CENTRO UNIVERSITÁRIO LUTERANO DE PALMAS

Recredenciado pela Portaria Ministerial nº 1.162, de 13/10/16, D.O.U. nº 198, de 14/10/2016
AELBRA EDUCAÇÃO SUPERIOR - GRADUAÇÃO E PÓS-GRADUAÇÃO S.A.

Giovanna Biagi Filipakis Souza

CRIAÇÃO DE SEQUÊNCIAS DIDÁTICAS UTILIZANDO REDES NEURAIS E PADRÃO SCORM NO CONTEXTO DOS CONTEÚDOS DE ESTRUTURA DE DADOS

Palmas – TO

2020

Giovanna Biagi Filipakis Souza

CRIAÇÃO DE SEQUÊNCIAS DIDÁTICAS UTILIZANDO REDES NEURAIS E PADRÃO SCORM NO CONTEXTO DOS CONTEÚDOS DE ESTRUTURA DE DADOS

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e Fabiano Fagundes.

Palmas – TO

2020

Giovanna Biagi Filipakis Souza

CRIAÇÃO DE SEQUÊNCIAS DIDÁTICAS UTILIZANDO REDES NEURAIS E PADRÃO SCORM NO CONTEXTO DOS CONTEÚDOS DE ESTRUTURA DE DADOS

Trabalho de Conclusão de Curso (TCC) II elaborado e apresentado como requisito parcial para obtenção do título de bacharel em Ciência da Computação pelo Centro Universitário Luterano de Palmas (CEULP/ULBRA).

Orientador: Prof. M.e Fabiano Fagundes.

Aprovado em: _____/_____/_____

BANCA EXAMINADORA

_____

Prof. M.e Fabiano Fagundes

Orientador

Centro Universitário Luterano de Palmas – CEULP

_____

Profa. Dra. Parcilene Fernandes de Brito

Centro Universitário Luterano de Palmas – CEULP

_____

Prof. Esp. Fabio Castro Araújo

Centro Universitário Luterano de Palmas – CEULP

Palmas – TO

2020

AGRADECIMENTOS

Agradeço a toda a minha família pelo apoio e o incentivo que eles nunca deixaram faltar, mesmo sem realmente entenderem com o que eu trabalho, mas a intenção é o que vale. Agradeço aos meus pais, Cristina e Flávio, por tudo o que fizeram por mim a minha vida toda. Especialmente a minha mãe por me confortar no dia em que eu chorei na sala dela porque não aguentava mais o curso de Direito, e ao meu pai por ter dado a ideia de eu fazer Ciência da Computação, uma coisa que nunca tinha passado pela minha cabeça antes.

Agradeço a minha irmã, Isadora, por ter entendido quando eu estava muito ocupada para dar atenção pra ela. Agradeço às minhas cachorras Brisa e Mel, por serem tão fofas. E agradeço também à Raimunda, por ter sido uma constante tão presente na minha vida.

Agradeço a todos os meus professores pelas aulas, correções, explicações e festas de aniversário com salgadinhos da KiDelícia. E em especial ao meu orientador, Fabiano Fagundes, tanto pela amizade quanto pelas orientações. Ele sempre dava ótimas sugestões sobre o meu trabalho depois que a gente já tivesse ficado de conversa fiada por 20 minutos.

Agradeço aos meus colegas de trabalho da Fábrica de Software, Alexandre, Augusto, Dionnys, Gabriel e Murillo, com quem eu aprendi muito e vivi os melhores rolês. E ainda ao meu chefe da Fábrica, Jackson Gomes, porque mesmo sendo nosso chefe em momento nenhum deixou de ensinar como um professor.

Agradeço ao grupo das ItGirls, pelo privilégio de fazer parte de algo tão bonito, e espero ficar sabendo quando cada uma das meninas que ainda estão no curso se formarem.

Agradeço a pessoa que corrigiu e comentou o meu artigo submetido no 22º Encoinfo. O artigo não foi aprovado, mas os comentários e as referências foram fantásticas e também ajudaram no TCC.

E agradeço a todas as pessoas que não foram citadas, mas que me acompanharam durante esses quatro anos de curso.

“Like what you do, and then you will do your best.”

Katherine Johnson

Conforme a ABNT NBR 14724 (2011b, p. 02) a epígrafe é o “texto em que o autor apresenta uma citação, seguida de indicação de autoria, relacionada com a matéria tratada no corpo do trabalho”. Por ser uma citação, deve ser “elaborada conforme a ABNT NBR 10520 [...]. Podem também constar epígrafes nas folhas ou páginas de abertura das seções primárias”. (ABNT NBR 14724, 2011b, p. 07).

RESUMO

FILIPAKIS, Giovanna Biagi. **Criação de Sequências Didáticas Utilizando Redes Neurais e Padrão Scorm no Contexto dos Conteúdos de Estrutura de Dados.** 2020. 53 f. Trabalho de Conclusão de Curso (Graduação) – Curso de Ciência da Computação, Centro Universitário Luterano de Palmas, Palmas-TO, 2020.

Mesmo com a tecnologia tendo possibilitado a disponibilização de aulas *on-line*, conteúdos físicos não podem ser compartilhados sem que se considerem as necessidades técnicas e metodológicas exigidas. Tendo como foco conteúdos de Estrutura de Dados, são propostas algumas técnicas para suprir essas necessidades. No que diz respeito à produção de conteúdo está a criação de objetos de aprendizagem e sequências didáticas. Para criar sequências didáticas, é necessário compreender a relação entre os diferentes conteúdos de estrutura de dados, o que pode ser feito com o uso de Redes Neurais. No que tange os aspectos técnicos envolvidos em disponibilizar o curso de estrutura de dados *on-line,* a solução proposta foi a aplicação do padrão SCORM. Desse modo, o objetivo geral do presente trabalho foi de desenvolver um módulo de geração de sequências didáticas utilizando Redes Neurais e seguindo o modelo do padrão SCORM, dentro do contexto da matéria de Estrutura de Dados. Os materiais utilizados foram a API de Redes Neural Keras, que utiliza a linguagem Python e o *framework TensorFlow*, as ferramentas Microsoft PowerPoint e iSpring Free para criação dos objetos de aprendizagem, e a ferramenta Simple SCORM Packager para criação de um curso no padrão SCORM composto pelos objetos de aprendizagem. O processo de desenvolvimento envolveu a revisão de literatura, definição dos conteúdos de Estrutura de Dados que compõem a sequência, criação das sequências de teste, tratamento dessas sequências para servirem como dados de entrada da rede, implementação da Rede Neural, criação dos objetos de aprendizagem e o agrupamento desses objetos em um curso segundo o padrão SCORM, ordenados segundo a sequência gerada pela rede. O módulo gerador de sequências resultante consiste de uma função que contém o modelo da Rede Neural e que recebe como entrada uma *string* inicial. O conteúdo previsto pelo modelo a partir da *string* inicial é utilizado como entrada na iteração seguinte, até que, conteúdo a conteúdo, seja gerada a sequência. Os resultados mostram que a Rede Neural LSTM é capaz de aprender a gerar sequências didáticas, e que essas sequências podem ser estruturadas segundo o padrão SCORM.

Palavras-chave: Sequências Didáticas, Redes Neurais, SCORM, LSTM.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

IA - Inteligência Artificial

SCORM - Shareable Content Object Reference Model

LMS - Learning Management Systems

SCOs - Sharable Content Objects

MOOCs - Massive Open Online Courses

ICT - Information Communication Technology

CAM - Content Aggregation Model

XML - Extensible Markup Language

RTE - Run-Time Environment

CSE - Computer Science and Engineering

CSV - Comma-Separated Values

LSTM - Long Short Term Memory

RNN - Recurrent Neural Network

LOM - Learning Object Metadata

RRWSA - Reversed Roulette Wheel Selection Algorithm

UFSCar - Universidade Federal de São Carlos

PIF - Package Interchange File

SSP - Simple SCORM Packager

SUMÁRIO

1 INTRODUÇÃO

O início do uso da internet na educação trouxe um grande número de novas possibilidades para essa área. Uma delas é a superação das distâncias, pois alunos de todo o mundo podem ter acesso a um mesmo material, não importando quem o publicou, desde que ele tenha sido disponibilizado on-line. Outro desenvolvimento positivo é a variedade de materiais para se aprender cada assunto, que variam entre si em questões de estilo, metodologia, ordem, mídias utilizadas, níveis de complexidade e vários outros fatores como esses que permitem que o aluno escolha o que melhor se adequar a ele.

Entretanto, as pessoas que pretendem criar e disponibilizar cursos de qualidade devem se atentar ao fato de que não basta traduzir literalmente o conteúdo do meio físico para o digital. Elas devem buscar empregar teorias de aprendizado pedagógicas para melhorar o uso da tecnologia para ensino no contexto de e-learning, e focar em metodologias aplicadas ao meio digital, e não apenas transferir o material tradicional sem se preocupar com os princípios pedagógicos envolvidos (MUSA; ARMESS, 2010).

Dito isso, existem várias técnicas para se adaptar o conteúdo que será ensinado para o ambiente de *e-learning*. Uma delas é a criação de sequências didáticas, que são a ordem em que os conteúdos de determinado assunto são apresentados e, segundo Hnida, Idrissi e Bennani (2016), podem influenciar no desempenho dos alunos, dependendo da sua distribuição e objetivo. Enquanto sequências didáticas já são utilizadas no ensino presencial, o ambiente de *e-learning* oferece possibilidades adicionais como sequências personalizáveis, novos tipos de materiais que podem ser compartilhados e ficam disponíveis posteriormente às aulas, regras de sequenciamento que definem o fluxo de aprendizado e as interações que alunos podem ter com o material, entre outras. Adicionalmente, outra inovação na área da educação que é pertinente ao ambiente de *e-learning* e pode ajudar alunos e educadores é o uso de tecnologias de Inteligência Artificial (IA).

De acordo com Silveira e Vieira Junior (2019), as plataformas educacionais baseadas em IA apresentam grandes vantagens como oferecer uma grande base de dados interativa ao aluno e captar informações dessas interações para identificar as necessidades de cada um, possibilitando melhores resultados a partir do ensino personalizado. Idoeta (2020) apresenta alguns exemplos de uso de IA para educação, e entre eles o caso da *AltSchool*, uma escola da Califórnia, que usa uma plataforma adaptada de ensino para cada aluno, em que eles possuem

sua própria *playlist* de objetos de aprendizagem elaborada de acordo com as necessidades e preferências de cada um.

Dentre essas tecnologias de IA encontram-se as Redes Neurais. Um exemplo desse modelo de utilização é apresentado por Siqueira-Batista et al. (2014), que revisam os aspectos da aplicação de Redes Neurais no ensino de medicina. Eles concluem que as Redes Neurais podem ser aplicadas em várias áreas da medicina, por ter uma gama tão grande de capacidades como reconhecimento de imagens, análise de espectros e tomada de decisão em problemas complexos.

Para o contexto do presente trabalho, considera-se que o desafio de gerar sequências envolve o reconhecimento de padrões que relacionam a ordem em que os conteúdos são apresentados e o desempenho dos alunos durante os processos avaliativos. No caso de uma grande quantidade de conteúdos e alunos envolvidos, essa é uma tarefa de alta complexidade. O funcionamento de Redes Neurais oferece vantagens como trabalhar com eventos que acontecem em tempo real e continuamente geram novos casos de entrada, lidar com grandes bases de dados, possuir a capacidade de aprender sozinhas, e de encontrar padrões de relacionamentos não-lineares. Também é possível tratar as variáveis de entrada para que elas possam ser categóricas, ao invés de apenas numéricas. Por isso, o uso de Redes Neurais se torna interessante para encontrar os relacionamentos necessários para a criação de sequências didáticas.

Outro aspecto importante referente à disponibilização de conteúdos on-line é justamente quais as necessidades técnicas devem ser atendidas para que eles se tornem acessíveis para outras pessoas. Para isso, a proposta deste trabalho também considera a utilização do padrão SCORM (Shareable Content Object Reference Model) para definir essas necessidades técnicas. Wang et al. (2004) apresentam SCORM como o padrão mais popular para a consistência do formato do curso entre os diferentes sistemas de e-learning. Dentre as capacidades de configuração que ele oferece estão definições de Sequenciamento e Navegação entre os conteúdos do curso e organização hierárquica desses conteúdos. O padrão SCORM também possibilita padronizar os assuntos de uma determinada matéria entre si, para que todos eles sejam SCOs (Sharable Content Objects) com a mesma estrutura básica.

Com isso em mente, foi proposto o problema de pesquisa de como modelar uma Rede Neural que gere Sequências Didáticas seguindo o padrão SCORM e utilizando os conteúdos da matéria de Estrutura de Dados. Esse problema gerou a hipótese de que os conteúdos da matéria de Estrutura de Dados podem ser tratados para servir de entrada para uma Rede

Neural que irá aprender a determinar em qual ordem esses conteúdos podem ser aplicados em aula para gerar uma sequência didática.

Portanto, este trabalho teve como objetivo geral implementar um módulo gerador de sequências didáticas para objetos de aprendizagem baseados no padrão SCORM utilizando Redes Neurais e aplicado aos conteúdos de Estrutura de Dados. Como objetivos específicos têm-se modelar uma Rede Neural para gerar Sequências Didática dos conteúdos de Estrutura de Dados, planejar o tratamento necessário para utilizar variáveis categóricas como entrada, implementar a Rede Neural, definir as sub especificações do padrão SCORM e gerar uma sequência didática integrada ao padrão SCORM.

2 REFERENCIAL TEÓRICO

2.1 E-learning e Objetos de Aprendizagem

O crescimento da quantidade de conteúdos disponíveis na internet que tem como objetivo a educação aconteceu de maneira muito rápida nos últimos anos, assim como o número de MOOCs (*Massive Open Online Courses*) que disponibilizam desde pequenas aulas sobre atividades do dia-a-dia até cursos de formação completos e certificados. Segundo Shah (2018), a criação dos primeiros MOOCs data do fim do ano de 2011 e em seis anos mais de 800 universidades de todo o mundo lançaram suas plataformas, assim como as empresas de tecnologia, acumulando mais de 9000 MOOCs até 2017.

Em grande parte, a popularização de métodos não tradicionais de aprendizado como o *e-learning* também ocorre em função de que a nova geração de alunos que nasceram no final da década de 90 sempre viveram em um mundo digitalizado. Por isso eles mesmos estão muito mais acostumados com a tecnologia e se adaptam mais rapidamente às mudanças do que qualquer outra geração. Ainda assim, há vários outros fatores que podem influenciar na receptividade dos alunos em relação a esses métodos. Brumini et al. (2014) concluem que idade, ano de estudo, frequência de uso do Facebook, uso da Internet na educação e número de cursos de *e-learning* influenciam na atitude positiva em relação ao *e-learning* entre estudantes de odontologia da Croácia, que compreendem o grupo de participantes da pesquisa. Eles também afirmam que caso professores queiram integrar técnicas de aprendizagem aprimoradas por ICT (*Information Communication Technology*), devem fazer isso nos anos iniciais do período do curso, começando gradualmente com ferramentas menos exigentes, a fim de familiarizar os alunos com as possibilidades de ICT na aprendizagem.

Uma grande vantagem do uso da tecnologia e MOOCs para o ensino é a flexibilidade que essas ferramentas oferecem. Elas possibilitam que sejam utilizados textos e imagens aos quais não se teria acesso fisicamente, vídeos, áudios, atividades interativas, jogos, e inúmeras outras técnicas. Assim, os professores não estão limitados ao uso de materiais tradicionais e isso oferece mais oportunidades aos alunos. Segundo Vaughn e Baker (2001) se os professores usarem uma variedade de métodos e estilos de ensino, os alunos são expostos a ambas maneiras familiares e não familiares de aprender, que oferecem tanto conforto e tensão durante o processo, dando aos alunos várias maneiras de obterem sucesso.

Outra vantagem que deve ser comentada está relacionada à capacidade que os MOOCs têm de armazenar os dados das interações entre a plataforma e os alunos, como o perfil dos alunos, a quantidade de cliques que eles dão dentro de uma página ou o tempo que ele gasta para realizar uma determinada atividade. Mesmo que o aluno não siga a sequência de

atividades sugerida, será mantido registro de suas ações. Desse modo, ele tem mais liberdade de adaptar o material à sua preferência, o que normalmente não ocorre em salas de aula tradicionais, e a plataforma ainda recebe dados que poderiam vir a ajudar a caracterizar diferentes grupos de alunos de acordo com seus hábitos de aprendizagem e distinguir as intenções dos alunos ao utilizarem MOOCs para entender melhor o relacionamento entre comportamentos e resultados de aprendizagem (CHEN et al., 2020).

Esse fenômeno da digitalização da educação tem como consequências quebrar barreiras geográficas, ajudar a disseminar e democratizar o conhecimento e flexibilizar os métodos de ensino-aprendizagem. E para possibilitar que os conhecimentos de sala de aula sejam aplicados no ambiente de *e-learning*, foram criadas maneiras de apresentar esses conhecimentos.

Os objetos de aprendizagem são uma das formas de representar o conhecimento que serão apresentadas em um ambiente de *e-learning*. Eles podem ser definidos como qualquer entidade, digital ou não digital, que pode ser usada, reutilizada ou referenciada durante o aprendizado suportado pela tecnologia (DRAY, 2005). Porém, objetos de aprendizagem não podem ser simplesmente disponibilizados *on-line* sem que haja cuidado em como eles são criados e organizados. Segundo Ghisi (2016), a criação de um objeto de aprendizagem é uma tarefa que requer trabalho, porque tão importante quanto o conhecimento das ferramentas utilizadas para o desenvolvimento, também é necessário saber como ocorre a construção do conhecimento.

Segundo Ritzhaupt (2010) e Watson (2010), os objetos de aprendizagem utilizados no ambiente de *e-learning* podem incluir imagens digitais, *feeds* de vídeo, animações, conteúdos textuais e até mesmo exercícios como os de múltipla escolha para testar o entendimento dos alunos. Porém, eles não são apresentados de forma aleatória. Eles são dispostos sob uma ou várias sequências, chamadas de sequências didáticas, que são indicadas para que os alunos as sigam durante o aprendizado.

2.2 Sequências Didáticas

Uma organização do conhecimento apresentado ao aluno usada no ambiente tradicional de ensino e que pode ser adaptada ao processo de ensino-aprendizagem do ambiente digital é a criação de sequências didáticas. Vale ressaltar que o termo “sequências didáticas” é o mais comumente utilizado, porém podem ser encontrados trabalhos com as expressões “sequências de ensino”, “sequências de aprendizado”, "sequências de estudo" e

"plano de estudos", e termos em inglês como "*learning sequence*", "*learning-teaching sequence*" e "*study plan*".

As sequências didáticas definem a ordem em que os conteúdos, ou objetos de aprendizagem, de um determinado assunto devem ser aplicados para possibilitar um melhor resultado. Musa e Armess (2010) afirmam que é muito importante que os materiais de aprendizado sejam planejados em uma sequência que esteja de acordo com o estilo de aprendizagem do aluno.

Essa mesma conclusão é embasada por Hnida, Idrissi e Bennani (2016), que apresentam a proposta de sequências didáticas baseadas em ontologia de domínio, ou seja, no conjunto de conceitos dentro do contexto de uma estrutura curricular. Eles afirmam que, nesse modelo, o sistema poderia inferir qual conteúdo seria apresentado e de que maneira, com relação a qual abordagem educacional, para alcançar quais resultados de aprendizagem, e com qual padrão de desempenho.

Existem várias maneiras de se definir uma sequência didática e elas irão ajudar a estabelecer os parâmetros que irão definir qual a melhor ordem em que se apresentar os conteúdos. De acordo com Ghirardini (2011), alguns dos métodos de sequenciamento que podem ser aplicados são:

- Pré-requisito: usa um aprendizado hierárquico de objetivos, ensinando primeiro as habilidades que parecem ser pré-requisitos para todas as outras habilidades;
- Curso orientado ao trabalho: o conteúdo é organizado para seguir a ordem das ações no ambiente real de trabalho;
- Curso não orientado ao trabalho: conteúdos são organizados de acordo com suas conexões estruturais, como:
    - descrever as características de um conceito ou classe antes dos membros que a compõe;
    - apresentar exemplos antes de conceitos;
    - começar com conceitos simples e concretos, antes de passar para os mais complexos e abstratos;
- Perfil dos alunos: se tiver um conhecimento extensivo dos perfis dos alunos, apresentar primeiramente os conceitos mais familiares a eles, antes dos que estiverem distantes de suas experiências de vida;
- Princípio de Zoom: apresentar primeiro uma visão geral dos assuntos de um curso, antes de focar em tópicos específicos;

- Princípio de Espiral: rever conceitos básicos e repetidamente trabalhar em cima deles, até que os alunos o compreendam completamente.

A escolha de qual método de sequenciamento deve ser utilizado vai depender da necessidade de cada curso e do conhecimento que se tem do público alvo, ou dos alunos. Por exemplo, se for um curso específico de especialização para os funcionários de uma empresa para que eles desenvolvam uma nova habilidade, o método escolhido pode ser o orientado ao trabalho. Já no caso de o público ser o de alunos com nível muito baixo de conhecimento no assunto a ser ensinado, o melhor método pode ser o não orientado ao trabalho, utilizando a abordagem de apresentar os conceitos mais simples antes dos mais complexos. Adicionalmente, não é obrigatório se limitar a utilizar apenas um método de cada vez, mas combinar dois ou mais métodos, sempre com o objetivo de facilitar o aprendizado e levando em conta o contexto do curso.

Definir o método de sequenciamento é uma das fases para se apresentar objetos de aprendizagem de maneira compreensiva, mas ainda é necessário possibilitar que esses conteúdos sejam disponibilizados *on-line*. Segundo Ghirardini (2011), para ser carregado adequadamente e acessível a partir de um LMS, aulas e cursos de *e-learning* devem estar em conformidade com um conjunto de padrões técnicos e instrucionais. Atualmente, um dos modelos líderes que oferece essa capacidade de integração com os LMS é o padrão SCORM.

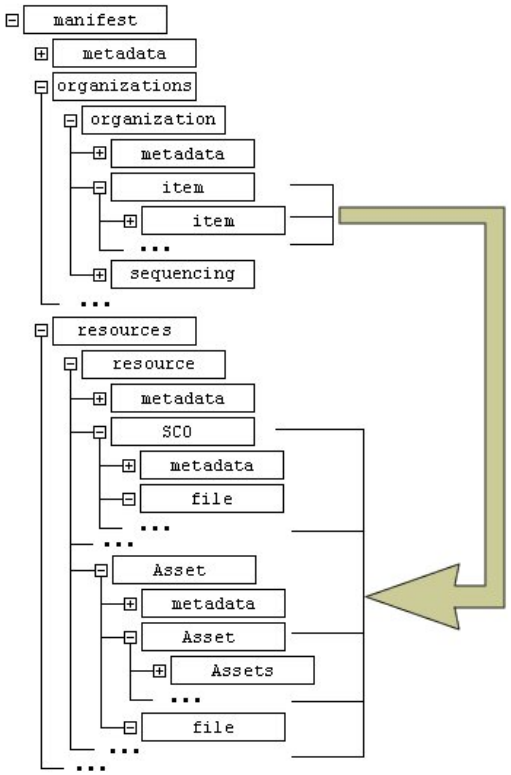
2.3 Sharable Content Object Reference Model (SCORM)

O SCORM, originado em 1997, foi promovido pelo Departamento de Defesa dos Estados Unidos (*Department of Defense - DOD*), e tem como objetivo estabelecer um mecanismo de reutilização e compartilhamento de livros didáticos, encurtar os materiais do ciclo de desenvolvimento e reduzir custos, fazendo com que os materiais adicionados circulem livremente na plataforma de aprendizado (JIUGEN et al., 2016).

De acordo com Rustici Software (2020), SCORM é um conjunto de padrões técnicos para produtos de software de *e-learning*, de natureza puramente técnica, e que estabelecem como conteúdos de aprendizagem *on-line* e Sistemas de Gestão de Aprendizagem (LMS) devem se comunicar uns com os outros. Isso quer dizer que se um LMS é compatível com SCORM, ele pode reproduzir qualquer conteúdo compatível com SCORM, e qualquer conteúdo compatível com SCORM pode ser reproduzido em qualquer LMS compatível com SCORM. Com esses padrões técnicos, o desenvolvedor do conteúdo de aprendizagem deve criar SCOs, que são caracterizados como a menor parte indivisível, que ainda seja independente e reutilizável, do conteúdo.

Segundo Kazanidis e Satratzemi (2009) e Rustici Software (2020), o SCORM é composto por três componentes. O primeiro é o Modelo de Agregação de Conteúdo (*Content Aggregation Model - CAM*), que especifica que o conteúdo do SCORM deve ser armazenado em um arquivo com com extensão *ZIP*.

Figura 1 - Estrutura do arquivo *manifest*



Fonte: SAVIC; KONJOVIC (2009, p.350)

Ele também possui um arquivo XML (*Extensible Markup Language*) chamado *manifest* que irá conter as informações necessárias para que os LMS saibam como apresentar o conteúdo do SCORM, como, por exemplo, metadados dos SCOs, os recursos e uma "árvore de atividade" que os organiza em uma estrutura de árvore. Segundo Savic e Konjovic (2009), os recursos são organizados dentro de tags `<organization>`. Os recursos são definidos dentro das tags `<resource>`. A organização contém uma hierarquia de tags `<item>`. A tag `<item>` faz referência à tag `<resource>` apropriada. Cada tag pode conter uma tag filha `<metadata>` que define os metadados do item.

O segundo componente se chama Ambiente de Execução (*Run-Time Environment - RTE*) que determina que o LMS deve lançar o conteúdo em uma página web, criando uma nova janela ou um *frameset*, a maneira em que o LMS vai se comunicar com o material, e também é responsável por registrar informações sobre as interações do usuário com o

conteúdo. Informações como o status da SCO (concluído, aprovado, reprovado, etc.), a pontuação que o aluno obteve, um marcador para rastrear a localização do aluno e a quantidade total de tempo que o aluno passou na SCO.

Por último, o terceiro componente é o de Sequenciamento e Navegação, que contém todas as regras a respeito de como um usuário deve navegar pelo curso e em como essa ação ocorre. As regras de sequenciamento ajudam o autor do conteúdo a controlar, por exemplo, qual a ordem em que os SCOs são apresentados, apresentar novamente um SCO anterior que não tenha sido completamente aprendido, ou fazer com que um SCO valha mais pontos que outro. Esse componente é o que está mais integrado com o foco deste trabalho de criar sequências didáticas.

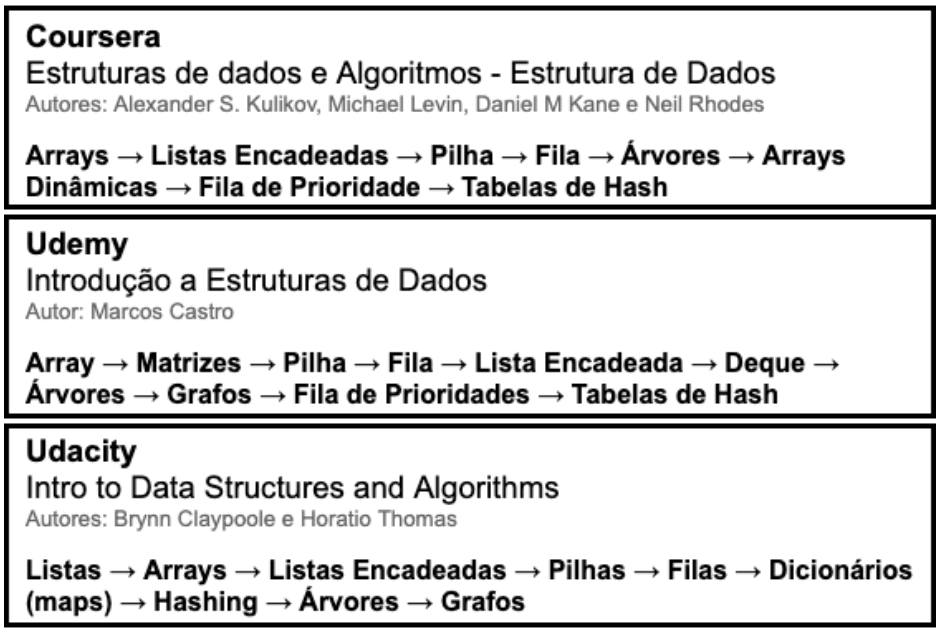
Tendo claro o conceito de sequências didáticas e da tecnologia utilizada para disponibilizar os SCOs *on-line*, ainda é necessário elaborar qual será o conteúdo desses SCOs. O presente trabalho considera o contexto dos conteúdos da matéria de Estrutura de Dados.

2.4 Visão geral da matéria de Estruturas de Dados

Learnbay.co (2007) divide os tipos de estruturas de dados em duas categorias. A primeira é a das estruturas de dados primitivas, que lidam com pequenas quantidades de tipos de dados como *boolean*, *char*, *float*, *integer*, entre outros. Já as estruturas abstratas lidam com grandes quantidades de dados complexos e o tipo de estrutura influencia em como os usuários de determinado sistema interagem com os dados.

De modo geral, a matéria de estruturas de dados compreende a lógica organizacional e métodos de armazenamento físico de dados com os quais o computador lida. Entretanto, por se tratar de conceitos altamente abstratos que dependem de uma lógica restrita, alunos costumam ter problemas para entendê-los (CAO; CAO, 2011). Ela é uma das matérias comumente oferecidas em cursos relacionados à tecnologia da informação, ciência e engenharia da computação, ciência de dados, entre outros. Naragund et al. (2016) apresentam Estrutura de Dados como um dos pilares da Ciência e Engenharia da Computação (*Computer Science and Engineering - CSE*), junto com linguagens de programação, fundamentos de sistemas, gerenciamento de informação e fundamentos de desenvolvimento de software. Como tal, é importante que os alunos tenham um bom nível de compreensão sobre esse assunto, e incluída na metodologia de ensino aplicada há a definição da ordem em que os conteúdos são ensinados.

Figura 2 - Comparação entre cursos *on-line* referentes à matéria de Estrutura de Dados



As informações dos cursos apresentados na Figura 1 foram retiradas de suas respectivas plataformas de ensino online (KULIKOV et al., 2020), (CASTRO, 2016), (CLAYPOOLE; THOMAS, 2020). Com elas, é possível notar que todos os cursos apresentam vários conteúdos em comum, mas que a ordem em que eles são apresentados aos alunos não é fixa, podendo variar dependendo de quem ensina e, por existirem tantas possibilidades, é difícil determinar qual delas deve ser aplicada em um determinado contexto.

A proposta de criar sequências didáticas com conteúdos de Estrutura de Dados pode ajudar no entendimento dos alunos sobre esse assunto, mas é também um desafio complexo. Não só os conteúdos podem ser apresentados em ordens diferentes, cada um ainda poderia ser dividido em parte prática e teórica. Uma tecnologia capaz de aprender, modelar relacionamentos complexos e gerar previsão de sequências são as Redes Neurais.

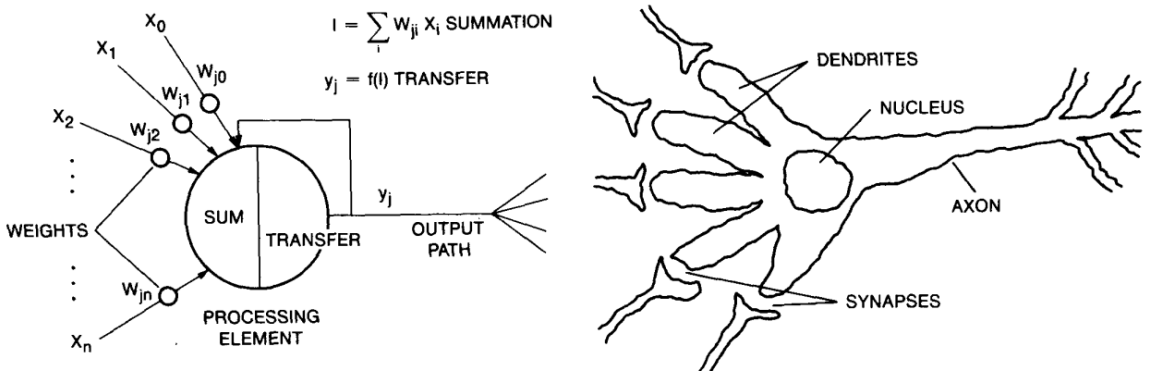
2.5 Redes Neurais

Redes Neurais são definidas como um sistema de hardware ou software desenvolvido com base no funcionamento dos neurônios no cérebro humano e no sistema nervoso. Elas são uma tecnologia da área da Inteligência Artificial (IA) que, por sua vez, é um campo próspero com muitas aplicações práticas e tópicos de pesquisa ativos, capaz de encontrar soluções para automatizar o trabalho de rotina, compreender fala e imagens, fazer diagnósticos de doenças e apoiar a pesquisa científica básica (GOODFELLOW et al., 2016).

Segundo Illingworth (1989) as primeiras pesquisas referentes ao que viriam a se tornar tecnologias de Redes Neurais e inteligência artificial datam do ano de 1943, quando

McCulloch e Pitts trabalhavam nos primeiros modelos de neurônios biológicos, para que mais tarde eles fossem interpretados para a criação de neurônios artificiais.

Figura 3 - Comparação entre os modelos de neurônio artificial e biológico



Fonte: ILLINGWORTH; WILLIAM (1989, p.1140)

O neurônio artificial é o componente mais básico de qualquer Rede Neural. Ele é composto por uma quantidade variável de dados de entrada, representados na Figura 3 pelos valores de x_0 a x_n , os pesos, representados por w_{j0} a w_{jn} , e a célula de processamento que recebe esses valores e aplica as equações necessárias para gerar uma saída y_j.

2.5.1 Perceptron

Existem vários tipos de arquiteturas de Redes Neurais diferentes. Uma das arquiteturas mais comuns e que mais facilmente exemplifica seus componentes básicos se chama *Perceptron*. Nessa arquitetura, há a camada de *inputs*, apenas uma unidade de processamento e o um valor de *output*. Seguindo o modelo de neurônio artificial da Figura 2, cada valor de entrada x_n possui um peso associado w_n. O processamento realizado no neurônio sobre esses valores consiste em encontrar o valor y em função de x, que é a soma do produto de cada valor de *input* pelo seu peso.

$$y(x) = \sum_{i=0}^{n} w_i x_i$$

Se o resultado dessa equação for maior ou igual a um valor mínimo chamado de limiar, então o neurônio é ativado. Se o valor for menor que o limiar, o neurônio permanece inativo (ALSMADI et al., 2009).

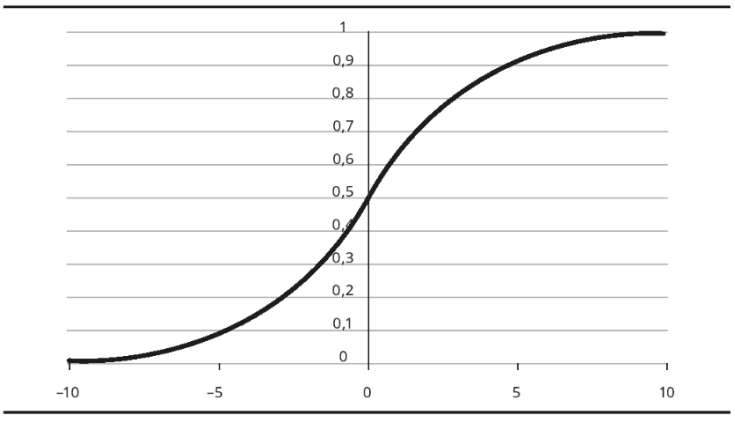
Esse limiar vai depender do tipo de cálculo que for realizado na célula de processamento. O neurônio pode realizar apenas a aplicação na equação da soma do produto dos *inputs* e pesos, ou pode também adicionar um *bias*, ou viés, que é um valor que vai servir para ajustar o valor de saída, sem se multiplicar com nenhuma entrada (MEDEIROS, 2018).

$$y(x) = \sum_{i=0}^{n} w_i x_i + b$$

Tendo o limiar definido *Z*, qualquer *output* acima de *Z* é igual a 1, e qualquer *output* abaixo é igual a 0. Se fosse adicionado um *bias* de -2, então os *outputs* positivos devem ser maiores que o *Z-2*, e os negativos são menores ou iguais a *Z-2*. Desse modo, é mais fácil para o *Perceptron* gerar uma saída que ative o neurônio.

Até o momento, o cálculo realizado pela célula de processamento funciona apenas com entradas que possam ser separadas linearmente. Segundo Medeiros (2018), isso significa que as entradas devem poder ser classificadas em classes distintas, de tal modo que essas classes possam ser separadas por uma reta em um gráfico. Para que se obtenha mais capacidade de classificação de entradas, são utilizados outros tipos de equações chamadas de funções de ativação. Guo et al. (2019) afirmam que a função de ativação refere-se a como reter e mapear as características dos neurônios ativados por meio funções não lineares, que é a chave para resolver problemas não lineares em Redes Neurais, e também discorrem sobre algumas dessas funções.

Figura 4 - Função Sigmoide



Fonte: MEDEIROS; LUCIANO FRONTINO DE (2018, p.153)

A função Sigmoide, por exemplo, define que o neurônio é ativado desde que o valor seja maior que o limiar, e não ativo se for menor, mas não classifica esses valores especificamente como 1 ou 0, e sim permite que eles sejam valores contínuos entre esses dois limites. O resultado dessa operação é o valor de *output* da rede.

A primeira iteração da rede é feita utilizando-se pesos aleatórios, então o resultado costuma ser longe do esperado. Para melhorar esses resultados, o próximo passo é o filtro adaptativo descrito por Haykin (1999). Ele consiste de dois passos:

- Processo de filtragem, que envolve o cálculo de dois valores:
  - o valor de *output* y_j
  - o valor de erro denotado como e_j , também conhecido como gradiente, obtido à partir da comparação entre o valor y_j e o resultado desejado d_j
- Processo de adaptação, que consiste no ajuste dos valores dos pesos w_{j0} a w_{jn} de acordo com o valor e_j

A repetição desse processo possibilita que os pesos sejam ajustados e os resultados das iterações seguintes sejam cada vez mais próximos dos resultados esperados, ou seja, a Rede Neural está "aprendendo" a encontrar o resultado correto.

O *Perceptron* foi um dos primeiros passos que levaram ao surgimento do campo que hoje se entende como Aprendizado de Máquina, mas por ter uma arquitetura tão simples, há várias tarefas que o *Perceptron* não é capaz de executar. Por isso, no decorrer do tempo uma grande quantidade de outras arquiteturas de Redes Neurais foram propostas para resolver diferentes desafios. Uma dessas arquiteturas se chama *Long Short Term Memory* (LSTM).

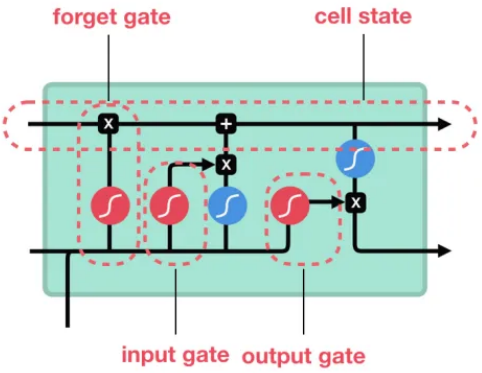
2.5.2 Long Short Term Memory (LSTM)

LSTM são um tipo de Rede Neural Recorrente (*Recurrent Neural Network - RNN*), que por sua vez podem ser descritas como Redes Neurais com um ou mais loops de *feedback* (HAYKIN, 1999), o que torna possível que, por exemplo, valores de *output* sejam enviados de volta para a camada de *input* da rede, oferecendo informações de *feedback*. Essa interação configura um tipo de memória de curto prazo, uma das vantagens da RNN em relação a outras arquiteturas, que possibilita que resultados anteriores sejam considerados no momento da tomada de decisão da rede.

Ainda assim, as RNNs possuem algumas limitações, como o problema dos gradientes instáveis (*unstable gradientes*). Isso ocorre quando o valor do gradiente é desproporcionalmente grande ou pequeno. Se ele for muito pequeno, os pesos que são atualizados com base no gradiente irão sofrer uma alteração quase insignificante, de modo que a Rede Neural para de "aprender". Isto é, os pesos se aproximam muito lentamente, ou não se aproximam, do valor ideal. Semelhantemente, quando o gradiente é muito grande os pesos também crescem de maneira exagerada, e se afastam do valor ideal. O primeiro caso é chamado de *vanishing gradiente*, e o segundo de *exploding gradient*.

Hochreiter e Schmidhuber (1997) consideram essa limitação das RNNs e apresentam a arquitetura LSTM como uma solução. Com ela, é possível criar Redes Neurais mais complexas enquanto se mantém a estabilidade do valor de gradiente.

Figura 5 - Célula LSTM

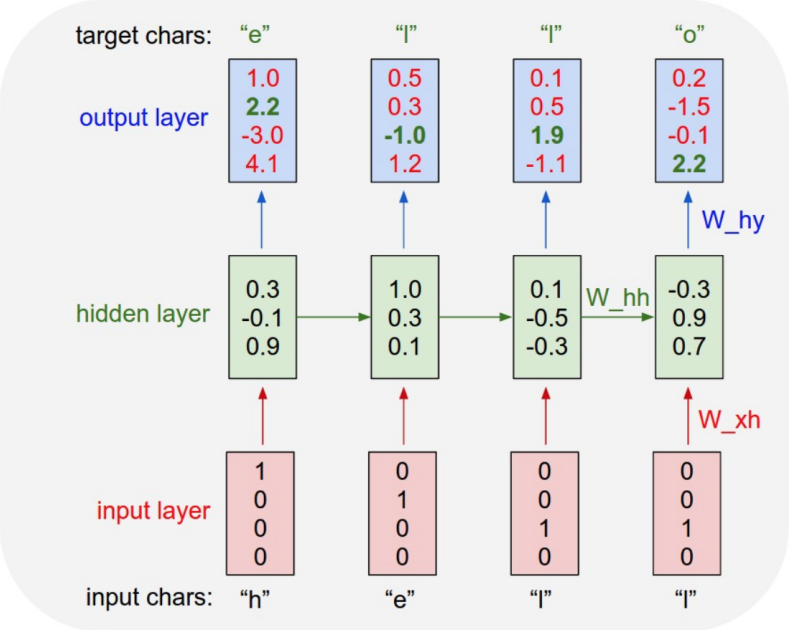


Fonte: DATA SCIENCE ACADEMY (2019)

O grande diferencial da LSTM é a configuração da sua célula de memória. Ela possui três estruturas chamadas de portões, ou *gates*, que são o *forget gate*, o *input gate* e o *output gate*. As operações realizadas dentro de um dos portões permitem que a célula controle quais valores de gradiente devem afetar o resto da Rede Neural e quais são desproporcionais e devem ser eliminados, ou esquecidos, impedindo que o problema com os gradientes instáveis aconteça. Esses portões atualizam o valor *cell state*, que é repassado a cada *loop*, e configura a memória de longo prazo da rede.

Para treinar uma Rede Neural LSTM, é necessário preparar grupos de exemplos. O primeiro são os exemplos de *input*, e o segundo são os exemplos de resultados esperados, ou *targets*, sendo que para cada *input*, há um resultado esperado correspondente. Também é necessário informar para a rede qual o conjunto de possíveis itens distintos existentes no contexto em questão. Como exemplo, a imagem abaixo apresenta uma situação em que a rede deve aprender a escrever a palavra *hello*.

Figura 6 - Modelos de linguagem a nível de caractere



Fonte: KARPATHY (2015)

Nesse contexto, o conjunto de possibilidades distintas existentes, ou vocabulário, é formado pelos caracteres *h*, *e*, *l* e *o*, o exemplo de *input* será *hell* e o *target* será *ello*. Desse modo, o primeiro valor do exemplo de *input h* tem como *target* o primeiro valor do conjunto de resultados esperados, que é o caractere *e*, e assim por diante. Isso configura um estilo de aprendizagem supervisionada.

Segundo James et al. (2017), casos de aprendizagem supervisionada ocorrem quando, para cada entrada x_i , sendo $i = 1, \ldots, n$, há uma resposta associada y_i. A relação entre essa resposta e os valores previstos é utilizada para ajustar o modelo, possibilitando que haja melhores previsões no futuro.

As aplicações de Redes Neurais LSTM incluem análise de sentimentos, tradução de textos, processamento de linguagem natural, criação de música e, no que diz respeito ao presente trabalho, geração de sequências a partir de conteúdos educacionais da matéria de Estrutura de Dados.

2.6 Trabalhos relacionados

2.6.1 Uma Estrutura para Definição de Seqüências de Estudos Baseada na Técnica Ant System

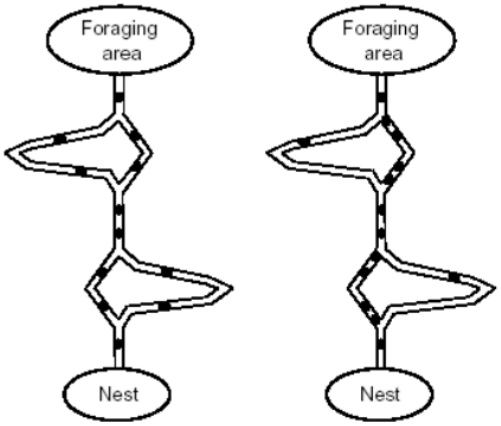
O trabalho desenvolvido por Brito et al. (2002) teve como objetivo verificar a viabilidade de se aplicar a técnica computacional *Ant System* no contexto de um sistema não-determinístico. Mais especificamente, o *Ant System* é utilizado para o desenvolvimento de

um sistema que tem o objetivo de definir sequências didáticas para o ensino da matéria de lógica de predicados.

Para isso, são utilizados agentes inteligentes, cujas funções podem ser definidas como: percepção das condições dinâmicas de um ambiente, ação de modo a afetar as condições do ambiente e raciocínio para interpretar percepções, realizar inferências e determinar ações (HAYES-ROT, 1995 apud BRITO et al., 2002). Utilizando essa definição, e ainda as descrições de Russel e Norvig (1995) e Maes (1994), Brito et al. (2002) definem agentes inteligentes como entidades autônomas que são especialistas na execução de uma determinada tarefa, tendo a capacidade de perceber o ambiente em que atuam, tomar decisões sobre informações obtidas deste ambiente e executam alguma tarefa (ação) como resultado.

Essas tomadas de decisões são feitas seguindo o modelo *Ant System* e os *Ant algorithms* que, por sua vez, são uma abordagem que utiliza vários agentes para solucionar problemas de otimização combinatorial, alocação e definição de rotas (BRITO et al., 2002 apud DORIGO, 1996), sendo que cada agente é a representação virtual de uma formiga. Eles se baseiam no comportamento que formigas reais apresentam ao buscarem o melhor caminho entre uma fonte de alimento e o formigueiro.

Figura 7 - Representação do *Ant System*

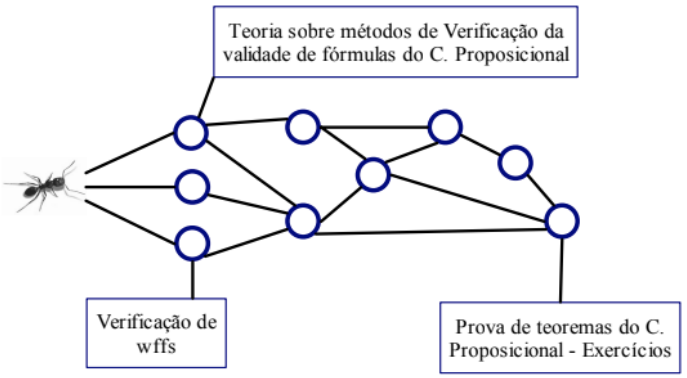


Fonte: BRITO et al. (2002, p.249)

Enquanto caminham, elas liberam uma substância chamada feromônio e as formigas tendem a escolher caminhos com a maior concentração de feromônios, sendo que essa substância evapora com o tempo. Desse modo, o caminho mais curto é aquele em que os feromônios têm menos tempo para evaporar, a concentração permanece maior do que em caminhos mais longos, e tende a ser mais selecionado pelas formigas. Computacionalmente,

os problemas são representados como grafos e as arestas têm como peso a quantidade de feromônio acumulada pela passagem das formigas por elas.

Figura 8 - Grafo com a sequência de passos para



Fonte: BRITO et al. (2002, p.250)

Na Figura 5 é apresentado um grafo para o domínio "Cálculo Proposicional" – um tópico da disciplina Lógica de Predicados. O usuário, representado pela formiga, percorre o caminho no grafo, em que cada nó representa um conteúdo específico do tópico.

Como resultado da proposta, observou-se que, além de que um problema tão subjetivo como o estabelecimento de caminhos de aprendizagem era possível de ser computado, o trabalho conjunto entre algumas técnicas de Inteligência Artificial pode trazer resultados muitas vezes melhores do que sua aplicação individual.

2.6.2 AntStudy: Proposta de definição de sequências de estudo utilizando Ant System

No trabalho desenvolvido por Araújo e Fagundes (2011) é apresentada a proposta do AntStudy, um algoritmo para a definição de sequências de estudo, também baseado no *Ant System*.

Para gerar os conteúdos que serão ordenados para formar a sequência de estudos, os autores utilizaram o conceito de *Learning Object Metadata* (LOM) que é uma estrutura de metadados referentes a um objeto de aprendizagem, que tem a finalidade de descrever características importantes sobre ele (ARAÚJO; FAGUNDES, 2011 apud IEEE, 2002, p. 5).

Outro fator importante considerado para a definição de sequências de estudo são as diferentes características de cada usuário do sistema que está sendo desenvolvido. Por isso, tanto os dados do LOM como os dados do modelo de usuário foram utilizados durante o processamento para definir as sequências de estudos a partir de um algoritmo baseado no comportamento das formigas.

Araújo e Fagundes (2011) descrevem a origem do *Ant System* e como ele deriva do comportamento de formigas, similar ao que foi explicado no trabalho de Brito et al. (2002) na seção 2.6.1. Adicionalmente, eles exemplificam os *Ant algorithms* no formato de pseudocódigo.

Figura 9 - Pseudocódigo do algoritmo *Ant System*

Pseudocódigo do algoritmo *Ant System*

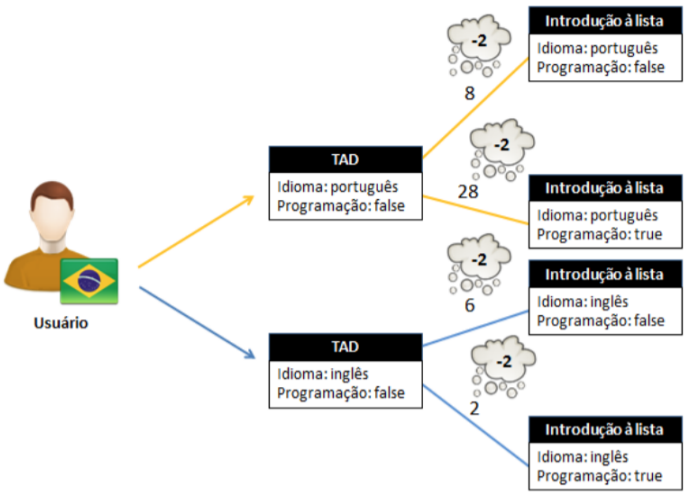
```
Inicializar parâmetros;
Repetir (condição de parada)
{
        Construir soluções;
        //atualização do feromônio nas próximas 2 etapas
        Evaporação do feromônio;
        Atualização do feromônio pelas formigas;
}
```

Fonte: ARAÚJO; FAGUNDES (2011, p.103)

O primeiro passo consiste em inicializar alguns parâmetros, como o grafo a ser percorrido e a quantidade de agentes que irão participar do processo. E o segundo passo diz respeito à atualização da trilha de feromônios e é dividido em duas partes. Na primeira, o algoritmo considera uma taxa de evaporação, possibilitando que soluções ruins tenham cada vez menos probabilidade de serem percorridas por formigas artificiais durante a execução, e na segunda, as taxas de feromônio presentes na trilha criada por um agente são modificadas.

O trabalho foi desenvolvido utilizando os conteúdos da matéria de Estrutura de Dados para a criação dos objetos de aprendizagem.

Figura 10 - Modelo de funcionamento do AntStudy



Fonte: ARAÚJO; FAGUNDES (2011, p.106)

Na Figura 7 os objetos de aprendizagem estão dispostos no formato de grafo, junto com os metadados referentes a eles. Também estão representados na imagem o usuário, a quantidade de feromônios nas arestas do grafo, e a taxa de evaporação desses feromônios em cada uma delas.

A proposta de um algoritmo baseado no *Ant System* para a definição de sequências de estudo a partir de informações do LOM e de modelos de usuário é exemplo de uma forma de aplicar um algoritmo inicialmente voltado para problemas objetivos, numéricos, também em problemas que envolvam certa subjetividade, como é o estabelecimento de uma sequência de estudos.

2.6.3 Applying learning styles to SCORM compliant courses

Kazanidis e Satratzemi (2009) desenvolveram um LMS compatível com modelo SCORM, chamado ProPer, que fornece a possibilidade de adaptação às necessidades do usuário, usando os recursos e especificações do sistema SCORM.

Para entender melhor quais seriam essas necessidades dos usuários, os autores trabalham com teorias de estilos de aprendizagem (*learning styles*). São apresentados modelos de diferentes autores que definem estilos de aprendizagem e como os alunos seriam divididos entre essas categorias, sendo que todas essas teorias diferenciam os alunos de acordo com os fatores que influenciam suas experiências de aprendizagem e visam ajudá-los a alcançarem os melhores resultados possíveis no processo de aprendizagem.

O objetivo deste trabalho é desenvolver um *framework* que permita o desenvolvimento e o *design* de cursos compatíveis com o padrão SCORM, com foco na adaptação com relação ao estilo de aprendizagem do usuário. Desse modo, o *framework* proposto consiste de duas principais camadas: a Camada Pedagógica de Projeto e a Camada Técnica. A primeira camada é onde o educador define os objetivos do curso, os parâmetros para medição de desempenho, os conteúdos que serão apresentados, as estratégias de ensino utilizadas, e qual o estilo de aprendizagem que será aplicado. Já a segunda camada é onde são definidos os recursos educacionais utilizados, os SCOs são construídos e o arquivo de manifesto do modelo SCORM, que é o arquivo XML que inclui as informações essenciais sobre a estrutura e sequência do curso e o sequenciamento.

A proposta do projeto foi avaliada por meio de um estudo de caso em que o LMS ProPer foi utilizado na construção de um curso. Trinta e um alunos estudaram o curso e foram questionados sobre a experiência de aprendizagem. Os alunos concordaram que a adaptação

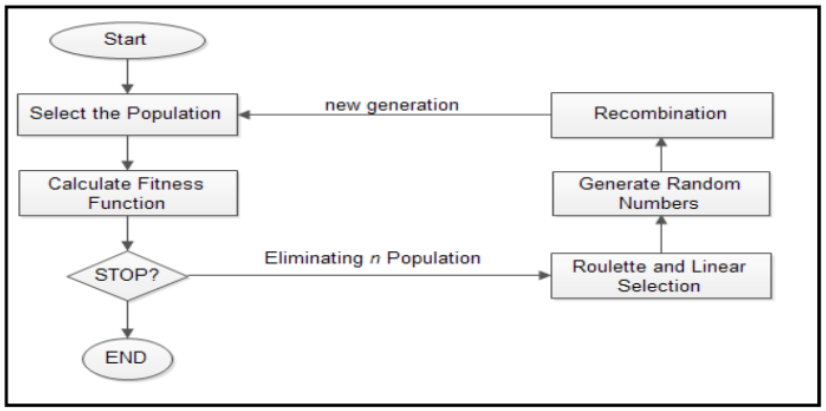
do estilo de aprendizagem foi útil e bem aplicada e apenas um pediu uma apresentação diferente do conteúdo.

2.6.4 Personalizing E-Learning Curriculum Using Reversed Roulette Wheel Selection Algorithm

Ballera et al. (2014) propõem a aplicação do algoritmo chamado *Reversed Roulette Wheel Selection Algorithm* (RRWSA) para a personalização de currículos de cursos no ambiente de *e-learning*.

RRWSA é um algoritmo de seleção bastante simples, comumente empregado para solucionar problemas de otimização e usado como o primeiro estágio de algoritmos genéticos nos quais genomas ou cromossomos individuais são escolhidos dentre a população para procriação, o que requer centenas ou milhares de conjuntos de amostras. Normalmente, o algoritmo de seleção de roleta funciona por organizar os cromossomos de acordo com sua função de aptidão ou valor de *fitness*. No contexto da construção de um currículo, os autores inicializaram o algoritmo ao selecionar a população inteira (conjunto de lições) como candidatos para eliminação.

Figura 11 - Fluxograma da personalização da sequência de aprendizagem



Fonte: BALLERA et al. (2014, p.92)

Nesse ciclo, cada um dos conteúdos possui seu valor de *fitness* e são filtrados com base nesse valor. As lições com um valor *fitness* mais alto que o limite serão eliminadas, as que permanecerem passarão por um processo de recombinação e o ciclo será repetido até a terceira geração. Depois de o currículo ter sido definido, os resultados do exame são informações diretas sobre o conhecimento do aluno e a interação dele com o curso. O desempenho do exame é dinamicamente construído com base no histórico do aluno na leitura dos materiais de aprendizagem.

Como parte da conclusão, os autores geraram uma série de tabelas que exemplificam os resultados das interações dos alunos com o sistema.

Figura 12 - Análise de desempenho

Lesson_No	Allow_time	Time_Reviewed	RST	PTA	irs	Actual-irs	dv	Review_Points	Review_Perf
L1	10	5	1	0.5	1	0.50	0.50	0.04	0.21
L2	10	4	1	0.4	1	0.60	0.60	0.05	0.25
L3	10	3	1	0.3	1	0.70	0.70	0.06	0.29
L4	10	2	1	0.2	1	0.80	0.80	0.07	0.33
L5	10	1	1	0.1	1	0.90	0.90	0.08	0.38
L6	10	1	1	0.1	1	0.90	0.90	0.08	0.38
L7	10	1	1	0.1	1	0.90	0.90	0.08	0.38
L8	10	3	1	0.3	1	0.70	0.70	0.06	0.29
L9	10	12	0	1.2	1	-0.20	0.00	0.00	0.00
L10	10	14	0	1.4	1	-0.40	0.00	0.00	0.00
L11	10	8	1	0.8	1	0.20	0.20	0.02	0.08
L12	10	2	1	0.2	1	0.80	0.80	0.07	0.33
									2.92

Fonte: BALLERA et al. (2014, p. 96)

Como exemplo, a Figura 9 mostra uma tabela que fornece os resultados resumidos da revisão da matriz de desempenho de 12 lições. O sistema dinamicamente coletou o número de vezes que os alunos pressionaram a seta indo e voltando de uma lição em particular. Quanto menor o tempo gasto registrado, maior o seu desempenho na revisão. Ao final do trabalho os autores informaram ter obtido sucesso no desenvolvimento de um sistema capaz de produzir uma sequência de aprendizado personalizada.

3 MATERIAIS E MÉTODOS

3.1 Materiais

O primeiro passo do desenvolvimento do presente trabalho envolveu utilizar a ferramenta StArt (ZAMBONI et al., 2010) da Universidade Federal de São Carlos (UFSCar) para fazer uma revisão de literatura com foco nos assuntos relevantes ao trabalho.

A criação da estrutura que contém os materiais referentes aos conteúdos de Estrutura de Dados foi feita utilizando um modelo padrão de arquivo do *SCORM 2004 4th Edition*, que é a versão mais recente do pacote. O modelo de agregação de conteúdo do SCORM especifica que o conteúdo deve ser compactado em um diretório independente ou em um arquivo com extensão *ZIP*. Essa entrega é chamada de PIF (*Package Interchange File*). O PIF sempre deve conter um arquivo XML chamado `imsmanifest.xml` na raiz (RUSTICI SOFTWARE, 2020). Os títulos que identificam cada um desses conteúdos foram utilizados para gerar a base de dados que foi usada para treinar a Rede Neural.

Para a implementação da rede, foi utilizado o *Keras*, que é uma API de Redes Neurais de alto nível, escrita em Python e capaz de rodar sobre *TensorFlow*, *CNTK* ou *Theano* (KERAS DOCUMENTATION, 2020). Também foi utilizada a biblioteca do *NumPy* do *Python*, por ela suportar o uso de estruturas como listas e matrizes multidimensionais (NUMPY.ORG, 2020).

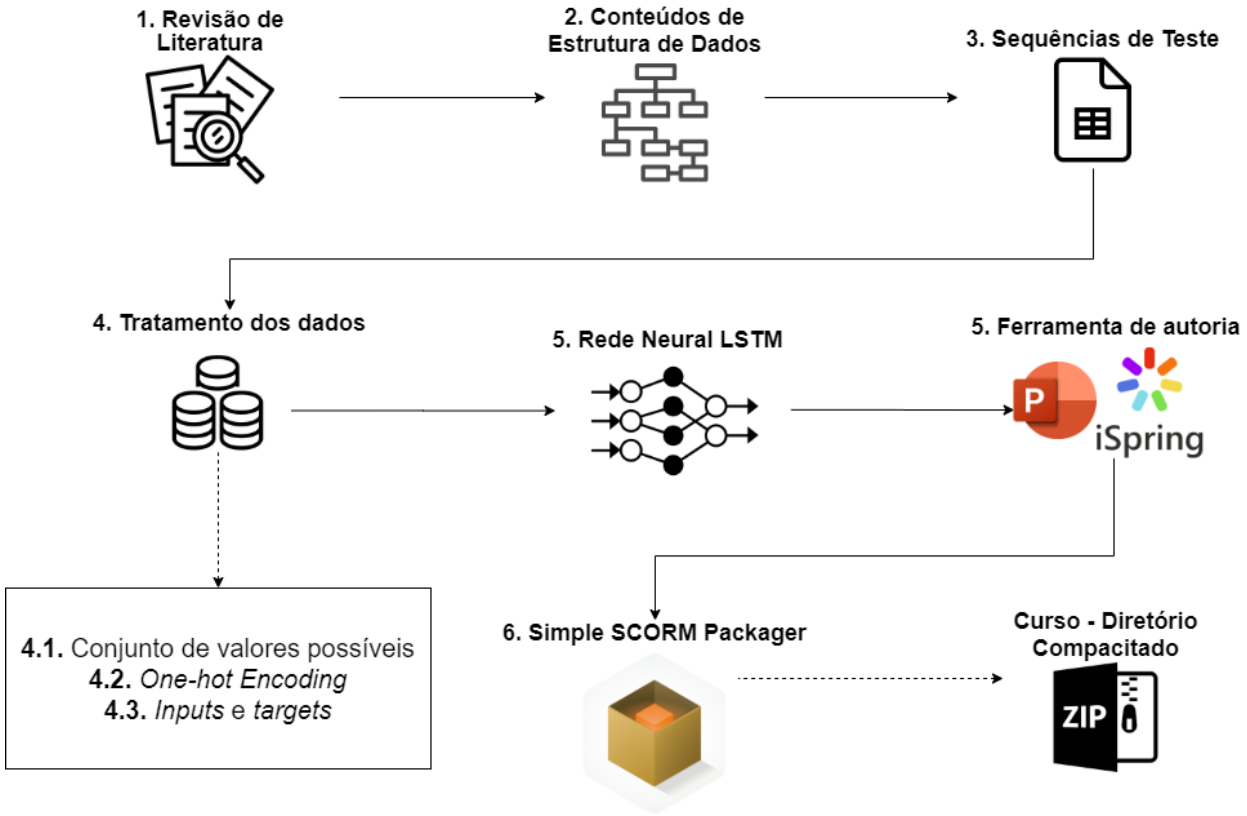
Os SCOs foram criados utilizando uma ferramenta de autoria chamada iSpring Free, que é uma extensão do Microsoft PowerPoint e auxilia no processo de criação de objetos de aprendizagem, seguindo o padrão SCORM.

Tendo os SCOs, a ferramenta Simple SCORM Packager (SSP) foi utilizada para criar um curso, ou pacote, no padrão SCORM, que possui vários módulos ou tópicos contidos em um mesmo curso. Propriedade da Jca Solutions (2020), o SSP cria os metadados e arquivos do arquivo *manifest*. Em seguida, compacta esses arquivos com o conteúdo do curso. O pacote resultante pode ser carregado em um LMS compatível e disponibilizado on-line.

3.2 Métodos

Todo o processo que levou ao desenvolvimento do sistema começou com a revisão de literatura e foi até a aplicação da sequência gerada na estrutura do curso.

Figura 13 - Fluxograma do desenvolvimento do módulo



O primeiro passo consistiu em realizar a revisão de literatura de trabalhos publicados que trouxessem informações sobre *e-learning*, sequências didáticas, objetos de aprendizagem, SCORM e Redes Neurais. As bases pesquisadas foram as da *SciELO*, *IEEE* e *Microsoft Academics*, sendo que a maioria dos trabalhos selecionados foram encontrados na *IEEE*.

O segundo passo foi o de determinar quais seriam os conteúdos e materiais utilizados para o ensino da matéria de Estrutura de Dados e definir como eles seriam organizados. Essa atividade contou com o auxílio de um especialista de domínio, que é professor deste conteúdo no ensino superior.

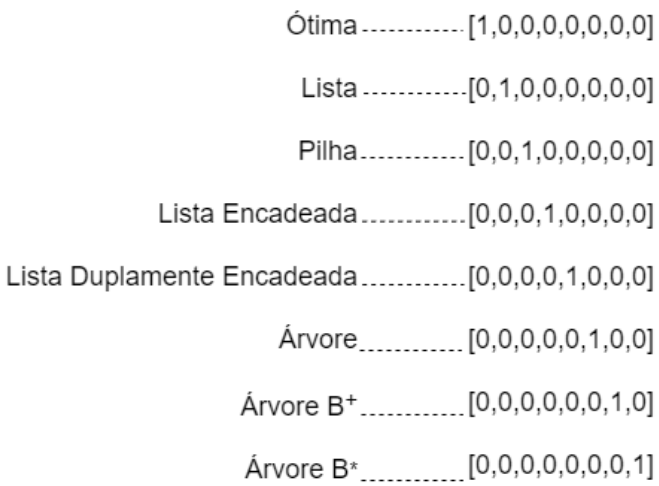
No terceiro passo, e ainda com a ajuda do especialista de domínio, os títulos identificadores dos conteúdos definidos no segundo passo foram utilizados para gerar um documento CSV (*Comma-separated values*) com as sequências de teste. Por exemplo, considerando-se que a matéria tenha sido dividida nos conteúdos Lista, Pilha, Lista Encadeada, Lista Duplamente Encadeada, Árvore, Árvore B^+ e Árvore B^*, os dados de teste iriam consistir de diferentes sequências formadas por esses conteúdos. Além dos conteúdos, o primeiro item de cada sequência é um valor nominal que indica a qualidade da sequência com base na nota fictícia tirada por um aluno que a tenha seguido. Sequências com notas maiores ou iguais a 0 e menores que 3 são classificadas como “Péssima”, maiores ou iguais a 3 e menores que 6 são classificadas como “Ruim”, maiores ou iguais a 6 e menores que 8 são

“Regular”, maiores ou iguais a 8 e menores que 9 são “Boa”, e maiores ou iguais a 9 e menores ou iguais a 10 são classificadas como “Ótima”.

No quarto passo, os dados de entrada foram tratados para que a Rede Neural pudesse utilizá-los para aprender a gerar uma sequência didática. A primeira etapa desse processo consistiu em informar para a Rede Neural o conjunto de possíveis itens distintos existentes no contexto em questão, utilizando uma função da biblioteca Python.

A segunda etapa foi a de transformar esses dados em valores que fossem interpretáveis por Redes Neurais, pois elas não trabalham diretamente com variáveis nominais, ou seja, variáveis não numéricas. Para que isso fosse possível, foi aplicada uma técnica sobre os dados, para que eles fossem convertidos em valores numéricos, chamada *One-Hot Encoding* (BROWNLEE, 2017).

Figura 14 - *One-Hot Encoding*



Essas variáveis nominais foram transformadas em vetores de tamanho fixo e com a mesma quantidade de elementos que a quantidade de variáveis existentes, sendo que cada variável é representada pelo valor binário 1 em uma determinada posição.

E a terceira etapa do quarto passo foi criar os exemplos de *input*, sendo que cada um corresponde a uma sequência exceto pelo último item, e os exemplos de *target*, que correspondem à sequência exceto pelo primeiro item. Seguindo a imagem anterior, o exemplo de *input* seria Ótima, Lista, Pilha, Lista Encadeada, Lista Duplamente Encadeada, Árvore e Árvore B^+, e o exemplo de *target* seria Lista, Pilha, Lista Encadeada, Lista Duplamente Encadeada, Árvore, Árvore B^+ e Árvore B^*.

O trabalho da rede foi o de utilizar essas entradas para encontrar os relacionamentos entre os conteúdos e sua ordenação, a fim de gerar sequências didáticas, associando essa sequência a um valor de classificação de qualidade.

No quinto passo, a Rede Neural foi implementada utilizando Keras e rodando sobre o *TensorFlow*, e os dados tratados foram inseridos nela para fins de treinamento. Também foi utilizado como base o modelo de Rede Neural para geração de textos caracter a caracter, de Karpathy (2015). O tipo de rede que foi utilizada foi escolhido levando-se em consideração as necessidades específicas do projeto. Para criar uma sequência é necessário que a arquitetura da Rede Neural possibilite que ela possua "memória", ou seja, ela precisa saber quais elementos já passaram por ela, para que saiba prever qual deve ser o próximo. De acordo com Kumar e Subha (2019), LSTM (*Long Short Term Memory*) é um tipo de Rede Neural Recorrente (*Recurrent Neural Network - RNN*), capaz de lidar com grandes conjuntos de dados e superar os problemas causados por dependências de longo prazo como desaparecimento (valores muito pequenos) e explosão (valores muito grandes) de gradientes.

O sexto passo consistiu em criar os objetos de aprendizagem utilizando o Microsoft PowerPoint e a ferramenta de autoria iSpring Free que permitiu que esses objetos fossem padronizados segundo o SCORM, transformando-os em SCOs.

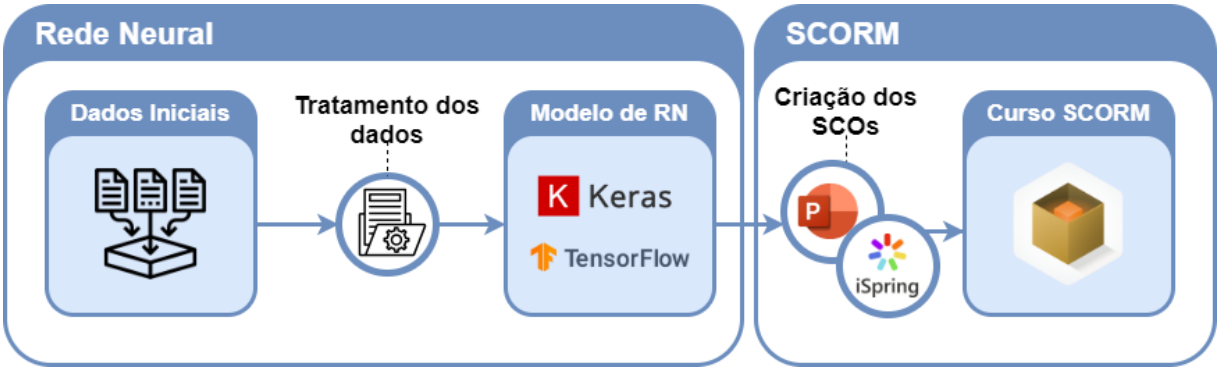
Por fim, tendo os SCOs criados e a sequência determinada, o último passo foi integrar esses dois resultados ao padrão SCORM utilizando o SSP, o que gerou como resultado o diretório compactado que constitui o curso.

4 RESULTADOS E DISCUSSÃO

4.1. Tratamento dos dados de entrada

Todo o processo de criação de uma sequência didática para ser adaptada segundo o padrão SCORM pode ser dividido em dois grupos de atividades. O primeiro engloba os passos necessários para gerar a sequência utilizando uma Rede Neural, desde o tratamento dos dados em seu formato inicial até o treinamento da rede e a criação da sequência. O segundo engloba as etapas para criar os SCOs dos conteúdos das matérias de Estrutura de Dados e o agrupamento desses SCOs em um curso SCORM seguindo a sequência gerada.

Figura 15 - Subdivisões do sistema



O formato inicial dos dados é o de um arquivo contendo todas as sequências juntas, uma por linha, e cada uma delas é composta por 26 elementos, sendo eles um valor nominal de classificação e 25 conteúdos. Esse arquivo é lido e as sequências são armazenadas em uma lista, uma seguida da outra. A partir dessa lista é possível identificar a existência de 30 elementos distintos do tipo *string*.

A variável contendo a lista de sequências foi alterada para ser uma *array* da biblioteca Python NumPy, que é uma Estrutura de Dados que pode ser multidimensional, é capaz de lidar com grandes quantidades de dados e exige que todos os seus dados sejam do mesmo tipo (DATACAMP TEAM, 2020). Tendo as sequências no formato *array*, os valores contidos foram transformados em *integers* identificadores. Considerando que há 30 elementos distintos, esses identificadores vão de 0 a 29. Em seguida, a *array* com os identificadores é codificada seguindo o método *One-Hot Encoding*, de tal modo que cada item se torna uma *array* com 30 posições, sendo que 29 delas tem o valor 0 e uma tem valor 1. A posição do valor 1 na *array* é correspondente ao valor do identificador do item.

Com todos os itens codificados, a *array* foi transformada em um tipo de dado chamado *tensor* pela função `tf.data.Dataset.from_tensor_slices()`, que é uma *array* multidimensional utilizada pelo *TensorFlow*. Esse *tensor* foi então subdividido em

batches, que são subconjuntos do *tensor* de mesmo formato, como mostra o trecho de código abaixo.

Figura 16 - Criação de *tensor* e *batches*

```
seq_length = 25
char_dataset = tf.data.Dataset.from_tensor_slices(onehot_encoded)
sequences = char_dataset.batch(seq_length+1, drop_remainder=True)
```

A definição de `drop_remainder = True` existe para que, na eventualidade de o último *batch* não possuir itens suficientes para ser uma sequência completa, ele seja descartado. E o tamanho é `seq_length+1` porque, enquanto uma sequência completa tem 26 itens, as sequências que servirão de *input* e *target* devem ter um item a menos.

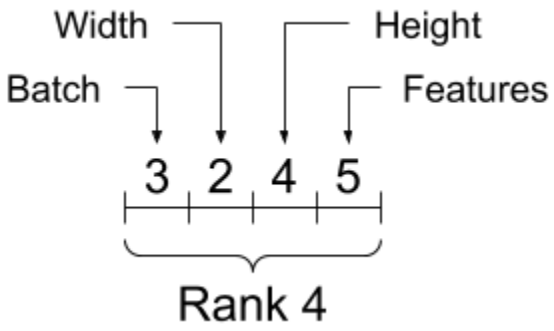
Figura 17 - Criação de *inputs* e *targets*

```
def split_input_target(chunk):
    input_text = chunk[:-1]
    target_text = chunk[1:]
    return input_text, target_text

dataset = sequences.map(split_input_target)
```

A função `map()` aplica o método `split_imput_target()` sobre cada um dos *batches* na variável `sequence`, o que resultou em um valor de `input_text` e um de `target_text` por *batch*. É importante notar qual a forma assumida por esses dados. A forma do *tensor* é indicada por um conjunto de valores numéricos, como mostra a Figura 18.

Figura 18 - Forma do *tensor*



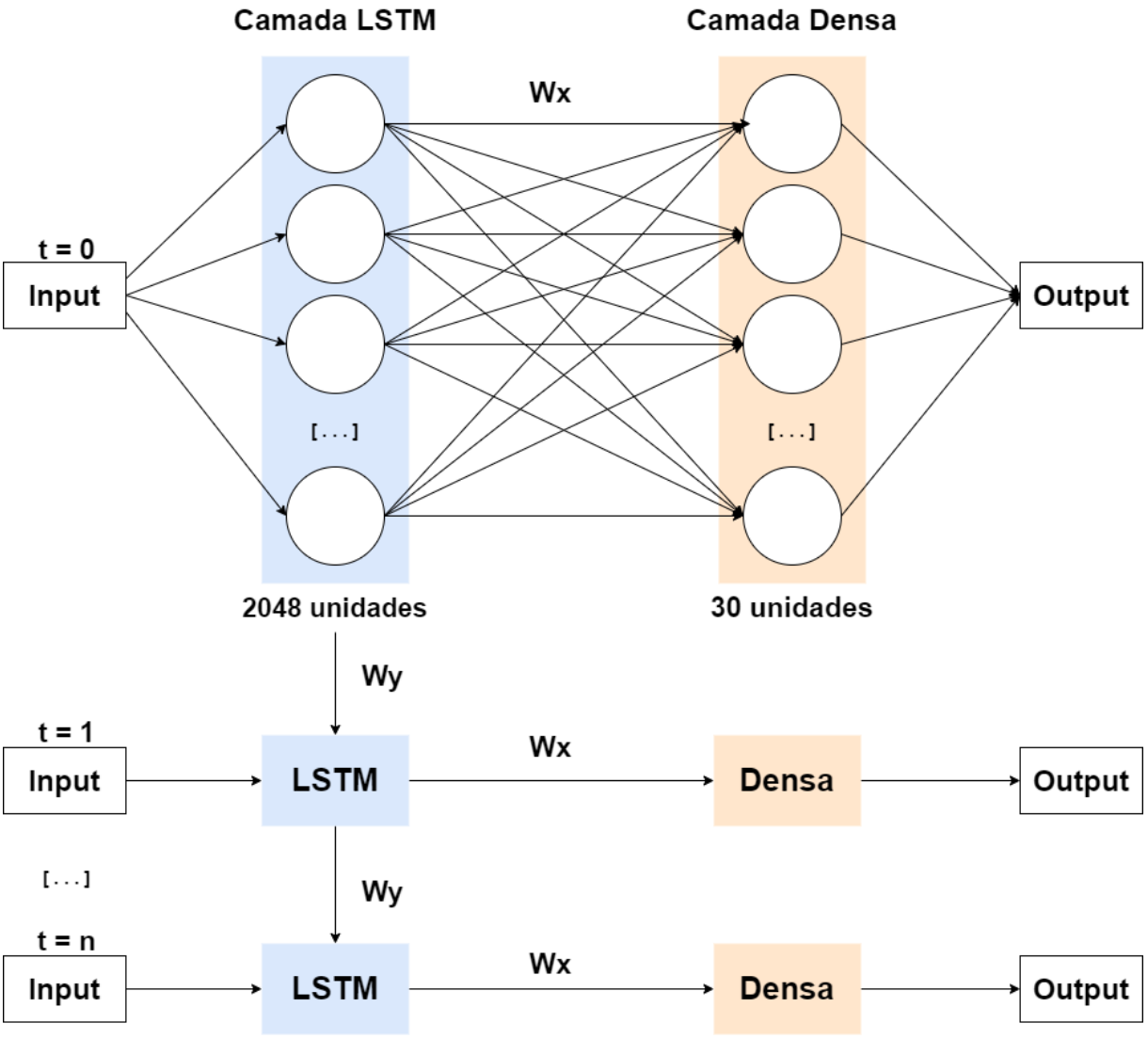
Fonte: TENSORFLOW.ORG (2020)

O valor de *rank* indica o número de dimensões do *tensor*. No contexto deste trabalho, a variável `sequence` possui *rank* 3 e forma (1, 25, 30), ou seja, *batches* de tamanho 1, com 25 itens, cada um formado por um vetor com 30 posições.

4.2. Modelo da Rede Neural

O modelo da Rede Neural foi criado com uma função `build_model()` e consiste em duas camadas da API do Keras, camada LSTM e camada Densa, encapsuladas pelo método `Sequential()` que agrupa linearmente as camadas em um modelo Keras.

Figura 19 - Modelo da Rede Neural



Na Figura 19, o *input* no momento t = 0 representa o primeiro valor de uma sequência que está sendo inserido na rede, e esse valor é enviado para as células da camada LSTM. Cada célula irá considerar o conteúdo recebido e utilizar os valores dos pesos da camada para prever o conteúdo seguinte. A camada LSTM envia a informação do resultado, representado por *Wx*, para a camada densa. A camada densa, por sua vez, recebe esse *input*, multiplica-o pelos valores dos pesos da sua própria camada e aplica também uma função de ativação. Ela possui 30 unidades, que representam os 30 valores distintos que podem ser escolhidos como o

próximo valor da sequência, e sua saída será determinada pelos valores de ativação calculados nessas unidades.

Esse mesmo ciclo ocorre várias vezes durante o treino da rede. Para cada uma dessas iterações, a camada LSTM armazena as informações das decisões que foram tomadas nas iterações seguintes, representado por *Wy*, que são utilizadas na iteração seguinte.

Durante o treinamento do módulo, o valor de *output* para cada iteração é comparado com os valores de *target*. A diferença entre a saída e o valor esperado é utilizado para recalcular os valores dos pesos das camadas. Cada ciclo de treinamento recebe o nome de *epoch*, e quanto maior o número de *epochs* mais fácil será para a rede trabalhar com os valores de treinamento. É importante lembrar que tanto pouco treino quanto treino em excesso podem ter impacto negativo na rede. Com pouco treino, ela não consegue aprender a lidar com os valores de treino, um problema chamado de *underfitting*. Com treino em excesso, a rede pode ficar condicionada aos valores de treino, e não conseguirá obter sucesso quando receber valores diferentes, o que é chamado de *overfitting*. Uma forma de determinar o desempenho da rede em relação a quantidade de treino é, tendo uma base de dados grande o suficiente, dividi-la em um grupo para treinar o modelo, e outro para testá-lo. Desse modo, a rede será testada com um conjunto de dados que não fazia parte do grupo de treinamento, e seria possível identificar se ela não foi treinada o suficiente ou em excesso a ponto de não apresentar um bom desempenho com os dados de teste.

Utilizando o Keras, é possível obter uma visualização do progresso do aprendizado da rede a cada *epoch*.

Figura 20 - Treinamento da Rede Neural

```
Epoch 44/50
50/50 [==============================] - 15s 291ms/step - loss: 0.0884
Epoch 45/50
50/50 [==============================] - 15s 292ms/step - loss: 0.0873
Epoch 46/50
50/50 [==============================] - 15s 293ms/step - loss: 0.0845
Epoch 47/50
50/50 [==============================] - 15s 297ms/step - loss: 0.0827
Epoch 48/50
50/50 [==============================] - 15s 298ms/step - loss: 0.0796
Epoch 49/50
50/50 [==============================] - 15s 291ms/step - loss: 0.0784
Epoch 50/50
50/50 [==============================] - 15s 294ms/step - loss: 0.0764
```

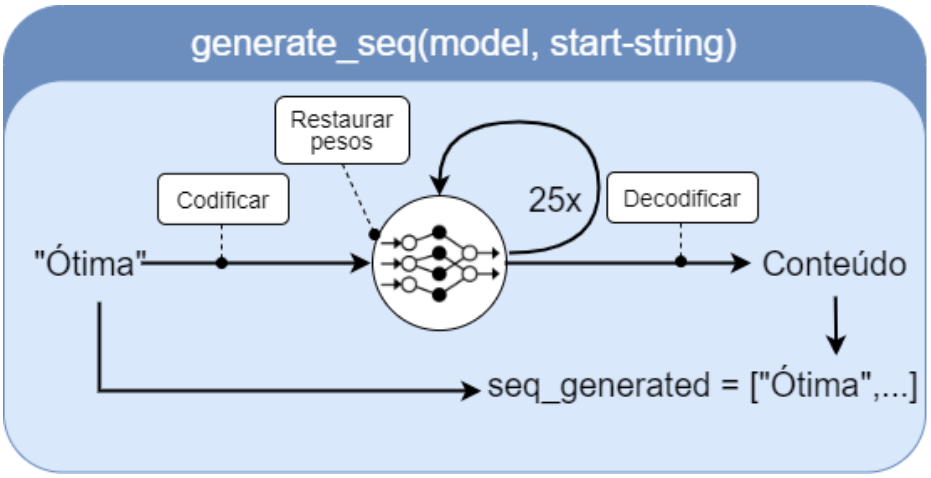
O valor *loss* representa a diferença entre o valor de saída da rede e o esperado para uma determinada entrada. A diminuição gradativa do valor de *loss* a cada *epoch* indica que a rede está obtendo mais sucessos ao fazer previsões.

Outro importante fator do processo de treinamento da rede é o armazenamento de *checkpoints*. *Checkpoints* são arquivos com os pesos utilizados pelo modelo da Rede Neural, sendo que esses pesos foram inicializados no formato de matriz de acordo com as regras da classe *Glorot normal* especificada no parâmetro `kernel_initializer`, definido na camada LSTM do modelo. Durante o treinamento os pesos são periodicamente salvos em um diretório para que, ao final do processo, os últimos, que obtiveram maior taxa de sucesso, possam ser restaurados e utilizados pela rede para que ela possa executar a tarefa que foi treinada para fazer.

4.3. Criação e padronização da sequência

Após a implementação do modelo da Rede Neural, é executada a função `generate_seq()` para gerar a sequência didática. A Figura 21 apresenta o funcionamento interno dessa função.

Figura 21 - Fluxo de criação da sequência didática



No exemplo da figura, a função recebe como parâmetro o modelo da Rede Neural e uma *string* inicial com valor “Ótima” que servirá como o valor de entrada para o modelo da rede e como o primeiro conteúdo da sequência. As tarefas executadas por essa função incluem, primeiramente, codificar a *string* de entrada e deixá-la no formato correto para ser inserida no modelo. Então ela restaura os valores dos últimos pesos salvos nos *checkoints* e, por fim, utiliza uma estrutura de *loop for* para gerar, passo a passo, uma sequência contendo

25 conteúdos, armazenando-os em uma lista. A Figura 22 apresenta uma sequência criada pela rede, após 200 *epochs* de treinamento.

Figura 22 - Sequência criada pela Rede Neural

```
0 :  Ótima
1 :  Filas
2 :  Listas Dinâmicas Simplesmente Encadeadas
3 :  Filas de Prioridade
4 :  Árvores Binárias de Busca Balanceadas
5 :  Aplicações de Filas
6 :  Pilhas Dinâmicas
7 :  Árvores Binárias
8 :  Listas Estáticas
9 :  Listas Dinâmicas Duplamente Encadeadas
10 :  Listas Estáticas Circulares
11 :  Listas Dinâmicas Simplesmente Encadeadas Circulares
12 :  Tipos Abstratos de Dados
13 :  Listas Estáticas Sequenciais
14 :  Pilhas
15 :  Listas
16 :  Filas
17 :  Listas Dinâmicas
18 :  Listas Dinâmicas Simplesmente Encadeadas
19 :  Árvores Binárias de Busca
20 :  Listas Estáticas Encadeadas
21 :  Recursividade
22 :  Percursos em Árvores Binárias
23 :  Aplicações de Pilhas
24 :  Listas Dinâmicas Duplamente Encadeadas Circulares
25 :  Filas Dinâmicas
```

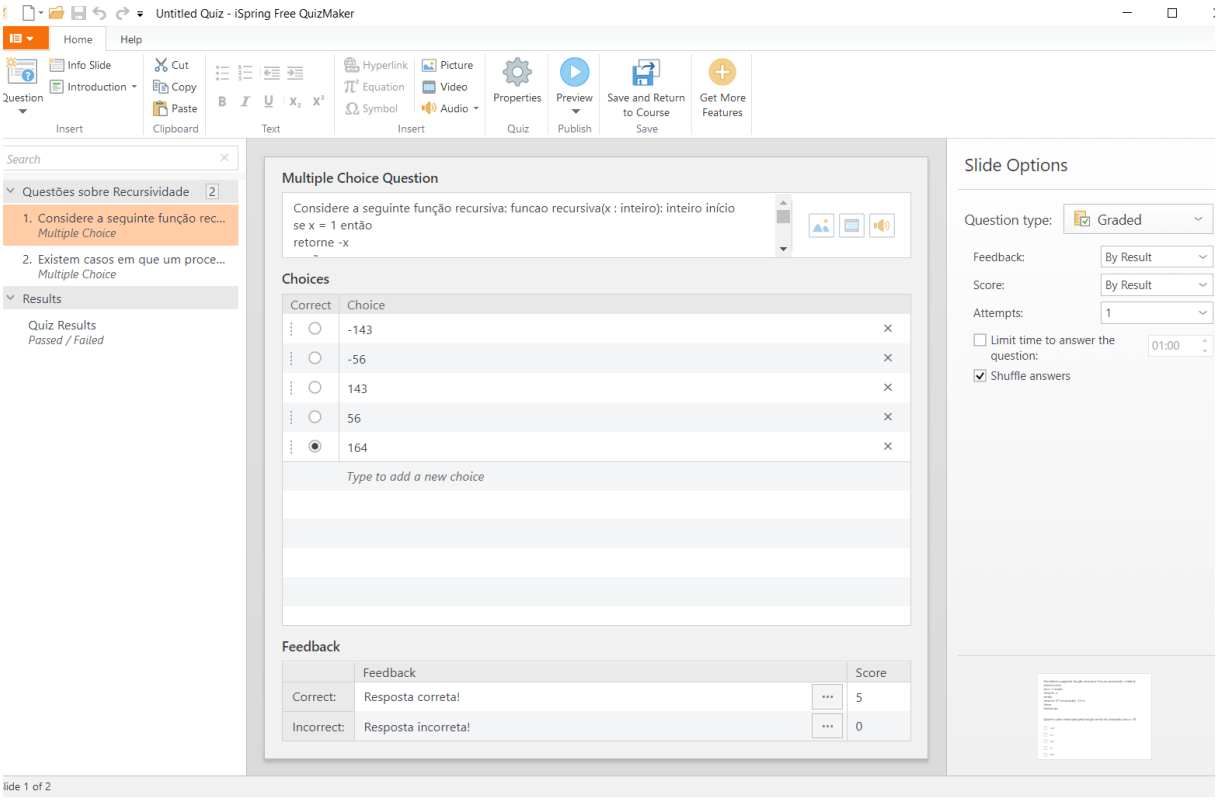
A partir desse resultado foi possível identificar que existem algumas discrepâncias em relação ao que seria esperado de uma sequência da matéria de Estrutura de Dados. Por exemplo, o fato de o conteúdo Árvores Binárias de Busca Balanceada (4) ser listado antes de Árvores Binárias (7) e Árvores Binárias de Busca (19). Isso ocorre em função de os dados utilizados no treinamento da rede serem fictícios. Ainda assim, foi possível identificar, junto com o especialista de domínio, que a Rede foi capaz de detectar e aprender os padrões existentes nas sequências. Esses padrões foram definidos de maneira que fossem facilmente reconhecíveis e, desse modo, fosse possível verificar o desempenho da Rede em identificá-los. Tais padrões incluem a não existência de ocorrências de valores indicadores de qualidade no decorrer da sequência, além do primeiro que foi inserido na função `generate_seq()`. Outro padrão criado foi o fato de que toda sequência iniciada com "Ótima" deveria ter "Filas" como o primeiro conteúdo selecionado.

Tendo a sequência criada, foi finalizado o primeiro grupo de atividades apresentados na metodologia, que diz respeito à Rede Neural. Em seguida foram iniciadas as atividades referentes ao padrão SCORM.

A primeira atividade foi a de criar manualmente objetos de aprendizagem seguindo o padrão SCORM para cada um dos conteúdos presentes na sequência. Esse procedimento foi

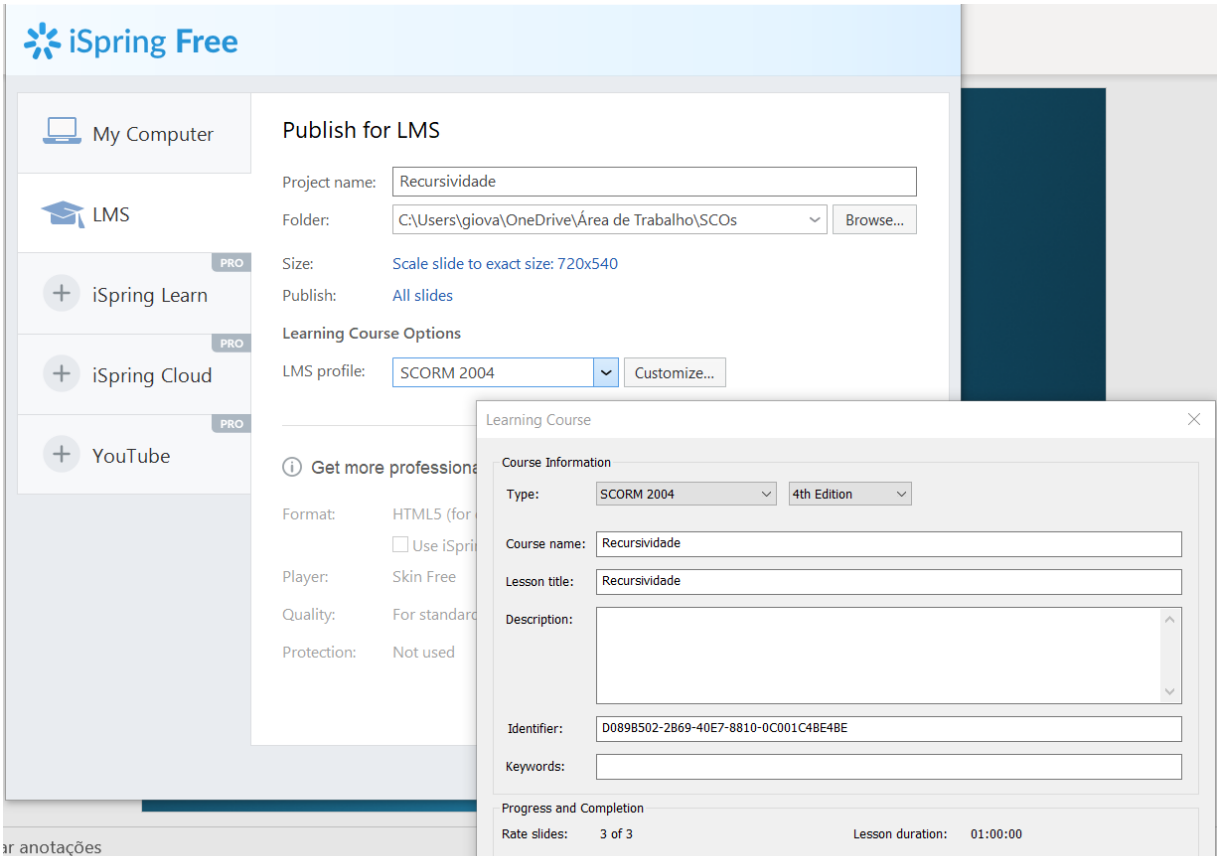
feito utilizando a ferramenta iSpring Free que permite a inserção de páginas web, quizzes pontuados ou questionários não pontuados aos slides. A Figura 23 mostra como exemplo a tela para inserção de objeto de aprendizagem no formato de um quiz, tratando do conteúdo de Recursividade.

Figura 23 - Adição de quiz em SCO



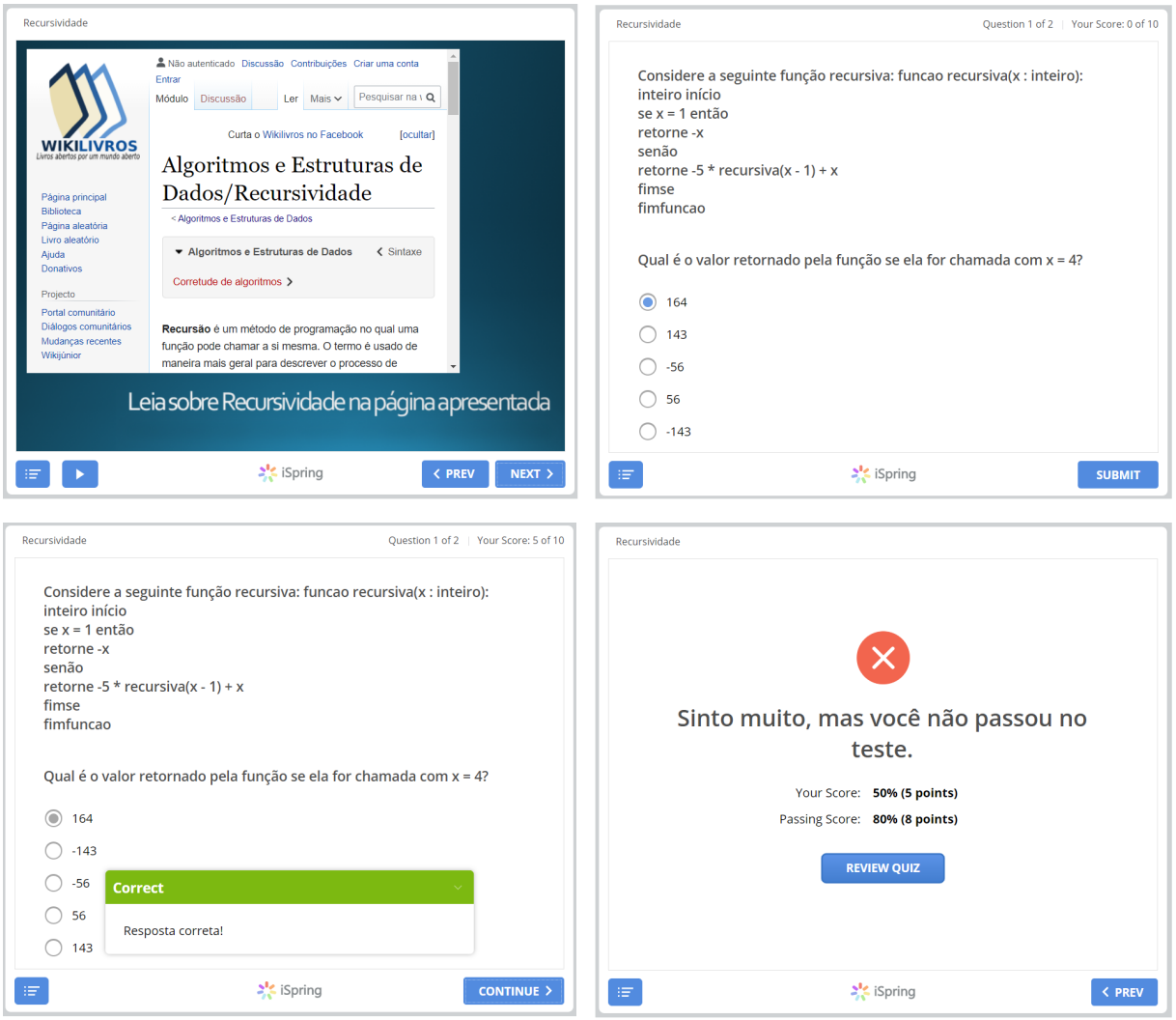
Essa funcionalidade de inserção de quiz permite que sejam adicionadas questões e suas alternativas, as mensagens exibidas caso o aluno acerte ou erre a resposta, a pontuação de cada questão, limite de tentativas permitidas, informação que define se as alternativas devem ser aleatorizadas ou não, entre outras opções de configuração. O iSpring Free também permite a exportação de cada slide como um SCO, como mostra a Figura 24.

Figura 24 - Exportação de SCO para LMS



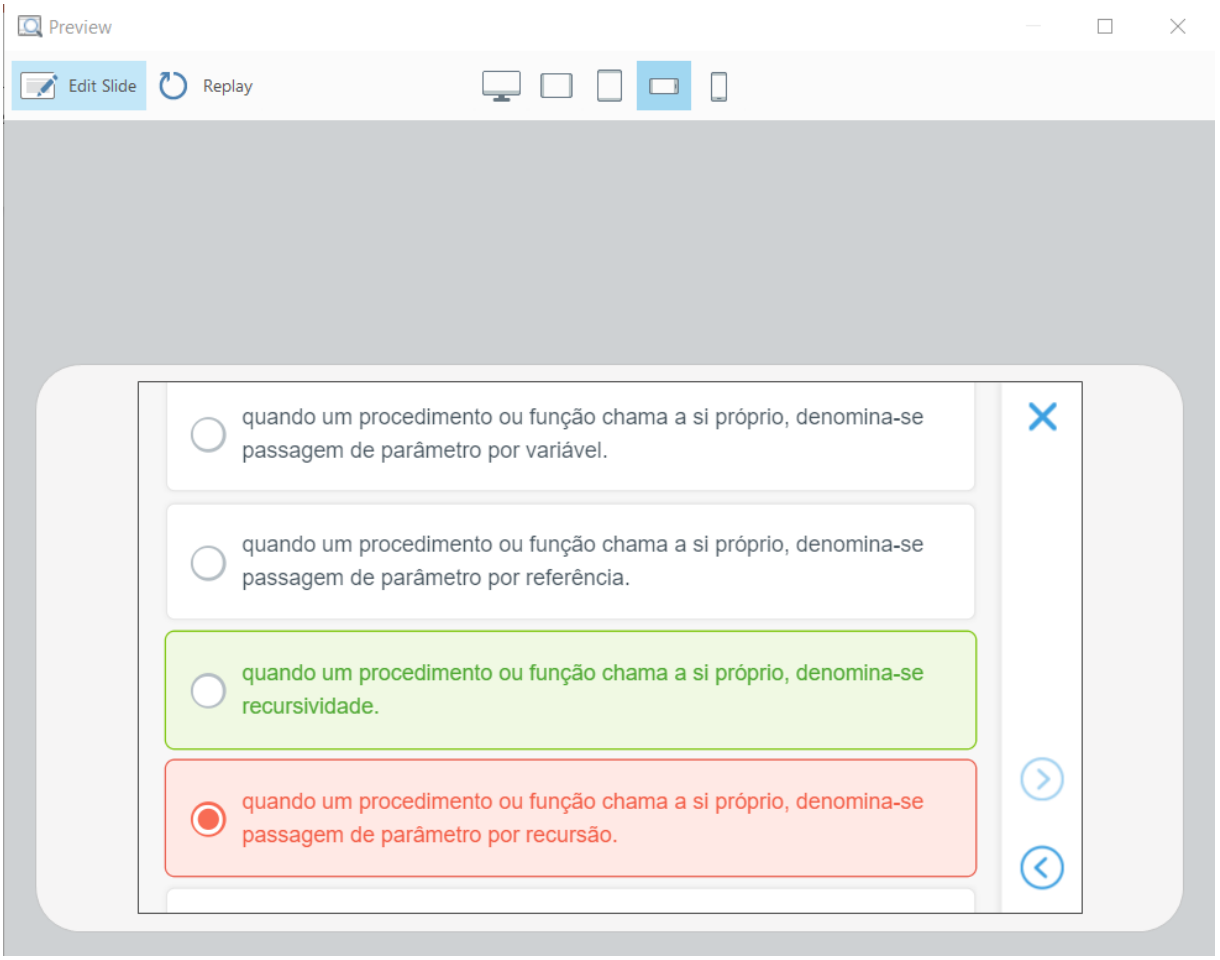
O campo *LMS profile* permite a seleção de qual será o padrão que deve ser seguido para exportação do objeto de aprendizagem. No contexto deste trabalho, os SCOs foram exportados segundo o *SCORM 2004 4th Edition*.

Figura 25 - Telas da visualização do SCO



A ferramenta iSpring Free ainda oferece a visualização do SCO resultante, incluindo a página web com rolagem, o quiz interativo, que pode ser respondido e oferece o *feedback* que foi definido durante a sua criação, o resultado do quiz e a correção das questões, como mostra a Figura 25. Adicionalmente, o modo de visualização *preview* exibe a capacidade de responsividade do conteúdo do SCO, para que ele possa ser apresentado em dispositivos de diferentes dimensões.

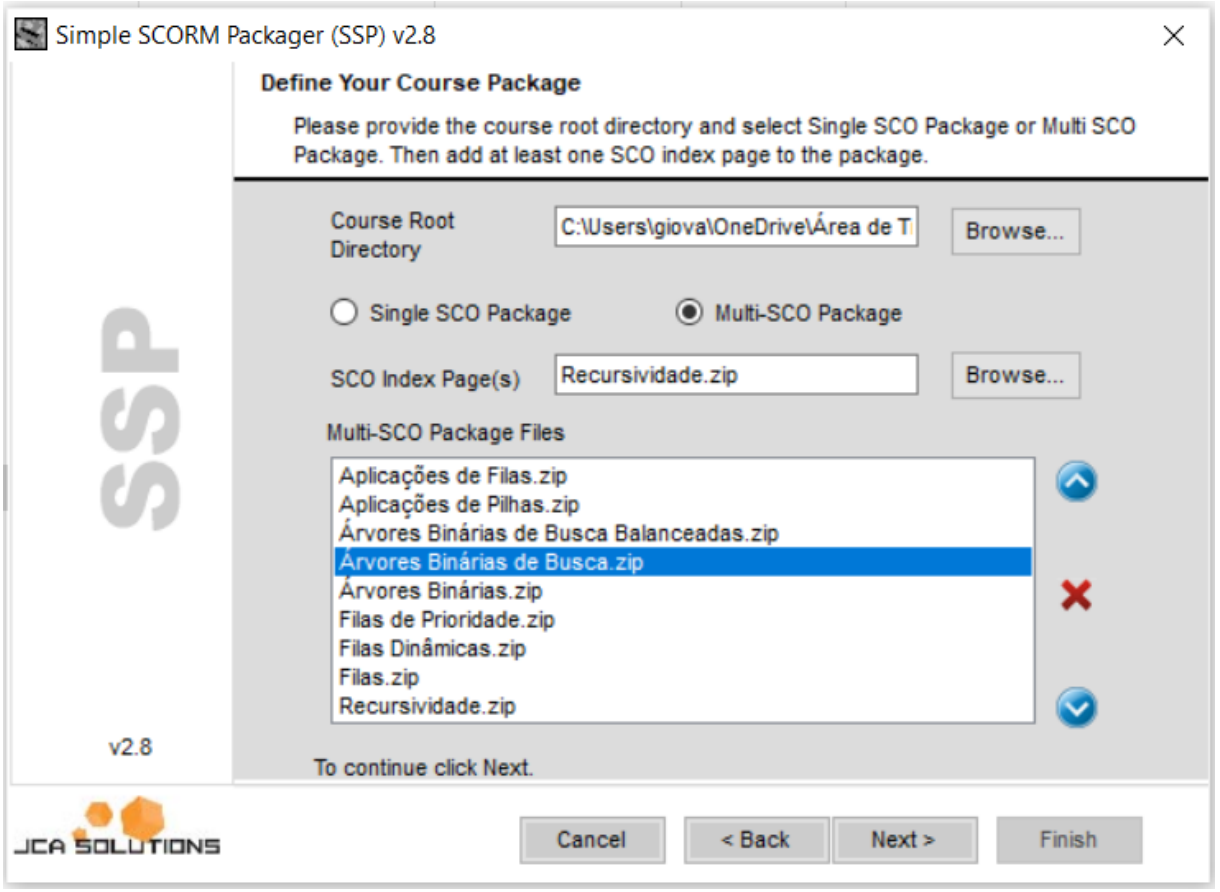
Figura 26 - Demonstração de responsividade do SCO



A Figura 26 apresenta a tela de correção de uma das questões do quiz, que foi respondida incorretamente, demonstrando como ficaria a visualização em uma tela de dispositivo *mobile* com orientação horizontal.

Após todos os SCOs terem sido exportados, o último passo foi utilizar a ferramenta SSP para criar um curso com múltiplos conteúdos, também na versão *SCORM 2004 4th Edition*, seguindo o padrão SCORM e a sequência que foi gerada anteriormente pela Rede Neural.

Figura 27 - Simple SCORM Packager



Na Figura 27 é apresentada a tela em que o SSP permite que seja definido o tipo de curso que será criado, e se ele possui um único SCO ou mais de um. Logo em seguida, no campo *SCO Index Page(s)* é possível importar os SCOs. E no campo abaixo, onde os SCOs importados são exibidos, é possível deletá-los ou alterar a ordem em que eles serão apresentados no curso utilizando os botões localizados ao lado direito do campo. Desse modo, foi possível configurar o curso para apresentar os conteúdos na sequência criada pela Rede Neural. O resultado gerado pelo SSP foi um diretório compactado no formato *.zip*, contendo o curso de Estrutura de Dados com múltiplos SCOs e de acordo com o padrão SCORM, com a sequência dos conteúdos tendo sido definida por uma Rede Neural LSTM.

5 CONSIDERAÇÕES FINAIS

O presente trabalho descreveu o desenvolvimento de um módulo para geração de sequências didáticas utilizando Redes Neurais e o padrão SCORM no contexto dos conteúdos de Estrutura de Dados.

A arquitetura do módulo envolve a área referente à Rede Neural e a área referente ao padrão SCORM. Incluídas no processo de desenvolvimento estão as fases de estruturação dos dados iniciais, tratamentos desses dados para servirem como dados de entrada para a rede, modelagem e implementação da Rede Neural.

Os dados iniciais são estruturados em um arquivo CSV com cada linha representando uma sequência, e cada sequência sendo formada por um valor classificatório na primeira posição, que indica a qualidade da sequência, e os 25 conteúdos que compõem a matéria de Estrutura de Dados. A fase de tratamento desses dados iniciais contou com a linguagem de programação Python e a API de desenvolvimento de Redes Neurais Keras junto com o *framework TensorFlow*. Além disso, para que a rede aceitasse valores nominais como entrada, eles foram codificados para serem vetores de tamanho único, utilizando a técnica *One-Hot Encoding*.

A Rede Neural em si foi desenvolvida tendo-se como base um modelo de geração de textos caracter a caracter, de Karpathy (2015). Esse modelo foi adaptado para o contexto do presente trabalho, e é composto por dois tipos de camadas de Redes Neurais, que são a camada LSTM e a camada Densa. A camada LSTM em específico foi escolhida para essa tarefa por ser capaz de armazenar informações de decisões tomadas em iterações anteriores da rede, e utilizá-las em iterações futuras, o que é ideal na tarefa de gerar sequências.

Ao final do desenvolvimento do módulo de criação de sequências didáticas, foi implementada uma função que recebe uma *string* inicial e a insere no modelo da Rede Neural, recebe o resultado previsto por ela e o insere novamente no modelo. Esse processo é executado em *loop* até que se tenha uma sequência completa, ou seja, 25 *loops*. Por fim, foram utilizadas as ferramentas Microsoft PowerPoint, iSpring Free e SSP para criar os objetos de aprendizagem e agregar esses objetos em um curso compatível com o padrão SCORM, ordenados seguindo a sequência determinada com a Rede Neural.

A respeito do desempenho da Rede Neural, é necessário que sejam feitas algumas observações. Primeiramente, considerando-se que a rede aprende com base nos dados de entrada que obtém, o seu desempenho melhora à medida que aumenta a quantidade de dados, de casos reais, que ela recebe como entrada. No contexto deste trabalho os dados de entrada eram fictícios, mas, ainda assim, eles foram criados seguindo padrões que fossem facilmente

reconhecíveis para que fosse possível apontar se a rede era capaz de identificá-los. Por exemplo, foi determinado que sequências ótimas começam apenas com o conteúdo de “Filas”, e a rede foi capaz de manter esse padrão ao gerar uma nova sequência tendo como *string* de entrada o valor “Ótima”. Ou ainda o fato de que ela não insere outro valor de definição de qualidade, como “Boa” ou “Péssima”, no decorrer da sequência, apenas conteúdos da matéria.

Por isso, para trabalhos futuros, é proposto que sequências reais que tenham sido seguidas por alunos sejam inseridas na rede. Isso pode ser feito ao se disponibilizar essas sequências em plataformas *on-line*. Adicionalmente, as sequências criadas pela rede também poderiam ser disponibilizadas, de forma que fossem validadas com base no desempenho dos alunos que a seguissem.

Além disso, o modelo da rede atualmente permite que haja repetições de conteúdos no decorrer da sequência. Considerando que atualmente ela possui um tamanho fixo, isso significa que outro conteúdo não estaria presente. Desse modo, é sugerido que seja realizada uma alteração na modelagem da rede que permita a geração de sequências de tamanhos variados. Esse objetivo poderia ser atingido ao, por exemplo, se determinar um valor nulo. Desse modo, a maior sequência seria formada por um número x de conteúdos, e as demais seriam formadas por um número $y + a$, sendo que y representa os conteúdos, e a representa valores nulos, de modo que $y + a = x$. Essa proposta já se apresenta possivelmente viável pelo fato de que o modelo que inspirou a Rede Neural deste trabalho, de geração de textos, permite a existência de espaços em branco entre as palavras. Com essa alteração as redes criadas poderiam possuir conteúdos repetidos sem que isso significasse a exclusão de outros conteúdos, o que não é algo incomum no contexto de metodologias para definição de sequências didáticas.

O modelo da Rede Neural resultante cumpre o objetivo proposto inicialmente de aprender a gerar sequências didáticas no contexto da matéria de Estrutura de Dados. Atualmente o processo de gerar a sequência é realizado de forma independente do processo de criar os SCOs e o curso, exportado no formato de um diretório compactado. O curso resultante foi sequenciado segundo a sequência didática gerada pela rede e, por seguir o padrão SCORM, possui capacidade de integração com os sistemas de LMS compatíveis.

1.

REFERÊNCIAS

ALSMADI, Mutasem Khalil; OMAR, Khairuddin Bin; NOAH, Shahrul Azman; ALMARASHDAH, Ibrahim. Performance Comparison of Multi-layer Perceptron (Back Propagation, Delta Rule and Perceptron) algorithms in Neural Networks. In: IEEE INTERNATIONAL ADVANCE COMPUTING CONFERENCE, 9., 2009, Patiala. **Proceedings […]** . Patiala: Ieee, 2009. p. 296-299.

ARAÚJO, Lucas Moreno de; FAGUNDES, Fabiano. AntStudy: Proposta de definição de sequências de estudo utilizando Ant System. In: Encontro de Computação e Informática do Tocantins, 13., 2011, Palmas. **Anais [...]** . Palmas: Ceulp, 2011. p. 98-108. Disponível em: http://ulbra-to.br/encoinfo/edicoes/2011/artigos/antstudy-proposta-de-definicao-de-sequencias-de-estudo-utilizando-ant-system. Acesso em: 21 jun. 2020.

BALLERA, Melvin; LUKANDU, Ismail Ateya; RADWAN, Abdalla. Personalizing E-Learning Curriculum Using Reversed Roulette Wheel Selection Algorithm. In: INTERNATIONAL CONFERENCE ON EDUCATION TECHNOLOGY AND COMPUTERS, 6., 2014, Lodz. **Proceedings [...]** . Lodz: Lodz University Of Technology, 2014. p. 91-97.

BRITO, Parcilene Fernandes de; FAGUNDES, Fabiano; ALVES, João Bosco da Mota. Uma Estrutura para Definição de Seqüências de Estudos Baseada na Técnica Ant System. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 13., 2002, São Leopoldo. **Anais [...]** . São Leopoldo: Unisinos, 2002. p. 247-253. Disponível em: https://www.br-ie.org/pub/index.php/sbie/article/view/185/171. Acesso em: 21 jun. 2020.

BROWNLEE, Jason. **Why One-Hot Encode Data in Machine Learning?** 2017. Disponível em: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/. Acesso em: 3 maio 2020.

BRUMINI, Gordana; SPALJ, Stjepan; MAVRINAC, M.; BIOčINA-LUKENDA, D.; STRUJIC, Mihovil; BRUMINI, Martina. Attitudes towards e‐learning amongst dental students at the universities in Croatia. **European Journal Of Dental Education**. p. 15-23. nov. 2013.

CASTRO, Marcos. **Introdução a Estruturas de Dados**. Plataforma Udemy, 2016. Disponível em: https://www.udemy.com/course/introducao-a-estruturas-de-dados/. Acesso em: 18 jun. 2020.

CAO, Min; CAO, Zhen. Teaching Data Structures and Software Architecture while Constructing Curriculum Platform. In: International Conference on Computer Science & Education, 6., 2011, Singapore. **Proceedings...** . Singapore: Springer, 2011. p. 1433-1437.

CHEN, Qing; YUE, Xuanwu; PLANTAZ, Xavier; CHEN, Yuanzhe; SHI, Conglei; PONG, Ting-chuen; QU, Huamin. ViSeq: Visual Analytics of Learning Sequence in Massive Open Online Courses. **Ieee Transactions On Visualization And Computer Graphics**, v. 26, n. 3, p.1622-1636, 1 mar. 2020. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/tvcg.2018.2872961. Disponível em: https://ieeexplore.ieee.org/document/8477163. Acesso em: 31 mar. 2020.

CLAYPOOLE, Brynn; THOMAS; Horatio. **Intro to Data Structures and Algorithms**. Plataforma Udacity, 2020. Disponível em: https:https://www.udacity.com/course/data-structures-and-algorithms-in-python--ud513. Acesso em: 18 jun. 2020.

CLEMENT, J.. **Facebook:** number of monthly active users worldwide 2008-2020. 2020. Disponível em: https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/. Acesso em: 02 maio 2020.

DALMOLIN, Ana Cláudia et al. Learning styles preferences and e-learning experience of undergraduate dental students. **Revista de Odontologia da Unesp**, Ponta Grossa, v. 47, n. 3, p. 175-182, jun. 2018. FapUNIFESP (SciELO). http://dx.doi.org/10.1590/1807-2577.05118. Disponível em: https://www.researchgate.net/publication/326659544_Learning_styles_preferences_and_e-learning_experience_of_undergraduate_dental_students. Acesso em: 8 abr. 2020.

DATACAMP TEAM. **Python Arrays**. 2020. Disponível em: https://www.datacamp.com/community/tutorials/python-arrays?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631748&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=332602034358&utm_targetid=aud-438999696719:dsa-429603003980&utm_loc_interest_ms=&utm_loc_physical_ms=1031867&gclid=CjwKCAjwtNf6BRAwEiwAkt6UQvYog_2SGlaZvjEB1Mqnr9eb8y-mMBQz7IBuPyNTvB8qgLF8KlQ8eBoCQx8QAvD_BwE. Acesso em: 07 set. 2020.

DATA SCIENCE ACADEMY. **Deep Learning Book.** 2019. Disponível em: http://www.deeplearningbook.com.br/. Acesso em: 28 out. 2020.

DRAY. **The Learning Object Metadata standard**. 2005. Disponível em: https://www.ieeeltsc.org/working-groups/wg12LOM/lomDescription/. Acesso em: 4 set. 2020.

GHIRARDINI, Beatrice. **E-learning methodologies**: a guide for designing and developing e-learning courses. Rome: Food And Agriculture Organization, 2011. 138 p.

GHISI, Me. Clarice. The Construction of Learning Object: tools for the teaching: Tools for the Teaching. **Sino-us English Teaching**, v. 13, n. 8, p. 627-643, 8 ago. 2016. David Publishing Company. http://dx.doi.org/10.17265/1539-8072/2016.08.006.

GOODFELLOW, Ian; BENGIO, Yoshua; COURVILLE, Aaron. **Deep Learning:** an mit press book: Mit Press, 2016. Disponível em: https://www.deeplearningbook.org/. Acesso em: 27 out. 2020.

GRÜBLER, Murillo. **Entendendo o funcionamento de uma Rede Neural Artificial**. 2018. Disponível em: https://medium.com/brasil-ai/entendendo-o-funcionamento-de-uma-rede-neural-artificial-4463fcf44dd0. Acesso em: 03 maio 2020.

GUO, Yanhua; SUN, Lei; ZHANG, Zhihong; HE, Hong. Algorithm Research on Improving Activation Function of Convolutional Neural Networks. In: Chinese Control and Decision Conference, 31., 2019, Jiangxi. **Proceedings...** . Jiangxi: Ieee, 2019. p. 3582-3586.

HAYKIN, Simon. **Neural Networks:** a comprehensive foundation. 2. ed. Hamilton: Pearson Prentice Hall, 1999.

HNIDA, Meriem; IDRISSI, Mohammed Khalidi; BENNANI, Samir. Adaptive Teaching Learning Sequence based on Instructional Design and Evolutionary Computation. In: International Conference on Information Technology Based Higher Education and Training, 15., 2016, Istanbul. **Proceedings [...]** . Istanbul: Unesco, 2016. p. 1-6.
HOCHREITER, Sepp; SCHMIDHUBER, Jürgen. Long Short-Term Memory. **Neural Computation**, v. 9, n. 8, p. 1735-1780, nov. 1997. MIT Press - Journals. http://dx.doi.org/10.1162/neco.1997.9.8.1735. Disponível em: https://www.researchgate.net/publication/13853244_Long_Short-term_Memory. Acesso em: 16 jul. 2020.

IDOETA, Paula Adamo. **Como a Inteligência Artificial já está mudando salas de aula no Brasil e no mundo.** 2017. Disponível em: https://www.bbc.com/portuguese/geral-40969450. Acesso em: 12 dez. 2020.

ILLINGWORTH, W.t. Beginner's guide to neural networks. **Ieee Aerospace And Electronic Systems Magazine**, [s.l.], v. 4, n. 9, p. 44-49, set. 1989. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/62.35668.

JIUGEN, Yuan; RONGRONG, Kuang; RUONAN, Xing. Design and Development of SCORM-based Mobile Learning System. In: INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY IN MEDICINE AND EDUCATION, 8., 2016, Fuzhou. **Proceedings [...]** . Fuzhou: Ieee Computer Society, 2016. p. 482-485.

JAMES, Gareth; WITTEN, Daniela; HASTIE, Trevor; TIBSHIRANI, Robert. **An Introduction to Statistical Learning:** with applications in r. New York: Springer, 2017. Disponível em: http://faculty.marshall.usc.edu/gareth-james/ISL/. Acesso em: 28 out. 2020.

JCA SOLUTIONS. **Simple SCORM Packager**: with its easy to use wizard interface, Simple SCORM Packager (SSP) allows you to package an existing course into an AICC or SCORM conformant package in a matter of minutes. Disponível em: https://www.jcasolutions.com/products/ssp/. Acesso em: 18 jul. 2020.

KARPATHY, Andrej. **The Unreasonable Effectiveness of Recurrent Neural Networks**. 2015. Disponível em: http://karpathy.github.io/2015/05/21/rnn-effectiveness/. Acesso em: 01 ago. 2020.

KAZANIDIS, Ioannis; SATRATZEMI, Maya. Applying learning styles to SCORM compliant courses. In: International Conference on Advanced Learning Technologies, 9., 2009, Riga. **Proceedings[…]** . Riga: IEEE, 2009. p. 147-151.

KERAS DOCUMENTATION. **Keras**: The Python Deep Learning library. Disponível em: https://keras.io/. Acesso em: 04 maio 2020.

KULIKOV, A. S.; LEVIN; M.; KANE, D. M.; RHODES, N. **Estruturas de Dados e Algoritmos**. Plataforma Coursera, 2020. Disponível em: https:https://www.coursera.org/learn/data-structures?specialization=data-structures-algorithms#syllabus. Acesso em: 18 jun. 2020.

KUMAR, S. D.; SUBHA, D. P. Prediction of Depression from EEG Signal Using Long Short Term Memory(LSTM). In: International Conference on Trends In Electronics and Informatics, 3., 2019, Tirunelveli. **Proceedings[...]** . Tirunelveli: Ieee, 2019. p. 1248-1253.

LEARNBAY.CO - DATA SCIENCE TRAINING IN BANGALORE. **Data Structure And Its Importance.** 2007. Disponível em: https://medium.com/@learnbay/data-structure-and-its-importance-9b559ba3a062. Acesso em: 02 maio 2020.

MEDEIROS, Luciano Frontino de. **Inteligência Artificial Aplicada**: uma abordagem introdutória. Curitiba: Intersaberes, 2018.

MUSA, Abuagila M.; ARMESS, Aymen M.. An Investigation of the Effectiveness of Learning Sequence in E-Learning Environment. In: International Conference on Educational and Information Technology, 1., 2010, Chongqing. **Proceedings [...]**. Chongqing: Ieee, 2010. p. 1-4.

NARAGUND, Jayalakshmi G.; KOTRE, Prakash A.; KANAKARADDI, Suvarna; SUJATA, C.. Philosophy of Data Structures in Engineering Education. In: International Conference on Learning and Teaching in Computing and Engineering, 4., 2016, Mumbai. **Proceedings...** . Mumbai: Ieee Computer Society, 2016. p. 134-135.

NUMPY.ORG. **NumPy**. Disponível em: https://numpy.org/. Acesso em: 4 maio 2020.

RITZHAUPT, Albert D.. Learning Object Systems and Strategy: a description and discussion. : A Description and Discussion. **Interdisciplinary Journal Of E-learning And Learning Objects**., p. 217-238. jan. 2010. Disponível em: https://www.researchgate.net/publication/288964268_Learning_Object_Systems_and_Strategy_A_Description_and_Discussion. Acesso em: 03 maio 2020.

RUSTICI SOFTWARE. **SCORM Cloud:** test, play and distribute eLearning. Disponível em: https://rusticisoftware.com/products/scorm-cloud/?utm_source=google&utm_medium=natural_search. Acesso em: 22 abr. 2020.

RUSTICI SOFTWARE. **Technical SCORM:** A guide to SCORM 1.2 and SCORM 2004 for developers. Disponível em: https://scorm.com/scorm-explained/technical-scorm/. Acesso em: 22 abr. 2020.

SAVIC, Goran; KONJOVIC, Zora. Learning Style Based Personalization of SCORM E-learning Courses. In: INTERNATIONAL SYMPOSIUM ON INTELLIGENT SYSTEMS AND INFORMATICS, 7., 2009, Subotica. **Proceedings [...]** . Subotica: Ieee Xplore, 2009. p. 349-353. Disponível em: https://www.researchgate.net/publication/224606128_Learning_style_based_personalization_of_SCORM_E-learning_courses#fullTextFileContent. Acesso em: 18 jul. 2020.

SHAH, Dhawal. **By The Numbers:** MOOCS in 2017. 2018. Disponível em: https://www.classcentral.com/report/mooc-stats-2017/. Acesso em: 03 abr. 2020.

SILVEIRA, Antônio Claudio Jorge da; VIEIRA JUNIOR, Niltom. A inteligência artificial na educação: utilizações e possibilidades. **Revista Interritórios**, v. 5, n. 8, p. 206-217, 22 jun. 2019. Revista Interritorios. http://dx.doi.org/10.33052/inter.v5i8.241622. Disponível em: https://periodicos.ufpe.br/revistas/interritorios/article/view/241622/32622. Acesso em: 21 jun. 2020.

SIQUEIRA-BATISTA, Rodrigo; VITORINO, Rodrigo Roger; GOMES, Andréia Patrícia; OLIVEIRA, Alcione de Paiva; FERREIRA, Ricardo dos Santos; ESPERIDIÃO-ANTONIO, Vanderson; SANTANA, Luiz Alberto; CERQUEIRA, Fabio Ribeiro. As redes neurais artificiais e o ensino da medicina. **Revista Brasileira de Educação Médica**, v. 38, n. 4, p. 548-556, dez. 2014. FapUNIFESP (SciELO). http://dx.doi.org/10.1590/s0100-55022014000400017.

TENSORFLOW.ORG. **Introduction to Tensors:** about shapes. About shapes. Disponível em: https://www.tensorflow.org/guide/tensor#about_shapes. Acesso em: 08 set. 2020.

VAUGHN, Lisa; BAKER, Raymond. Teaching in the medical setting: balancing teaching styles, learning styles and teaching methods. : balancing teaching styles, learning styles and teaching methods. **Medical Teacher**, Cincinnati, v. 23, n. 6, p. 610-612, jan. 2001. Informa UK Limited. http://dx.doi.org/10.1080/01421590120091000.

WANG, Wei; WENG, Jui-feng; SU, Jun-ming; TSENG, Shian-shyong. Learning Portfolio Analysis and Mining in SCORM Compliant Environment. In: ANNUAL FRONTIERS IN EDUCATION, 34., 2004, Savannah. **Proceedings[...]** . Savannah: Asee/ieee, 2004. p. 17-24.

WATSON, Julie. A Case Study: Developing Learning Objects with an Explicit Learning Design. **Electronic Journal Of E-learning**., p. 41-50. jan. 2010.

ZAMBONI, Augusto; THOMMAZO, André di; HERNANDES, Elis Cristina; FABBRI, Sandra. StArt: uma ferramenta computacional de apoio à revisão sistemática. In: CONGRESSO BRASILEIRO DE SOFTWARE: TEORIA E PRÁTICA, 1., 2010, Salvador. **Anais [...]** . Salvador: Ufba, 2010.