTS, W. A logical Calculus of Ideas Immanent in Nervous Activity. Bull. Mathematical Biophysics. Vol. 5. , [S.l.], 1943.

MANI, I.; ZHANG, I. kNN approach to unbalanced data distributions: a case study involving information extraction. In: OF WORKSHOP ON LEARNING FROM IMBALANCED DATASETS, 2003. **Proceedings...** [S.l.: s.n.], 2003. v. 126.

MANSKE, M.; CONATI, C. Modelling Learning in an Educational Game. In: AIED, 2005. **Anais...** [S.l.: s.n.], 2005. p. 411–418.

MATZ, M. Towards a process model for high school algebra errors. In: INTELLIGENT TUTORING SYSTEMS, 1982. **Anais...** [S.l.: s.n.], 1982. p. 25–50.

MAYO, M.; MITROVIC, A. Optimising ITS behaviour with Bayesian networks and decision theory. , [S.l.], 2001.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **The bulletin of mathematical biophysics**, [S.l.], v. 5, n. 4, p. 115–133, 1943.

MIKOLOV, T.; KARAFIÁT, M.; BURGET, L.; ČERNOCKỲ, J.; KHUDANPUR, S. Recurrent neural network based language model. In: ELEVENTH ANNUAL CONFERENCE OF THE INTERNATIONAL SPEECH COMMUNICATION ASSOCIATION, 2010. **Anais...** [S.l.: s.n.], 2010.

MILLÁN, E.; DESCALÇO, L.; CASTILLO, G.; OLIVEIRA, P.; DIOGO, S. Using Bayesian networks to improve knowledge assessment. **Computers & Education**, [S.l.], v. 60, n. 1, p. 436–447, 2013.

MILLÁN, E.; LOBODA, T.; CRUZ, J. L. Pérez-de-la. Bayesian networks for student model engineering. **Computers & Education**, [S.l.], v. 55, n. 4, p. 1663–1683, 2010.

MINSKY, M.; PAPERT, S. Perceptrons. **Cambridge, MA: MIT Press**, [S.l.], v. 18, p. 19, 1969.

MITROVIC, A. An intelligent SQL tutor on the web. **International Journal of Artificial Intelligence in Education**, [S.l.], v. 13, n. 2-4, p. 173–197, 2003.

MITROVIC, A.; KOEDINGER, K. R.; MARTIN, B. A comparative analysis of cognitive tutoring and constraint-based modeling. In: INTERNATIONAL CONFERENCE ON USER MODELING, 2003. **Anais...** [S.l.: s.n.], 2003. p. 313–322.

NEAPOLITAN, R. Probabilistic Reasoning in Expert Systems: theory and algorithms john wileys. **New York**, [S.l.], 1990.

NG, A. Machine learning yearning. **URL: http://www. mlyearning. org/(96)**, [S.l.], 2017.

OHLSSON, S. Constraint-based student modeling. In: **Student modelling**: the key to individualized knowledge-based instruction. [S.l.]: Springer, 1994. p. 167–189.

PARDOS, Z. A.; HEFFERNAN, N. T. Modeling individualization in a bayesian networks implementation of knowledge tracing. In: INTERNATIONAL CONFERENCE ON USER MODELING, ADAPTATION, AND PERSONALIZATION, 2010. **Anais...** [S.l.: s.n.], 2010. p. 255–266.

PEARL, J. **Probabilistic reasoning in intelligent systems**: networks of plausible inference. [S.l.]: Elsevier, 2014.

PIECH, C.; BASSEN, J.; HUANG, J.; GANGULI, S.; SAHAMI, M.; GUIBAS, L. J.; SOHL-DICKSTEIN, J. Deep knowledge tracing. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, 2015. **Anais...** [S.l.: s.n.], 2015. p. 505–513.

PROVOST, F.; FAWCETT, T. Robust classification for imprecise environments. **Machine learning**, [S.l.], v. 42, n. 3, p. 203–231, 2001.

PU, S.; YUDELSON, M.; OU, L.; HUANG, Y. Deep Knowledge Tracing with Transformers. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE IN EDUCATION, 2020. **Anais...** [S.l.: s.n.], 2020. p. 252–256.

PURVES, D.; AUGUSTINE, G.; FITZPATRICK, D.; HALL, W.; LAMANTIA, A.; WHITE, L. Neurociências. 2ª Edição. **Porto Alegre, Artmed editora**, [S.l.], 2005.

ROSENBLATT, F. **The perceptron, a perceiving and recognizing automaton Project Para**. [S.l.]: Cornell Aeronautical Laboratory, 1957.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **nature**, [S.l.], v. 323, n. 6088, p. 533–536, 1986.

RUSSELL, S.; NORVIG, P. Artificial intelligence: a modern approach. 3rd. **Prentice Hall Press, Upper Saddle River, NJ, USA. isbn**, [S.l.], v. 136042597, p. 9780136042594, 2009.

So BIOLOGIA. **Células nervosas**. Acessado em: 13/05/2020, <https://www.sobiologia.com.br/conteudos/FisiologiaAnimal/nervoso2.php>.

SEFFRIN, H. M. Avaliando o conhecimento algébrico do estudante através de redes bayesianas dinâmicas: um estudo de caso com o sistema tutor inteligente pat2math. , [S.l.], 2015.

SEFFRIN, H. M.; RUBI, G.; CARLOTTO, T.; MELLO, G.; JAQUES, P. A. Um resolvedor de equações algébricas como ferramenta de apoio à sala de aula no ensino de equações algébricas. In: WORKSHOP DE INFORMÁTICA NA ESCOLA, 2009. **Anais...** [S.l.: s.n.], 2009. v. 1, n. 1, p. 1791–1800.

SEFFRIN, H.; RUBI, G.; JAQUES, P. O Modelo Cognitivo do Sistema Tutor Inteligente PAT2Math. In: BRAZILIAN SYMPOSIUM ON COMPUTERS IN EDUCATION (SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO-SBIE), 2011. **Anais...** [S.l.: s.n.], 2011. v. 1, n. 1.

SELF, J. The defining characteristics of intelligent tutoring systems research: itss care, precisely. **International Journal of Artificial Intelligence in Education (IJAIED)**, [S.l.], v. 10, p. 350–364, 1998.

SHINZATO, P. Y. **Sistema de identificação de superfícies navegáveis baseado em visão computacional e redes neurais artificiais**. 2010. Tese (Doutorado em Ciência da Computação) — Universidade de São Paulo, 2010.

SHWE, M.; MIDDLETON, B.; HECKERMAN, D.; HENRION, M.; HORVITZ, E.; LEHMANN, H.; COOPER, G. A probabilistic reformulation of the Quick Medical Reference System. In: ANNUAL SYMPOSIUM ON COMPUTER APPLICATION IN MEDICAL CARE, 1990. **Proceedings...** [S.l.: s.n.], 1990. p. 790.

SILVA, E.; OLIVEIRA, A. C. d. Dicas para a Configuração de Redes Neurais. **Rio de Janeiro**, [S.l.], 2001.

SOCHER, R.; LIN, C. C.; MANNING, C.; NG, A. Y. Parsing natural scenes and natural language with recursive neural networks. In: ICML-11), 28., 2011. **Proceedings...** [S.l.: s.n.], 2011. p. 129–136.

SONKAR, S.; WATERS, A. E.; LAN, A. S.; GRIMALDI, P. J.; BARANIUK, R. G. qDKT: question-centric deep knowledge tracing. **arXiv preprint arXiv:2005.12442**, [S.l.], 2020.

SOUZA, F. A. A. d. **Análise de desempenho da rede neural artificial do tipo multilayer perceptron na era multicore**. 2012. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal do Rio Grande do Norte, 2012.

SPIEGELHALTER, D. J.; FRANKLIN, R. C.; BULL, K. Assessment, criticism and improvement of imprecise subjective probabilities for a medical expert system. In: **Machine Intelligence and Pattern Recognition**. [S.l.]: Elsevier, 1990. v. 10, p. 285–294.

TOMEK, I. et al. AN EXPERIMENT WITH THE EDITED NEAREST-NIEGHBOR RULE. , [S.l.], 1976.

VANLEHN, K. The behavior of tutoring systems. **International journal of artificial intelligence in education**, [S.l.], v. 16, n. 3, p. 227–265, 2006.

VANLEHN, K. Intelligent tutoring systems for continuous, embedded assessment. **The future of assessment: Shaping teaching and learning**, [S.l.], p. 113–138, 2008.

VANLEHN, K. The relative effectiveness of human tutoring, intelligent tutoring systems, and other tutoring systems. **Educational Psychologist**, [S.l.], v. 46, n. 4, p. 197–221, 2011.

WANG, L.; SY, A.; LIU, L.; PIECH, C. Deep knowledge tracing on programming exercises. In: FOURTH (2017) ACM CONFERENCE ON LEARNING@ SCALE, 2017. **Proceedings...** [S.l.: s.n.], 2017. p. 201–204.

WENGER, E. Artificial intelligence and tutoring system: computational and cognitive approaches to the communication of knowledge. **Califórnia: Morgan Kaufmann Publishers. Texto publicado na: Pátio-revista pedagógica Editora Artes Médicas Sul Ano**, [S.l.], v. 1, p. 19–21, 1987.

WILLIAMS, R. J.; ZIPSER, D. A learning algorithm for continually running fully recurrent neural networks. **Neural computation**, [S.l.], v. 1, n. 2, p. 270–280, 1989.

WOOLDRIDGE, J. M. **Introductory econometrics**: a modern approach. [S.l.]: Nelson Education, 2016.

WOOLF, B. P. **Building Intelligent Interactive Tutors**: student-centered strategies for revolutionizing e-learning. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

WOOLF, B. P. **Building Intelligent Interactive Tutors Student-centered strategies for revolutionizing e-learning**. [S.l.: s.n.], 2010. 480 p.

XIONG, X.; ZHAO, S.; VAN INWEGEN, E. G.; BECK, J. E. Going Deeper with Deep Knowledge Tracing. **International Educational Data Mining Society**, [S.l.], 2016.

YUDELSON, M. V.; KOEDINGER, K. R.; GORDON, G. J. Individualized bayesian knowledge tracing models. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE IN EDUCATION, 2013. **Anais...** [S.l.: s.n.], 2013. p. 171–180.