Instituto de Pesquisas Tecnológicas do Estado de São Paulo

Mario Leandro Pires Toledo

Uso do histórico de navegação em redes neurais recorrentes para tratamento de *Cold-Start Problem* em sistemas de recomendação

São Paulo

2019

Mario Leandro Pires Toledo

Uso do histórico de navegação em redes neurais recorrentes para tratamento de Cold-Start Problem em sistemas de recomendação

Dissertação de Mestrado apresentada ao Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT, como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Computação

Data da aprovação ____/_____/_____

_____

Prof. Dr. Marcelo Novaes de Rezende (Orientador)
IPT – Instituto de Pesquisas Tecnológicas do Estado de São Paulo

Membros da Banca Examinadora:

Prof. Dr. Marcelo Novaes de Rezende (Orientador)
IPT – Instituto de Pesquisas Tecnológicas do Estado de São Paulo

Prof. Dr. Claudio Luiz Larieira (Membro)
IPT – Instituto de Pesquisas Tecnológicas do Estado de São Paulo

Prof. Dr. Aparecido Valdemir de Freitas (Membro)
Universidade Federal de São Caetano do Sul

Mario Leandro Pires Toledo

Uso do histórico de navegação em redes neurais recorrentes para tratamento de *Cold-Start Problem* em sistemas de recomendação

Exame de Defesa apresentado ao Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT, como parte dos requisitos para a obtenção do título de Mestre em Engenharia de Computação.
Área de Concentração: Engenharia de Software

Orientador: Marcelo Novaes de Rezende

São Paulo
2019

Ficha Catalográfica
Elaborada pelo Departamento de Acervo e Informação Tecnológica – DAIT
do Instituto de Pesquisas Tecnológicas do Estado de São Paulo - IPT

T649u **Toledo, Mario Leandro Pires**
Uso do histórico de navegação em redes neurais recorrentes para tratamento de Cold-Start Problem em sistemas de recomendação. / Mario Leandro Pires Toledo. São Paulo, 2019.
139p.

Dissertação (Mestrado em Engenharia de Computação) - Instituto de Pesquisas Tecnológicas do Estado de São Paulo. Área de concentração: Engenharia de Software.

Orientador: Prof. Dr. Marcelo Novaes de Rezende

1. Sistemas de recomendação 2. Histórico de navegação 3. Cold-Start Problem 4. Redes neurais recorrentes 5. Tese I. Rezende, Marcelo Novaes de, orient. II. IPT. Coordenadoria de Ensino Tecnológico III. Título

2020-13 CDU 004.41(043)

Bibliotecária responsável: Maria Darci Cornellas Narciso – CRB 8/3569

Dedico este trabalho aos meus professores e mestres que, ao longo desta grande jornada de aprendizado, nunca deixaram de acreditar no potencial de seu aluno.

AGRADECIMENTOS

Primeiramente, agradeço aos meus pais por todo esforço realizado durante todos estes anos para me oferecer as oportunidades que tive para chegar até aqui.

A Danielle Santos, agradeço por toda a compaixão e compreensão, e por ser a melhor companheira que alguém pode ter ao seu lado.

Aos meus professores, obrigado por todo o aprendizado dentro e fora da sala de aula.

Aos colegas de turma, agradeço pelo apoio mútuo oferecido e pela troca de experiências durante o mestrado.

Ao meu orientador, Marcelo Rezende, obrigado pelo suporte e pelas dicas oferecidas durante o desenvolvimento deste trabalho.

Aos profissionais do IPT, agradeço ao atendimento, profissionalismo e dedicação oferecidos durante o tempo de mestrado.

"Aqui, no entanto, nós não olhamos para trás por muito tempo.
Nós continuamos seguindo em frente,
abrindo novas portas e fazendo coisas novas, porque somos curiosos...
e a curiosidade continua nos conduzindo por novos caminhos."
(Walt Disney)

RESUMO

Cold-Start Problem é um problema recorrente em Sistemas de Recomendação nas seguintes situações: quando um novo item é adicionado ao sistema e não possui nenhuma avaliação prévia; ou quando um usuário sem histórico de avaliação entra no sistema. Avaliando as diferentes situações em que o *Cold-Start Problem* se apresenta, é possível considerar o uso do histórico de navegação como alternativa para geração de recomendações. Levando em conta o formato sequencial do dados, estudos sugerem o uso de Redes Neurais Recorrentes (RNN) por permitir maior entendimento da sequência de dados e seu contexto. Durante a revisão sistemática realizada neste trabalho, as arquiteturas de LSTM, GRU e híbridas aparecem com frequência entre as pesquisas relacionadas ao tema. Entretanto, os autores dos trabalhos revisados não comparam as arquiteturas entre si, o que é crucial para o entendimento das vantagens e desvantagens do uso de dados do histórico de navegação com RNN. Este estudo propõe a comparação das arquiteturas de LSTM, GRU e híbridas de RNN através da criação de protótipos utilizando a mesma base de entrada, avaliando suas performances através dos valores de *Accuracy*, *Recall*, *Precision* e *F1-Score*. Diferentes experimentos foram realizados ao alterar a função de ativação, o otimizador, o tamanho do *batch* e a quantidade de épocas utilizadas durante o treinamento das redes. Os resultados são então comparados entre si e com os resultados de outros autores, identificando o uso de arquiteturas de GRU como a mais vantajosa dentro do contexto deste trabalho a partir das variáveis e do conjunto de dados de entrada utilizados.

Palavras-chaves: sistemas de recomendação; histórico de navegação; cold-start problem; redes neurais recorrentes;

ABSTRACT

Cold-Start Problem is a recurring problem in Recommendation Systems in the following situations: when a new item is added to the system and has no prior evaluation; or when a user with no valuation history enters the system. Assessing the different situations in which Cold-Start Problem presents itself, it is possible to consider the use of navigation history as an alternative to generate recommendations. Taking into account the sequential format of the data, studies suggests the use of Recurrent Neural Networks (RNN) because it allows a better understanding of the data sequence and its context. During the systematic review performed in this study, LSTM, GRU and hybrid architectures often appear among research related to the theme. However, the authors of the reviewed studies does not compares the architectures with each other, which is crucial for understanding the advantages and disadvantages of using RNN with the navigation history data. This study proposes the comparison of LSTM, GRU and hybrid architectures for RNN by the use of prototypes with the same input base, evaluating their performance through the Accuracy, Recall, Precision and F1-Score values. Different experiments were performed by changing the activation function, the optimizer, the batch size and the number of epochs used during the training of the networks. The results are then compared with each other and with the results of other authors, identifying the use of GRU architecture as the most advantageous within the context of this work from the variables and input dataset used.

Keywords: recommendation systems; user navigation history; cold-start problem; recurrent neural networks;

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

MAE	Erro Médio Absoluto
MLP	Modelo de Multicamadas
RMSE	Raiz do Erro Médio Quadrático
RNN	Redes Neurais Recorrentes
RNA	Redes Neurais Artificiais
SR	Sistemas de Recomendação
SRN	Redes Recorrentes Simples

SUMÁRIO

1 INTRODUÇÃO

1.1 Motivação

A evolução da Internet trouxe um crescimento exponencial nos dados disponibilizados através da *World Wide Web*. De acordo com a NetCraft (2018), foram identificadas mais de 1,5 bilhão de páginas da Internet em mais de 200 milhões de domínios únicos, o que gera uma enorme concorrência durante a tentativa de obtenção de informações a partir de mecanismos de busca. Isso acontece devido aos sistemas tradicionais de busca não fornecerem um serviço individualizado que considere o perfil do usuário, obtendo resultados independentemente de seus interesses pessoais Deng, Huang e Xu (2014).

Sites como o Netflix, um dos principais serviços on-line de transmissão de filmes e séries de TV por assinatura, e Amazon, uma das principais lojas virtuais da Internet, possuem milhares de itens em seus catálogos, criando uma grande sobrecarga de conteúdos disponíveis para a escolha do usuário. Para tratar esse problema, ambos os sites utilizam de Sistemas de Recomendação Wei et al. (2017), oferecendo sugestões de itens disponíveis diretamente em seus catálogos.

Sistemas de Recomendação (RS) têm como objetivo sugerir itens que possam ser relevantes ao usuário de acordo com suas preferências (DENG; HUANG; XU, 2014). Seu princípio considera que a obtenção de recomendação através de fontes confiáveis é um componente crítico do processo natural de decisão humana. Dessa forma, os Sistemas de Recomendação assumem o papel de oferecer ao usuário indicações de itens que possam ser de seu interesse.

Tais sistemas possuem uso nos mais diferentes tipos de aplicações. Sistemas de comércio eletrônico, por exemplo, podem sugerir itens de compra ao usuário considerando seu histórico de compra e identificando, assim, itens de interesse que possam se relacionar com compras que foram efetuadas no passado Wei et al. (2017), enquanto *sites* de notícia podem, por exemplo, recomendar assuntos relacionados a outros conteúdos acessados anteriormente pelo usuário.

Segundo Shah et al. (2017), entre os tipos mais utilizados de Sistemas de Recomendação estão os Sistemas de Recomendação por Filtragem Colaborativa (*Colaborative Filtering*), que realizam recomendações através da similaridade entre os usuários e os Sistemas de Recomendação Baseados em Conteúdo (*Content-Based*), ou seja, aqueles que realizam recomendações através da similaridade entre itens. Ambos os sistemas dependem de informações prévias do usuário para que as recomendações possam ser feitas. No caso da Netflix, dados como avaliações de

filmes feitas pelo usuário ou metadados de filmes assistidos pelo usuário são utilizados para a recomendação personalizada.

Entretanto, a escassez desses dados influencia diretamente na qualidade das recomendações. Ao se tentar recomendar itens para um usuário do qual não se tem informações prévias, os Sistemas de Recomendação podem gerar recomendações que não se relacionem com o perfil de interesse do usuário ou não gerar recomendação alguma. Esse tipo de problema é identificado como *Cold-Start Problem*, aparecendo com frequência nos estudos relacionados a Sistemas de Recomendação (NADIMI-SHAHRAKI; BAHADORPOUR, 2014; DENG; HUANG; XU, 2014; GOMEZ-URIBE; HUNT, 2015; SHAH et al., 2017; PANDEY; RAJPOOT, 2016).

De acordo com Pandey e Rajpoot (2016), o *Cold-Start Problem* acontece quando o sistema não consegue gerar recomendações de itens em duas situações específicas: quando um novo item é adicionado ao Sistema de Recomendação e não possui nenhuma avaliação; ou quando um usuário sem histórico de avaliação entra no sistema. Para Lika, Kolomvatsos e Hadjiefthymiades (2014), o *Cold-Start Problem* é comum em Sistemas de Recomendação, sendo também o aumento da qualidade das recomendações em situações de *Cold-Start* um desafio-chave para sistemas do tipo.

Uma possível alternativa adotada para se lidar com o *Cold-Start Problem* é solicitar que o usuário avalie itens no início do uso do sistema (MASTHOFF, 2011), ou forçar esse usuário a responder perguntas que permitam associá-lo a estereótipos ou a outros conjuntos de usuários (como dados demográficos ou de idade, por exemplo). Nadimi-Shahraki e Bahadorpour (2014) sugerem um método que obrigue o novo usuário a avaliar itens previamente, antes de entrar no sistema. Enquanto esta é considerada uma das formas mais diretas para se obter informações do usuário, é necessária que haja uma estratégia bem definida para que os itens disponibilizados para seleção pelo usuário possam de fato aumentar a qualidade da recomendação. Entretanto, com esse método, existe a possibilidade de insatisfação do usuário por conta da necessidade de uma avaliação inicial, já que isso exige seu esforço, podendo levar o cliente a avaliar os itens de forma incorreta ou a desistir do uso do sistema.

Trabalhos como de Pandey e Rajpoot (2016) sugerem o uso de metadados do usuário, como dados demográficos, para a recomendação de itens por similaridade de usuário. Porém, sistemas web que não obrigam o registro do usuário para seu uso tendem a ter pouco ou quase nenhum dado para agrupamento de usuários similares, impossibilitando, assim, a recomendação personalizada de itens. Como alternativa, sistemas de comércio eletrônico utilizam o histórico de navegação do usuário para identificar padrões de compras e, dessa maneira, sugerir itens similares aos itens comprados (CHOU et al., 2010).

Avaliando essas situações, é possível considerar como uma alternativa para

lidar com o *Cold-Start Problem* a recomendação de itens para o usuário conforme seu histórico de navegação. Dessa forma, é possível realizar recomendações sem o uso de dados do perfil do usuário (por exemplo, sem que o usuário tenha realizado um cadastro no sistema).

O uso do histórico de navegação do usuário para recomendação exige a identificação de padrões a partir da páginas ou *sites* navegados pelo usuário. Por exemplo, se uma página X foi acessada com frequência antes dos usuários acessarem a página Y, então a página Y tem maior relevância na recomendação para usuários que acessaram a página X.

O reconhecimento de padrões é uma área relacionada a Inteligência Artificial e Aprendizado de Máquina, onde algoritmos de aprendizado são utilizados para identificar padrões em dados de entrada de forma automatizada. Nesse contexto, as Redes Neurais se mostram como uma das alternativas mais conhecidas para reconhecimento de padrões, podendo aprender a classificar dados de acordo com um conjunto de treinamento através do aprendizado supervisionado (BISHOP, 2006).

No caso, Redes de Aprendizado Profundo (*Deep Learning Networks*) têm obtido sucesso em tarefas de reconhecimento de imagem, de falas e de texto, nas quais dados não estruturados são processados através de diversas camadas convolucionais, atraindo o interesse de pesquisas para uso de Aprendizado Profundo em Sistemas de Recomendação (TAN; XU; LIU, 2016; WEI et al., 2017; HEINZ; BRACHER; VOLLGRAF, 2017; Chu et al., 2017; FU et al., 2018).

Considerando que a principal característica dos dados de histórico de navegação é o seu formato como dados sequenciais (ou seja, a navegação entre uma página e outra deve ser levada em consideração), estudos como os de Hidasi et al. (2015) sugerem o uso de Redes Neurais Recorrentes devido ao tipo de entrada do sistema. Nesse processo, cada estado de navegação do usuário – cada página visitada, por exemplo – produzirá uma saída (no caso, uma recomendação) que depende dos estados anteriores.

Por possuir uma memória interna capaz de armazenar entradas anteriores, as Redes Neurais Recorrentes permitem um maior entendimento da sequência de dados e de seu contexto se comparado a outros algoritmos. Dessa forma, aplicar o histórico de navegação como entrada de uma rede neural recorrente permite a geração de recomendações independentemente do histórico de avaliações ou dados de perfil do usuário (TAN; XU; LIU, 2016).

1.2 Problema

Apesar de estudos como o de Tan, Xu e Liu (2016) fortalecerem o uso de Redes Neurais Recorrentes como alternativa para Sistemas de Recomendação a partir do histórico do usuário, ainda é necessário entender quais arquiteturas de Redes Neurais Recorrentes podem se adequar neste contexto.

Através da revisão sistemática realizada neste trabalho, foram identificados três tipos de arquiteturas que aparecem com frequência nos estudos obtidos: Redes de Memória de Longo Prazo (LSTM); Redes de Unidades Recorrentes de Bloqueio (GRU); e Redes Híbridas, que constituem uma arquitetura que utiliza ambas as redes citadas anteriormente. Muitos desses estudos mostram propostas de novas arquiteturas baseadas nas arquiteturas citadas, comparando os resultados através de diferentes métricas de avaliação.

Porém, os trabalhos revisados não comparam as arquiteturas entre si, ou seja: não foi possível identificar um trabalho que realize a comparação dos resultados entre uma rede que utilize Memória de Longo Prazo com uma que utilize Unidades Recorrentes de Bloqueio ou Híbrida. Tal comparação é crucial para entender as vantagens e as desvantagens do uso de cada arquitetura para o contexto da recomendação por dados do histórico de navegação.

Logo, surge a definição do seguinte problema:

- Falta de avaliações que comparem os resultados de Redes de Memória de Longo Prazo, Redes de Unidades Recorrentes de Bloqueio e Redes Híbridas para recomendação a partir do histórico de navegação;

1.3 Objetivo

Considerando os problemas listados no item anterior, este trabalho tem o seguinte objetivo primário:

- Avaliar a eficiência de Redes de Memória de Longo Prazo, de Redes de Unidades Recorrentes de Bloqueio e de Redes Híbridas no contexto da recomendação de itens através do histórico de navegação do usuário;

Para que os resultados das redes sejam avaliados e comparados, é necessária a construção de protótipos que utilizem cada uma das arquiteturas de Redes Neurais Recorrentes citadas no objetivo primário. Tais protótipos devem utilizar a mesma base de dados como insumo e terem as mesmas métricas utilizadas para a avaliação dos resultados obtidos.

Dessa forma, é levantado o seguinte objetivo específico:

- Construção de protótipos que utilizem arquiteturas de Redes de Memória de Longo Prazo, de Redes de Unidades Recorrentes de Bloqueio e de Redes Híbridas para que seja possível realizar a avaliação dos resultados

Para a avaliação dos resultados de cada protótipo, serão utilizadas as métricas de *Accuracy*, *Recall*, *Precision* e *F1-Score*, identificadas com frequência entre os estudos obtidos na revisão sistemática deste trabalho. Como insumo do sistema de recomendação, será utilizado a base de dados de um site chinês de *live streaming*, conforme proposto no trabalho de (FU et al., 2018).

1.4 Contribuição

Estudos como o de Nadimi-Shahraki e Bahadorpour (2014) exigem que o usuário responda uma série de perguntas antes do uso do sistema, enquanto outros trabalhos como o de Pandey e Rajpoot (2016) utilizam dados demográficos obtidos pelo usuário. Porém, o usuário pode se insatisfazer pela demora do tempo da resposta ou pelo grande número de perguntas realizadas, podendo ainda não liberar o uso de informações demográficas e outros tipos de dados que considere confidenciais.

Diferente desses processos, o método adotado por este trabalho não exige o uso de perguntas ou informações extras do usuário, exceto seu histórico de navegação que será armazenado conforme o indivíduo utilizar o sistema. Sendo assim, esse método contribui também para a implantação de sistemas de recomendação sem a necessidade de dados pessoais do usuário.

A revisão sistemática realizada para este trabalho também pode servir como contribuição para outros trabalhos similares, permitindo, assim, identificar os principais autores e estudos ligados ao uso de Redes Neurais em Sistemas de Recomendação e possibilitando a expansão dos estudos através da modificação dos protocolos anexados.

Por fim, a construção de protótipos e a análise dos resultados permitirá identificar as vantagens e as desvantagens entre Redes de Memória de Longo Prazo, Redes de Unidades Recorrentes de Bloqueio e Redes Híbridas dentro do contexto de recomendações a partir do uso do histórico de navegação do usuário, podendo contribuir para estudos ou para a implantação de sistemas que estejam dentro deste tema.

1.5 Metodologia

A seguir estão descritas as atividades necessárias para a realização desta pesquisa, bem como seu referencial teórico:

- Revisão bibliográfica: a revisão bibliográfica consiste no levantamento bibliográfico dos principais fundamentos relacionados a este projeto. Os estudos utilizados nesta revisão foram selecionados a partir de bases de pesquisas de grande relevância para a área científica, relacionando os seguintes tópicos: Sistemas de Recomendação e seus diferentes tipos; Métricas de Avaliação de Sistemas de Recomendação; *Cold-Start Problem*; Aprendizado de Máquina; Redes Neurais; Redes Neurais Recorrentes; e Histórico de Navegação.

- Revisão sistemática: uma revisão sistemática foi realizada para a construção do Estado da Arte deste trabalho a partir de um protocolo definido no Apêndice A, com o objetivo de identificar o uso de Histórico de Navegação em Redes Neurais Recorrentes, as arquiteturas de Redes Neurais Recorrentes mais frequentemente utilizadas e as Métricas de Avaliação mais comuns para Redes Neurais Recorrentes.

- Definição e aplicação da arquitetura proposta: a partir dos resultados obtidos na revisão para o Estado da Arte, os trabalhos de (ANIL et al., 2018), (RUOCCO; SKREDE; LANGSETH, 2017), (KONG; LI; LV, 2018), (KOROTAEV; LYADOVA, 2018), (FU et al., 2018) e (Chu et al., 2017) foram utilizados para propor protótipos de uma Rede de Memória de Longo Prazo, de uma Rede de Unidades Recorrentes de Bloqueio e de uma rede híbrida que unifique ambos os modelos.

- Aplicação da arquitetura proposta: foram desenvolvidos protótipos de Sistemas de Recomendação que utilizem as arquiteturas de Rede de Memória de Longo Prazo, de Rede de Unidades Recorrentes de Bloqueio e de Redes Híbridas. Como insumo do sistema de recomendação, foi utilizada a base de dados do Live Streaming, conforme proposto no trabalho de (FU et al., 2018).

- Avaliação dos resultados: os resultados das recomendações de cada modelo foram avaliados a partir das métricas de *Accuracy*, de *Recall*, de *Precision* e de *F1-Score*, identificadas com frequência nos demais trabalhos revisados dentro do Estado da Arte.

- Conclusão e trabalhos futuros: apresentação da conclusão do trabalho e sua continuidade para melhoria em trabalhos futuros;

2 REVISÃO BIBLIOGRÁFICA

2.1 Conceito

Esta seção conta com os principais conceitos e fundamentos sobre os demais itens de interesse explorados neste trabalho. Nos itens abaixo, são definidos os conceitos de Sistemas de Recomendação, juntamente com suas principais aplicações e os problemas mais comuns encontrados. Também são definidos os principais conceitos de Aprendizado de Máquina, de Redes Neurais e de Redes Neurais Recorrentes, além das suas aplicações em Sistemas de Recomendação.

2.2 Sistemas de Recomendação

Sistemas de Recomendação (SR) têm o objetivo de sugerir itens ao usuário a partir do uso de técnicas e ferramentas de forma que possa orientar sua tomada de decisão (RICCI et al., 2011). Apesar da área de pesquisa existir desde meados dos anos 90 (KUMAR et al., 1998), os Sistemas de Recomendação têm se tornado cada vez mais populares atualmente em sistemas web, especialmente em sistemas de comércio eletrônico, oferecendo ao usuário itens que ele possa ter interesse. Sites como Facebook ou Amazon, por exemplo, utilizam Sistemas de Recomendação para sugerir novos itens de compras ou para a sugestão de outros usuários com gostos similares (RICCI et al., 2011).

Em seu trabalho, Ricci et al. (2011) citam alguns dos motivos pelos quais os provedores de serviço utilizam os SR. São eles:

- Aumento do número de itens vendidos: a venda de itens adicionais baseados nos gostos do usuário;
- Vendas de itens mais diversos: seleção de itens que podem ser difíceis de serem encontrados devido ao tamanho do catálogo;
- Aumento da satisfação do usuário: caso o usuário se interesse pelas recomendações do sistema, ele poderá ter um aumento de satisfação com o sistema;
- Aumento da fidelidade do usuário: caso o usuário entenda as recomendações como relevantes, ele passará a confiar mais no sistema que está utilizando;
- Melhor entendimento do que o usuário quer: coleta de informações das preferências do usuário de forma explícita ou preditas pelo sistema.

A princípio, os SR partem da premissa de que um usuário tem o costume de optar por recomendações de outras pessoas durante uma tomada de decisão. Dessa forma, os algoritmos voltados para SR foram originados para alavancar recomendações produzidas por uma comunidade de usuários para um usuário específico (RICCI et al., 2011). Segundo Shah et al. (2017), as recomendações podem ser classificadas em dois tipos: personalizadas, nas quais diferentes usuários recebem diferentes recomendações; e não-personalizadas, nas quais diferentes usuários recebem as mesmas recomendações.

Existem diferentes tipos de SR que diferem entre o domínio, o aprendizado utilizado e o algoritmo implementado (RICCI et al., 2011). Enquanto alguns estudos apresentam apenas as classes de sistemas baseadas em conteúdo ou em filtragem colaborativa ((SHAH et al., 2017), (KHATWANI; CHANDAK, 2016)), Ricci et al. (2011) citam a seguinte classificação entre os SR:

- Baseados em conteúdos;
- Filtragem colaborativa;
- Demográficos;
- Baseados em conhecimento;
- Baseados em comunidade;
- Híbridos.

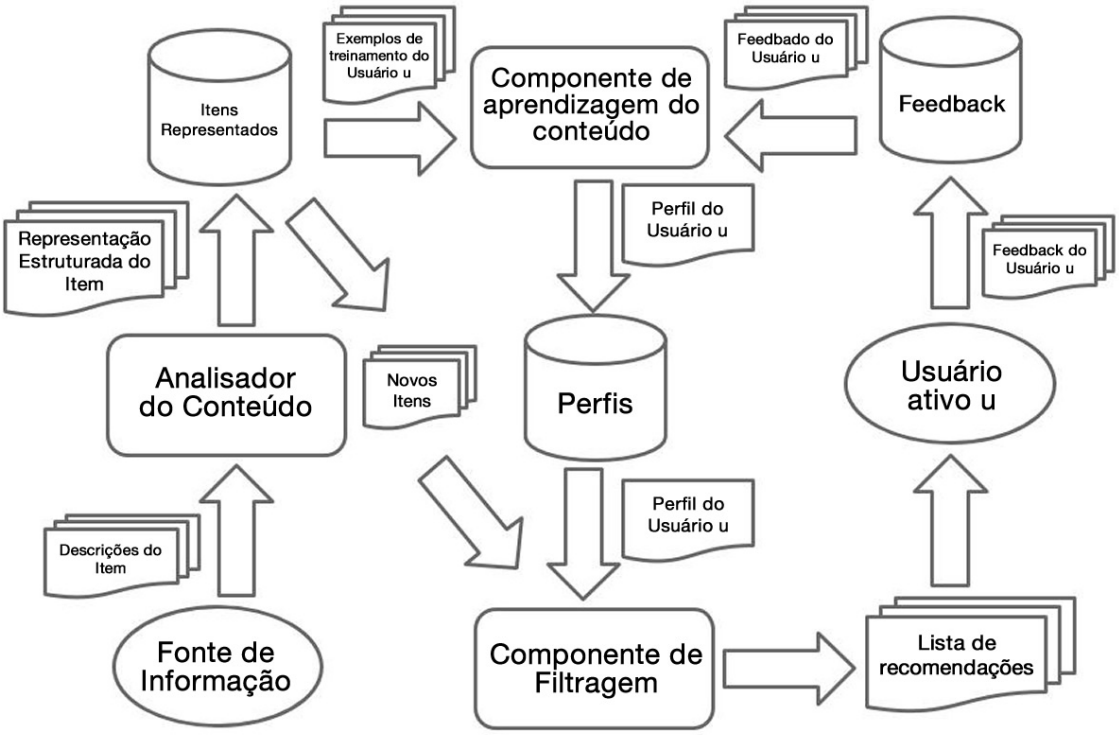
2.2.1 Sistemas de Recomendação Baseados em conteúdos

Sistemas de Recomendação baseados em conteúdos têm como objetivo a recomendação de itens similares aos que o usuário gostou, avaliou ou interagiu no passado, analisando a descrição de documentos ou de itens no histórico de interação do usuário e construindo um modelo ou perfil de itens de interesses (LOPS; GEMMIS; SEMERARO, 2011). O processo consiste na relação de atributos do usuário a atributos de um item, gerando, dessa forma, um conjunto de itens que refletem melhor as preferências do usuário.

Segundo Lops, Gemmis e Semeraro (2011), são necessárias técnicas e estratégias específicas para que o sistema seja capaz de relacionar o perfil de um usuário a uma representação de itens, descrevendo três passos para que a recomendação possa ser realizada: a análise de conteúdo, que pré-processa o conteúdo adquirido para formatá-lo adequadamente para as próximas etapas; o aprendizado de perfil, que generaliza a informação para construir um modelo de interesses do usuário; e o

componente de filtragem, que utiliza o modelo de interesses do usuário para relacionar com os itens do sistema e gerar as recomendações;

Figura 1 – Arquitetura de um sistema de recomendação por conteúdo.



Fonte: Lops, Gemmis e Semeraro (2011)

2.2.2 Sistemas de Recomendação por Filtragem Colaborativa

Filtragem Colaborativa é considerada uma das técnicas de recomendação mais famosas e usadas em Sistemas de Recomendação (RICCI et al., 2011). Seu conceito se baseia no uso do histórico da pontuação dada por um usuário para um ou mais itens, a fim de identificar alguma similaridade entre os usuários.

A principal suposição da abordagem de filtragem colaborativa é de que se um usuário A possui gostos similares ao do usuário B em um determinado assunto, então A tem mais chances de ter gostos similares ao B em outros assuntos do que um usuário aleatório. Por exemplo, se os usuários A e B avaliam positivamente um item x, e B avalia positivamente um item y, então A tem grandes chances de também ter y como recomendação.

Segundo Yao et al. (2015), os métodos de filtragem colaborativa podem ser agrupados em duas classes principais: métodos baseados em memória e métodos baseados em modelos.

2.2.2.1 Filtragem Colaborativa Baseada em Memória

Métodos de filtragem colaborativa baseados em memória utilizam as avaliações fornecidas por um usuário em relação a um item específico, que são armazenadas em um banco de dados (ZHANG et al., 2014). Em essência, tais algoritmos se baseiam no conceito de recomendação por "boca a boca", na qual uma pessoa confia em uma recomendação de outra pessoa que tenha gostos parecidos com o seu.

De acordo com Al-bashiri et al. (2018), a abordagem dos algoritmos de filtragem colaborativa baseados em memória é de computar a relação entre usuários e os itens para produzir uma pontuação que prediz a chance de um usuário adquirir um item no futuro e de prover recomendações correspondentes. Entre os principais métodos de filtragem colaborativa baseado em memória, está a recomendação baseada em usuário, que gera recomendações de acordo com a similaridade entre usuários, e a baseada em item, que computa similaridades entre um espaço de itens para encontrar relações fortes de itens que já tenham sido avaliados pelo usuário.

2.2.2.1.1 Sistemas de Recomendação por Filtragem Colaborativa Baseada em Usuário

A principal abordagem dos algoritmos de filtragem colaborativa baseada em usuário é prever a avaliação de um usuário para um item que ele ainda não avaliou, de acordo com seu histórico de avaliações e de preferências, comparando sua relação com outros usuários de uma mesma base de dados.

Supondo a predição da preferência de um usuário u por um item i, que ele não avaliou, é levado em consideração a matriz de avaliação N, dada pela avaliação de cada usuário para cada item da base de dados.

Tabela 1 – Exemplo da matriz de avaliação, em uma escala de 1 a 5 pontos. Valores demarcados por '?' representam itens não avaliados pelo usuário.

	Batman Begins	**Alice in Wonderland**	**Dumb and Dumber**	**Equilibrium**
Usuário A	4	?	3	5
Usuário B	?	5	4	?
Usuário C	5	4	2	?
Usuário D	2	4	?	3

Usuário E	3	4	5	?

Fonte: Ekstrand, Riedl e Konstan (2011). Adaptado pelo autor.

A partir da matriz de avaliação N, é necessária a geração de uma matriz M de similaridade entre usuários, definida a partir de uma métrica de distância que mede a similaridade entre dois usuários. Segundo Ekstrand, Riedl e Konstan (2011), o método de *Pearson Correlation* demonstra os melhores resultados, computando a correlação estatística entre avaliações em comum de dois usuários para determinar sua similaridade.

$$p_{X,Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y} \tag{2.1}$$

onde:

(X, Y) define um par de variáveis aleatórias

σ_X é a variância padrão de σ_X

σ_Y é a variância padrão de σ_Y

μ_X é a média de μ_X

μ_Y é a média de μ_Y

E é a esperança

Através da matriz M de similaridade entre usuários, é possível realizar a predição da avaliação de itens não avaliados por um usuário. Isso é feito a partir da média ponderada da soma da avaliação de todos os outros usuários, onde o peso é definido pela similaridade entre os usuários.

$$p_{u,i} = \bar{r}_u + \frac{\sum u' \in N^{s(u,u')}(r_{u,u'} - \bar{r}'_u)}{\sum u' \in N^{|s(u,u')|}} \tag{2.2}$$

onde:

U é o conjunto de usuários da base

N é o conjunto de usuários vizinhos de u, onde $N \subseteq U$

$s(u, u')$ é o valor da similaridade entre o usuário u e o usuário u'

$\bar{r}_u$ é a média das avaliações dada pelo usuário u

$\bar{r}'_u$ é a média das avaliações dada pelo usuário u'

$\bar{r}_{u',i}$ é a avaliação dada pelo usuário u' ao item i

Com a predição das avaliações dos itens, pode-se ordenar os valores em ordem decrescente para recomendar os itens que tenham maior valor de avaliação, tendo assim maiores chances de se adequar às preferências do usuário.

2.2.2.1.2 Sistemas de Recomendação por Filtragem Colaborativa Baseada em Item

Diferente da Filtragem Colaborativa Baseada em Usuário, a Filtragem Colaborativa Baseada em Item tem como principal abordagem o cálculo da similaridade entre itens para a recomendação. Se tais itens tendem a ter os mesmos usuários que os avaliaram, então eles são similares e é esperado que os usuários tenham preferências por tais itens (EKSTRAND; RIEDL; KONSTAN, 2011).

A similaridade entre itens deve ser representada através de uma matriz esparsa N, na qual itens sem similaridade são representados pelo valor 0. Para tal, são utilizados os valores de recomendação entre dois itens a partir de usuários que avaliaram ambos e, em seguida, é utilizado o método de similaridade pela função cosseno, considerado o método mais eficiente para cálculo de similaridade na Filtragem Colaborativa Baseada em Item, conforme definida pela Equação 2.3.

$$cos(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||} \tag{2.3}$$

onde x e y representam vetores de tamanho dimensional n.

A predição de um item para um usuário é realizada através do cálculo da média ponderada da soma das avaliações dadas pelo usuário para os itens similares ao item desejado, onde o peso é definido pela similaridade entre os itens. Tal qual como o método de Filtragem Colaborativa Baseada em Usuário, pode-se ordenar os valores em ordem decrescente para recomendar os itens que tenham maior valor de avaliação.

Tabela 2 – Filmes avaliados e não avaliados pelo usuário, e seus respectivos valores de acordo com o método de similaridade pela função coseno.

Filme	$\|\|x\|\| \cdot \|\|y\|\|$	$x \cdot y$	$\frac{x \cdot y}{\|\|x\|\| \cdot \|\|y\|\|}$

Batman Begins (b)	7,348	24	0,607
Alice in Wonderland (a)	8,544	8	0,174
Dumb and Dumber (d)	7,348	15	0,382
Equilibrium (e)	5,385		

Fonte: Ekstrand, Riedl e Konstan (2011). Adaptado pelo autor.

2.2.2.2 Sistemas de Recomendação por Filtragem Colaborativa Baseada em Modelo

O método de Filtragem Colaborativa Baseada em Modelo utiliza as avaliações dos usuários para construir um modelo preditivo capaz de representar as características latentes entre usuários e itens, permitindo classificá-los em classes (KOREN; BELL, 2011). O modelo então é treinado com os dados disponíveis e usado para predizer as avaliações de usuários para novos itens.

Os autores Do, Nguyen e Nguyen (2010) citam cinco técnicas mais comuns para se trabalhar com Filtragem Colaborativa Baseada em Modelo: *Clustering, Classification*, *Latent Model*, *Markov Decision Process* (MDP) e *Matrix Factorization*. De acordo com Koren e Bell (2011), a técnica de *Matrix Factorization* ganhou popularidade graças a sua precisão e sua escalabilidade, se tornando um dos principais métodos para modelagem das preferências do usuário através da Decomposição de Valor Singular (SVD).

2.2.3 Sistemas de Recomendação Demográficos

Sistemas de Recomendação demográficos utilizam as informações demográficas do perfil do usuário para gerar recomendações, com a suposição de que diferentes recomendações devem ser geradas para diferentes nichos demográficos (RICCI et al., 2011). Já no estudo de Aggarwal (2016) são apresentados resultados de trabalhos relacionados a sistemas de recomendação demográficos, na qual uma página da Internet gera recomendações e guia a navegação do usuário a partir dos dados demográficos coletados, enquanto Pandey e Rajpoot (2016) apresentam uma pesquisa de *marketing*, mostrando que dados demográficos podem ser usados para recomendação de itens.

De acordo com Aggarwal (2016), Sistemas de Recomendação demográficos não produzem um resultado satisfatório quando são utilizados sozinhos, sendo melhor utilizados como componentes de sistemas híbridos, geralmente combinados com Sistemas de Recomendação Baseados em Conhecimento para aumentar sua robustez.

2.2.4 Sistemas de Recomendação Baseados em Conhecimento

Em Sistemas de Recomendação Baseados em Conhecimento, as avaliações do usuário não são levadas em consideração para gerar a recomendação. Em vez disso, as avaliações dependem de um processo que utiliza uma base de conhecimento para relacionar requisitos do usuário com descrições de itens e do uso de restrições que especificam os requisitos do usuário (AGGARWAL, 2016).

Segundo Ricci et al. (2011), esses tipos de SR dependem de uma função de similaridade para estimar o quanto a necessidade do usuário bate com as recomendações, de tal forma que a similaridade é mensurada como a utilidade da recomendação gerada para o usuário. Ricci et al. (2011) também citam que Sistemas de Recomendação Baseados em Conhecimento tendem a funcionar bem no começo, mas que podem ser superados por outros métodos caso não utilizem componentes de aprendizado.

2.2.5 Sistemas de Recomendação Baseados em Comunidade

Sistemas de Recomendação Baseados em Comunidade, também chamados de Sistemas de Recomendação Social, utilizam informações da relação social do usuário a partir das avaliações de seus amigos para, assim, gerar suas recomendações. A popularidade das redes sociais garantiu um crescente interesse em Sistemas de Recomendação Social, uma vez que dados de redes sociais permitem identificar a relação entre amigos para, em seguida, identificar possíveis recomendações (RICCI et al., 2011).

2.2.6 Sistemas de Recomendação Híbridos

Sistemas de Recomendação Híbridos são sistemas que combinam duas ou mais técnicas de recomendação, visando compensar as desvantagens de uma técnica com as vantagens de outra (RICCI et al., 2011). Um exemplo disso são os Sistemas de Recomendação por Filtragem Colaborativa Baseada em Item, que sofrem da falta de recomendação quando ninguém avaliou um item ainda e podem ser compensadas com recomendações baseadas em conteúdo ao utilizar similaridades de texto e encontrar semelhanças de itens a partir de metadados.

De acordo com com Ekstrand, Riedl e Konstan (2011), Sistemas de Recomendação Híbridos podem ser classificados da seguinte forma:

- Recomendação ponderada: adquire uma pontuação produzida por diferentes sistemas de recomendação e combina esses sistemas para gerar recomendações para o usuário;

- Recomendação alternada: alterna entre diferentes sistemas de recomendação e usa um sistema de recomendação específico para obter o melhor resultado dependendo do contexto;
- Recomendação mista: parecido com a recomendação ponderada, a recomendação mista apresenta o resultado de diferentes sistemas de recomendação juntos, porém podendo ou não os combinar em uma única listagem;
- Recomendação por combinação de recurso: usa múltiplas fontes de recomendação como entrada para um único sistema de metarrecomendação;
- Recomendação em cascata: encadeia a saída de um SR na entrada de outro;
- Recomendação por aumento de recurso: usa a saída de um SR como um recurso de entrada de outro;
- Recomendação por metanível: treina um modelo utilizando um SR e usa este modelo como entrada de outro SR;

2.2.7 Métricas de avaliação de sistemas de recomendação

A avaliação de SR é extremamente necessária quando se é desejado identificar a qualidade das recomendações geradas pelo sistema. De acordo com Anástacio (2018), a qualidade das recomendações é algo muito discutido dentro da literatura de SR, uma vez que o mecanismo de recomendação, capaz de prever opiniões entre usuários e itens, está presente na maioria dos SR.

Segundo Shani e Gunawardana (2011), a qualidade da avaliação é independente de uma interface para o usuário e pode ser medida por um experimento *offline*. O conceito de qualidade de uma recomendação é diferente da predição do comportamento de um usuário sem recomendações e é próximo da recomendação da verdadeira predição do sistema em si.

Entre as métricas de avaliação de precisão mais utilizadas, está o Erro Médio Absoluto (MAE) e a Raiz do Erro Médio Quadrático (RMSE), conforme descritos nos itens a seguir:

2.2.7.1 Erro Médio Absoluto

O Erro Médio Absoluto (MAE) é, essencialmente, uma medida da diferença entre duas variáveis contínuas de observações pareadas que representam o mesmo fenômeno. Considerando um gráfico de dispersão de n pontos, onde o ponto i possui as coordenadas (x_i, y_i), o MAE é a média da distância entre cada ponto e sua linha de origem (Willmott; Matsuura, 2005).

Em Sistemas de Recomendação, o MAE é a média da diferença absoluta entre valores de recomendações gerados pelo sistema e valores de recomendações de uma base de teste com valores reais de recomendação para um usuário (SHANI; GUNAWARDANA, 2011), conforme definido pela Equação 2.4.

$$MAE = \frac{1}{\tau} \sum_{(u,i) \in \tau} |\hat{r}_{ui} - r_{ui}| \tag{2.4}$$

onde:

$\hat{r}_{ui}$ é um conjunto de predições geradas pelo sistema

τ é um conjunto que representa a base de testes

(u, i) é um par de usuário-item

r_{ui} é um conjunto de predições reais de valores conhecidos

2.2.7.2 Raiz do Erro Médio Quadrático

A métrica de Raiz do Erro Médio Quadrático (RMSE), tal qual o MAE, também serve para medir as diferenças entre os valores preditos por um modelo e os valores observados, agregando a magnitude dos erros em predições em uma única medida.

Porém, diferente da métrica de MAE, o RMSE utiliza a raiz quadrada da média da diferença quadrática entre valores de recomendações gerados pelo sistema e valores de recomendações de uma base de teste com valores reais de recomendação para um usuário (SHANI; GUNAWARDANA, 2011), conforme definido pela Equação 2.5.

$$RMSE = \sqrt{\frac{1}{\tau} \sum_{(u,i) \in \tau} (\hat{r}_{ui} - r_{ui})^2} \tag{2.5}$$

onde:

$\hat{r}_{ui}$ é um conjunto de predições geradas pelo sistema

τ é um conjunto que representa a base de testes

(u, i) é um par de usuário-item

r_{ui} é um conjunto de predições reais de valores conhecidos

2.2.7.3 Raiz do Erro Médio Quadrático X Erro Médio Absoluto

De acordo com Shani e Gunawardana (2011), quando comparado ao MAE, o RMSE desproporcionalmente penaliza erros largos. Como os erros em RMSE são elevados antes de serem calculados, o RMSE pondera pesos maiores para erros maiores, indicando que o método pode ser mais útil quando erros maiores são particularmente indesejáveis, conforme apresentado na Tabela 3.

Tabela 3 – 5 casos hipotéticos de erro, e seus correspondentes totais, MAE e RMSE, onde cada e_j ($i = 1, 2, 3, 4$) é um erro hipotético

Variável	**Caso 1**	**Caso 2**	**Caso 3**	**Caso 4**	**Caso 5**
e_1	2	1	1	0	0
e_2	2	1	1	0	0
e_3	2	3	1	1	0
e_4	2	3	5	7	8
$\sum \lvert e_i \rvert$	8	8	8	8	8
MAE	2	2	2	2	2
$\sum \lvert e_i \rvert^2$	16	20	28	50	64
RMSE	2,0	2,2	2,6	3,5	4,0

Fonte: Willmott e Matsuura (2005). Adaptado pelo autor.

Enquanto parte da literatura sugere o uso de ambas as métricas, estudos como o de (Willmott; Matsuura, 2005) sugerem que eles devem ser interpretados de acordo com o tamanho dos erros de predição do modelo.

2.2.8 Limitações de métodos de Sistemas de Recomendação

A qualidade das recomendações em SR pode variar muito de acordo com as limitações dos diferentes tipos de SR. Segundo Shani e Gunawardana (2011), alguns sistemas podem prover recomendações com alta qualidade, mas apenas para uma pequena porção dos itens que eles possuem uma maior quantidade de informação.

Entre os problemas mais conhecidos na literatura de Sistemas de Recomendação, está o *Cold-Start Problem*, descrito em detalhes no item a seguir.

2.2.8.1 Cold-Start Problem

O *Cold-Start Problem* está relacionado ao problema de se gerar recomendações quando há uma falta de avaliações no sistema. Uma vez que não existem avaliações suficientes para se identificar itens ou usuários relacionados, o sistema fica impossibilitado de gerar recomendações personalizadas para um usuário.

De acordo com Bobadilla et al. (2012), o *Cold-Start Problem* acontece em três situações diferentes: com nova comunidade; com novos itens; ou com novos usuários, conforme detalhado nos itens a seguir.

2.2.8.1.1 Cold-Start Problem por nova comunidade

O *Cold-Start Problem* por nova comunidade é, essencialmente, o problema de se gerar recomendações quando o sistema é inicializado, não possuindo assim nenhum dado de avaliação que o Sistema de Recomendação possa utilizar para se gerar recomendações personalizadas. Neste caso, o sistema possui tanto as desvantagens do problema por novos itens quanto por novos usuários.

As soluções mais comuns adotadas neste tipo de situação são encorajar as avaliações através de outros meios (por exemplo, forçando o usuário a avaliar itens antes do uso do sistema) ou não utilizar Sistemas de Recomendação por Filtragem Colaborativa até que haja avaliações suficientes para a geração de recomendações.

2.2.8.1.2 Cold-Start Problem por novos itens

O problema de recomendação por novos itens acontece pelo fato de que itens que entraram em um sistema normalmente não possuem avaliações, o que impede a geração da recomendação pelo sistema. Dessa forma, um item que não é recomendado passa desapercebido por usuários do sistema que, por sua vez, não o recomenda por não ter noção da existência de tal item. Isso pode gerar um círculo vicioso, no qual um conjunto de itens nunca é recomendado pela falta de avaliações.

O trabalho de Hou, Pan e Liu (2018) coloca a falta de qualidade de avaliações de itens dentro do problema de *Cold-Start* por itens. Nesse caso, o termo "*popularity bias*" indica que itens que são mais populares tendem a receber maior interação que itens que não são populares, afetando suas avaliações. Nesse contexto, itens que são mais populares tendem a receber avaliações com maior qualidade que itens que são menos populares, dificultando o processo de recomendação do sistema.

De acordo com Madhukar (2014), o *Cold-Start Problem* para itens de um sistema pode ser resolvido através do uso de Sistemas de Recomendação Baseados em Conteúdo, uma vez que tais sistemas não dependem de recomendação e não são

afetados pelo número de interações em um item, podendo relacionar um item a outros de um mesmo sistema através de seus atributos. Já Bobadilla et al. (2012) sugerem o uso de um mesmo conjunto de usuários para sempre que houver a necessidade de avaliar itens novos ou com pouca popularidade.

2.2.8.1.3 Cold-Start Problem por novos usuários

O problema da recomendação para novos usuários é um dos maiores problemas de Sistemas de Recomendação (BOBADILLA et al., 2012). Ele ocorre quando um usuário ingressa em um sistema, mas ainda não deu avaliações o suficiente para o sistema gerar recomendações personalizadas. O problema pode infligir na perda de interesse do usuário pelo sistema, já que ele pode entender que este não está funcionando como deveria.

A estratégia mais comum para se lidar com *Cold-Start Problem* por novos usuários é solicitar ao usuário informações iniciais para, assim, gerar um perfil e obter informações de avaliação de outros usuários com atributos parecidos. Porém, a quantidade de informações adquiridas influencia diretamente na busca por usuários similares e o questionamento de muitas informações ao usuário pode desmotivá-lo a utilizar o sistema.

Outra estratégia comum baseia-se em solicitar que o usuário avalie itens iniciais para permitir o uso do sistema. No entanto, isso pode levar o cliente a avaliar itens que não condigam com sua real preferência ou itens os quais ele não obteve nenhuma interação, apenas para garantir o uso do sistema.

Pesquisas atuais apresentam o uso de técnicas de Aprendizado de Máquina (*Machine Learning*) para predizer a avaliação de um usuário, podendo inseri-lo em um cluster de outros usuários ou utilizando de fatorização de matrizes para criar uma matriz de relacionamento entre usuários e itens. Da mesma forma, sistemas híbridos ganharam popularidade na comunidade na tentativa de reduzir as desvantagens da filtragem colaborativa por usuários através de outros tipos de Sistemas de Recomendação.

2.3 Aprendizado de Máquina

Aprendizado de Máquina é uma área da Inteligência Artificial com o objetivo de desenvolver técnicas computacionais de aprendizado para criação de sistemas capazes de adquirir conhecimento de forma automática (MONARD; BARANAUSKAS, 2003), utilizando de experiências bem-sucedidas para a tomada de decisão em problemas futuros.

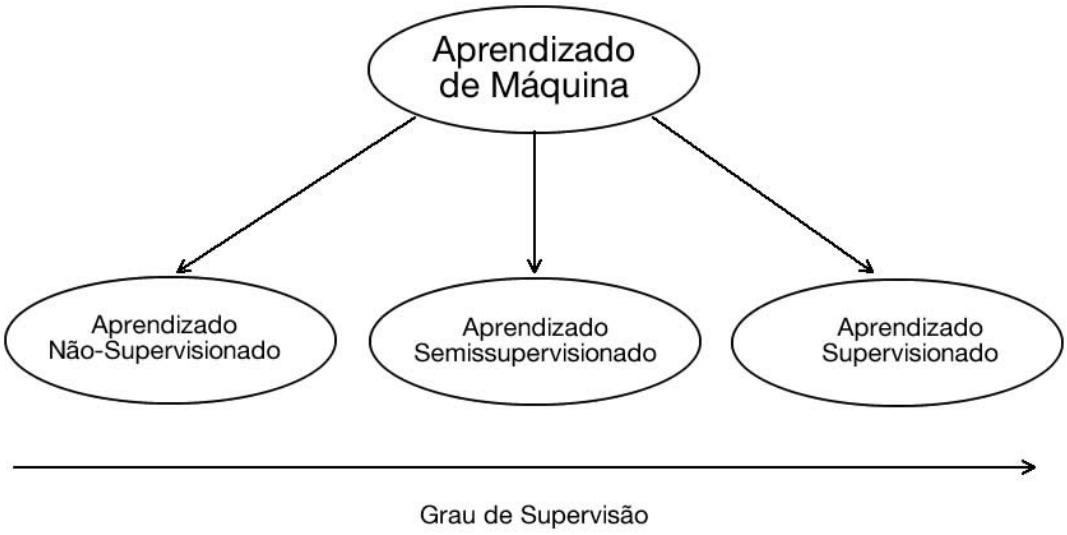
Em sua obra, Mitchell (1997) define formalmente o Aprendizado de Máquina como um programa de computador capaz de aprender pela experiência E, com respeito

a algum tipo de tarefa T e performance P, se sua performance P nas tarefas em T, na forma medida por P, melhoram com a experiência E. Como exemplo, um programa capaz de aprender a jogar xadrez pode assumir uma tarefa T, como "aprender a jogar xadrez"; uma performance P, como "percentual de jogos vencidos contra oponentes"; e uma experiência E, como "jogar jogos de treino contra si mesmo".

O processo de Aprendizado de Máquina aplica o conceito de indução utilizado pelo cérebro humano, obtendo conclusões genéricas a partir de um conjunto de exemplos. Segundo Monard e Baranauskas (2003), a inferência indutiva é um dos métodos mais utilizados para derivação de novo conhecimento, na qual um conceito é aprendido a partir de hipóteses geradas de exemplos conhecidos, podendo ou não preservar a verdade. Dessa forma, a indução deve ser utilizada com cuidado, uma vez que se os exemplos não forem bem escolhidos, as hipóteses podem ter pouco valor.

O aprendizado indutivo pode ser dividido em Aprendizado Supervisionado, Aprendizado Semissupervisionado, e Aprendizado Não Supervisionado, classificado de acordo com os métodos adotados no processo de generalização do conhecimento.

Figura 2 – Hierarquia do aprendizado indutivo conforme o grau de supervisão.



Fonte: Monard e Baranauskas (2003)

No Aprendizado Supervisionado são utilizadas técnicas para que algoritmos indutores façam inferências a partir de um conjunto de treinamento com exemplos rotulados. Neste processo, os exemplos utilizados possuem um resultado conhecido, além de ser definido por um conjunto de atributos e por um valor de classe ao qual o exemplo pertence. Na prática, um agente observa um exemplo de pares de entrada e de saída e aprende uma função que mapeie uma entrada para uma saída (RUSSELL; NORVIG, 2009).

Em seu trabalho, Russell e Norvig (2009) define formalmente uma tarefa de Aprendizado Supervisionado da seguinte forma:

> Dado um conjunto de treinamento com N pares de exemplo de entrada e saída $(x_1, y_1), (x_2, y_2), ...(x_n, y_n)$, onde cada y_j foi gerado por uma função desconhecida $y = f(x)$, descobrir uma função h que se aproxima da função verdadeira f.

Onde x e y podem assumir qualquer valor, independentemente de serem números ou não, e h representa uma hipótese. O aprendizado é realizado a partir da busca de uma hipótese que se aproxime dos valores de um conjunto de testes fornecido. É dito que uma hipótese generaliza bem se ela prediz corretamente os valores de novos exemplos.

De forma geral, o objetivo do Aprendizado Supervisionado é aprender uma regra que mapeie novos exemplos não rotulados em rótulos de classe. Quando a classe assume um valor discreto (como classe A ou B), o método de classificação é utilizado com o objetivo de classificar futuros exemplos de cada uma das classes preestabelecidas. Quando a classe assume valores contínuos (como por exemplo, o valor numérico de uma ação da bolsa), o método de regressão é utilizado com o objetivo de predizer o valor do atributo baseado nos exemplos apresentados.

No Aprendizado Não Supervisionado, os dados não são rotulados e o algoritmo não possui os dados de classe de cada exemplo. Neste caso, o objetivo é agrupar os exemplos por similaridades entre os valores de seus atributos em grupos ou em clusters. Sendo assim, o próprio algoritmo definirá a qual grupo um exemplo pertence, podendo unir exemplos que já tenham atributos previamente rotulados ou distinguir exemplos que tenham atributos dispersos. Entre os algoritmos mais famosos, estão o de k-médias e o algoritmo de agrupamento hierárquico (MONARD; BARANAUSKAS, 2003).

Já em Aprendizado Semissupervisionado, tanto o Aprendizado Supervisionado quanto o Aprendizado Não Supervisionado são utilizados para encontrar uma hipótese que classifique novos exemplos baseados em conjuntos de entrada rotulados e não-rotulados.

2.3.1 Redes Neurais Artificiais

A partir do modelo de aprendizado indutivo, as Redes Neurais Artificiais (RNA) foram criadas como um modelo computacional que simula o sistema nervoso dos seres vivos, adquirindo conhecimento através da experiência. As células de uma RNA

possuem uma interligação maciça, conhecidas como neurônios, a fim de garantir seu desempenho.

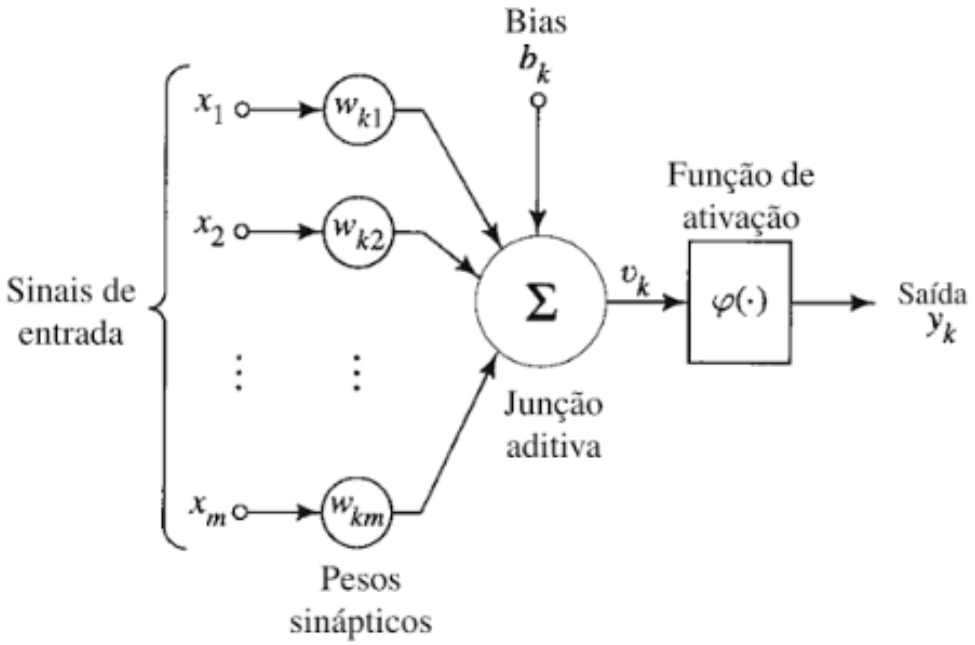
As forças de conexão entre os neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido. O processo de aprendizagem de uma RNA depende de um algoritmo de aprendizagem cuja função é modificar os pesos sinápticos da rede de maneira ordenada, com a finalidade de alcançar um objetivo (HAYKIN, 2009). Dessa forma, as RNA podem ser definidas como um mapeamento não-linear de um vetor de entrada para um vetor de saída através da ativação de neurônios, no qual as entradas são somadas com seus respectivos pesos para gerar uma saída (AGUIAR, 2010).

De acordo com Haykin (2009), um neurônio é a unidade fundamental de uma RNA e conta com três elementos básicos em seu modelo:

- Um conjunto de sinapses e elos de conexão, cada um contendo um peso ou uma força própria. Em essência, uma sinapse j de um neurônio k conta com um sinal x_i que é multiplicado por um peso w_{kj};
- Uma junção aditiva, com o objetivo de somar os sinais da entrada, ponderado através das sinapses do neurônio;
- Uma função de ativação, que introduz um componente de não-linearidade na rede neural ao restringir o limite do sinal da saída para um valor finito;

Por fim, o modelo de $bias$ (ou viés) pode ser aplicado externamente com o objetivo de aumentar ou de diminuir a entrada da função de ativação. O modelo provê um grau de liberdade a mais para a equação, que não é afetado pela entrada nessa expressão.

Figura 3 – Modelo não-linear de um neurônio artificial.



Fonte: Haykin (2009)

2.3.1.1 Aprendizado por Gradiente Descendente

Um dos algoritmos mais comuns para encontrar os pesos que minimizem o erro da rede é o algoritmo de Gradiente Descendente. Essencialmente, o Gradiente Descendente é um algoritmo de otimização baseado em uma função convexa que ajusta os parâmetros de forma iterativa para minimizar uma função específica ao mínimo local. De acordo com Lex Fridman, do MIT, um gradiente mede o quanto da saída de uma função muda se você muda um pouco a entrada (PERDANAPUTRA, 2017).

De forma geral, o gradiente pode ser considerado como a inclinação de uma função, quanto maior o gradiente mais íngreme a inclinação e mais rapidamente o modelo pode aprender. Entretanto, se a inclinação for 0, o modelo para de aprender (DONGES, 2018). A equação que descreve o Gradiente Descendente apresenta o próximo passo do ponto de mínimo local da função considerando o ponto do passo atual.

$$b = a - \gamma \Delta f(a) \tag{2.6}$$

onde:

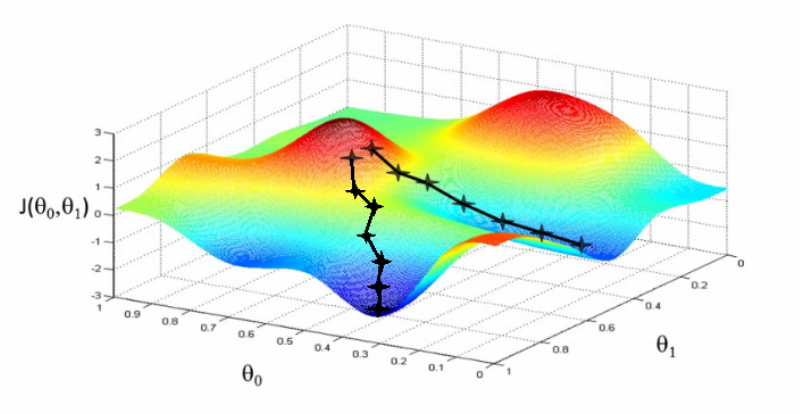
b é o valor da função no passo seguinte

a é o valor da função no passo atual

γ é a taxa de aprendizado, um valor positivo constante que determina o tamanho do passo a ser tomado na busca do gradiente descendente

f é a função a ser minimizada

Figura 4 – Representação de minimização de uma função bidimensional através de gradiente descendente.



Fonte: Perdanaputra (2017)

Em outras palavras, o termo $\gamma \Delta f(a)$ é subtraído de a para que o passo se mova de forma contrária ao gradiente, em direção ao mínimo local. Sendo assim, o valor mínimo de f no passo 0 é iniciado com um chute e, após n passos, é esperado que o resultado convirja para o mínimo local.

2.3.1.2 Modelo de Multicamadas

O modelo original de um perceptron de uma camada, concebido no trabalho de Rosenblatt (1958), não permite que uma rede neural faça a herança de recursos. Nesse caso, tal modelo é incapaz de realizar uma classificação não-linear, como a função XOR (momento em que a ativação da função ocorre apenas quando uma das entradas é ativada), conforme demonstrado no trabalho de Minsky e Papert (1987).

Para tal, o Modelo de Multicamadas (MLP) surgiu como uma classe de uma Rede Neural Artificial de alimentação avante composta por uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Cada camada oculta funciona como uma unidade de processamento da rede, capaz de realizar uma aproximação

de qualquer função contínua. Com o uso de camadas ocultas adicionais, o modelo de multicamadas é capaz de resolver problemas de classificação não-linear, incluindo a função XOR (GOODFELLOW; COURVILLE, 2016).

Entre as topologias mais básicas de RNA, estão as Redes de *Feedfoward*, que funcionam de maneira similar ao modelo clássico de perceptron, na qual os valores de entrada e de bias da camada de entrada são multiplicados por seus respectivos pesos. Entretanto, existe um peso para cada combinação de entrada e de unidade oculta. O produto do valor de entrada de uma camada de entrada e seus respectivos pesos são então passados como entrada para as camadas ocultas, que utilizam de uma função de ativação para gerar a saída como entrada de outra camada de unidades ocultas (GOODFELLOW; COURVILLE, 2016).

2.3.1.2.1 Treinamento por Backpropagation

O *Backpropagation* é considerado um dos algoritmos de aprendizagem mais importantes dentro do contexto de Redes de Aprendizagem Profunda (ACADEMY, 2019) e também um dos tipos de treinamento mais utilizados em Redes Neurais Artificiais (Sá, 2015). O objetivo do *Backpropagation* é otimizar a saída de uma função que modela os dados da rede através da seleção de pesos durante o treinamento. Isto é feito a partir do cálculo do erro na camada de saída, recalculando os pesos da última camada de neurônios e atualizando os pesos de trás para frente até atingir a camada de entrada.

Em outras palavras, o *Backpropagation* busca minimizar a função custo ao ajustar os pesos e os *bias* da rede, onde o nível dos ajustes é feito através do gradiente da função custo a partir dos parâmetros citados. Seja k o número de neurônios da camada l com suas i entradas, seus pesos sinápticos definidos por w_{ki}^{l} e a saída do respectivo neurônio definida por I_k, a Equação (2.7) define a atualização do $i-zima$ peso do neurônio k, referente a saída y_k^2 do mesmo neurônio (Sá, 2015).

$$w_{ki}^{3}(n+1) = w_{ki}^{3}(n) + \eta\delta_k^3(n)y_k^2(n) \tag{2.7}$$

onde:

η é a taxa de aprendizagem

$w_{ki}^{3}(n)$ é o peso anterior

$w_{ki}^{3}(n+1)$ é o peso atual

δ_k^3 é o gradiente definido pela Equação (2.8)

$$\delta_k^3(n) = (d(n) - y^3(n))\varphi'(I_k^3(n)) \tag{2.8}$$

onde:

$d(n)$ é o valor desejado para a saída da rede

$y^3(n)$ é o valor de saída da rede

A taxa de aprendizagem é introduzida como uma constante na Equação 2.7 com o propósito de forçar a atualização do peso de forma suave, evitando grandes passos e comportamentos caóticos (ACADEMY, 2019).

Em seguida, a atualização das camadas intermediárias é determinada pela Equação (2.9, utilizando o gradiente da camada de saída e da atividade pós sináptica da própria camada (Sá, 2015).

$$w_{ki}^2(n+1) = w_{ki}^2(n) + \eta\delta_k^2(n)y_k^1(n) \tag{2.9}$$

onde:

η é o valor de aprendizagem

$w_{ki}^2(n)$ é o peso anterior

$w_{ki}^2(n+1)$ é o peso atual

δ_k^2 é o gradiente definido pela Equação (2.10)

$$\delta_k^2(n) = (\sum_{j=1}^{n_{c3}} \delta_j^3(n)w_{ji}^3(n))\varphi'(I_k^2(n)) \tag{2.10}$$

onde:

δ_k^3 é o gradiente da camada seguinte

$w_{ki}^3(n)$ é o peso da camada seguinte

Por fim, a camada de entrada é atualizada conforme definido na Equação (2.11, utilizando o gradiente das camadas intermediárias e na atividade pós sináptica da própria camada (Sá, 2015).

$$w_{ki}^1(n+1) = w_{ki}^1(n) + \eta\delta_k^1(n)x_i(n) \tag{2.11}$$

onde:

η é o valor de aprendizagem

$w_{ki}^{1}(n)$ é o peso anterior

$w_{ki}^{1}(n+1)$ é o peso atual

δ_{k}^{1} é o gradiente definido pela Equação (2.12)

$$\delta_{k}^{1}(n) = (\sum_{j=1}^{n_{c3}} \delta_{j}^{2}(n) w_{ji}^{2}(n)) \varphi'(I_{k}^{2}(n)) \tag{2.12}$$

onde:

δ_{k}^{2} é o gradiente da camada seguinte

$w_{ki}^{2}(n)$ é o peso da camada seguinte

Para Academy (2019), existem diversos fatores que influenciam na decisão da quantidade de interações necessárias para que a função de perda atinja o valor mínimo global. São eles:

- Valor da taxa de aprendizado. Quanto mais alta a taxa de aprendizado, mais rápido será o aprendizado, mas com maior instabilidade;
- Quanto mais hiperparâmetros na rede (como quantidade de camadas e complexidade de funções não lineares) mais tempo levará para o valor convergir;
- Uso do método de otimização, uma vez que os otimizadores possuem diferentes performances quando comparados entre si;
- Inicialização aleatória da rede, podendo estar com os pesos ideais no início ou não;
- Qualidade do conjunto de treinamento;

2.3.1.2.2 Otimizadores

Além da regra delta, existem outros métodos de atualização de peso, chamados de otimizadores. Seu uso varia de acordo com a necessidade de suprir a demanda de desafios encarados pelo uso do Gradiente Descendente, tais como: escolha de uma taxa de aprendizagem apropriada; variação da taxa de aprendizagem durante o

treinamento e diferenças de resultado em dados esparsos e com diferentes frequências (RUDER, 2016).

A seguir, são listados os principais otimizadores e a descrição de seu algoritmo:

- Gradiente Descendente Estocástico (SGD): o algoritmo de SGD reduz a computação redundante em dados de larga escala ao realizar a atualização dos parâmetros uma única vez por interação, realizando atualizações mais frequentes com uma alta taxa de variância;
- Adagrad: o algoritmo de Adagrad adapta a taxa de aprendizagem dos parâmetros ao realizar pequenas atualizações para parâmetros associados com valores que aparecem com maior frequência, e atualizações maiores para parâmetros associados a valores que aparecem com menos frequência. Um dos principais benefícios no uso do Adagrad é que o algoritmo elimina a necessidade de regular manualmente a taxa de aprendizagem, onde muitas implementações usam o valor padrão de 0,01 sem realizar alterações futuras;
- RMSProp: o algoritmo de RMSProp foi desenvolvido com o propósito de lidar melhor com a situação de taxas de aprendizagem radicalmente descrescentes que ocorre com o uso do algoritmo de Adagrad, dividindo a taxa de aprendizado por uma média quadrática exponencialmente decrescente dos gradientes;
- Adam: tal qual como o RMSProp, o Adam também armazena uma média quadrática exponencialmente decrescente dos gradientes anteriores. Porém, diferente dos algoritmos citados, o Adam também armazena a média geral exponencialmente decrescente dos gradientes anteriores, estimando a média e a variância não centrada denominada da função.

Em seu trabalho, Ruder (2016) apresenta um comparativo entre os principais otimizadores utilizados. Segundo o autor, o RMSprop é uma extensão do método de Adagrad que lida com taxas de aprendizagem radicalmente decrescentes sem a necessidade do uso do parâmetro RMS durante a atualização dos valores, enquanto o Adam adiciona a correção de *bias* e *momentum* ao RMSProp. Entretanto, muitos trabalhos ainda utilizam o SGD com uma taxa de aprendizagem padrão, apresentando sucesso ao atingir o mínimo da função apesar da demora na execução se comparado aos demais otimizadores.

2.3.1.2.3 Redes Neurais Recorrentes

Redes Neurais Recorrentes (RNN) são um tipo de RNA que demonstram um comportamento dinâmico pelas suas conexões de neurônios terem um formato de um

grafo direcionado através de uma sequência temporal. Esse tipo de rede se destaca pelo uso comum de uma memória interna, capaz de processar sequências de dados e de realizar predições em relação aos dados armazenados.

De acordo com Fogel, Liu e Keller (2016), uma Rede Neural Recorrente é resultante da incorporação do elemento tempo através do uso de feedback, englobando uma ou mais camadas de neurônios ocultos de uma rede. Para Lex Fridman, do MIT, uma Rede Neural Recorrente deve ser utilizada sempre que existir uma sequência de dados e também sempre que a dinâmica temporal que conecta tais dados for mais importante que o espaço do conteúdo de cada quadro individual.

Diferentemente de Redes de *Feedfoward*, uma RNN leva em consideração a entrada atual juntamente com o que aprendeu de entradas recebidas anteriormente, podendo propagar a informação através de laços dentro de sua rede. Dessa forma, uma RNN pode mapear uma entrada para múltiplas saídas, enquanto Redes de *Feedfoward* mapeiam uma entrada para uma única saída.

Um dos exemplos mais comuns de uso de RNN é o processamento de linguagens naturais. Na abordagem de linguagens naturais é comum se trabalhar com sequências de entrada como palavras (sequências de letras), frases (sequências de palavras) e documentos. Por permitir representar entradas sequenciais de tamanho arbitrário em vetores de tamanho fixo, levando em consideração a estrutura das propriedades da entrada, diferentes tipos de RNN podem ser utilizados para se lidar com esse problema (GOLDBERG; HIRST, 2017).

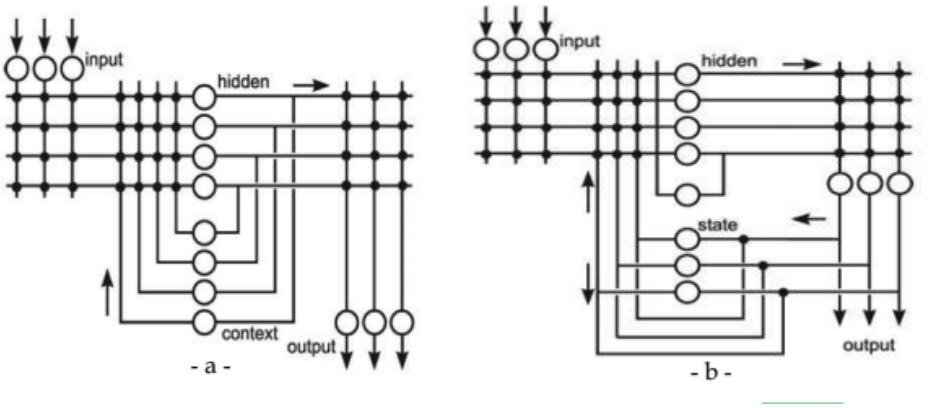
O modelo original da Rede de Hopfield (HOPFIELD, 1982), considerado um dos primeiros usos de redes com conexões recorrentes, conta com uma rede totalmente recorrente que, quando iniciada em um estado de ativação aleatório, estabelece a ativação ao longo do tempo em um estado que representa uma solução. Cada unidade da Rede de Hopfield é uma unidade de limiar binário, aceitando um valor ou outro dependendo se a entrada é maior ou não que um limite específico.

Já o modelo de Elman (1990) consiste em uma rede de três camadas com a adição de um conjunto das chamadas "unidades de contexto", conectadas em um modelo de 1 para 1 com as camadas ocultas da rede. Para que a recorrência seja feita, a saída da camada de contexto também é conectada com a entrada da camada oculta, provendo ao sistema uma memória de curto prazo. Dessa forma, as unidades ocultas não só observam a entrada atual, como também obtêm informações sobre seu próprio estado no último passo temporal através da camada de contexto.

Da mesma forma, o modelo de Jordan (1997) é similar ao modelo de Elman, se diferenciando na forma em que as unidades de contexto são alimentadas, recebendo a entrada a partir da camada de saída. Na obra de Jordan, as unidades de contexto são

chamadas de “camada de estado” e possuem conexões recorrentes para si mesmas.

Figura 5 – À esquerda, o modelo de rede de Elman; À direita, o modelo da rede de Jordan. Apenas os pesos nos canais de alimentação adiante podem ser modificados.



Fonte: Al-Mashhadany (2012)

Tanto o modelo da rede de Elman quanto o modelo da rede de Jordan são também chamados de "Redes Recorrentes Simples"(SRN), descritos pelas Equações (2.13) e (2.14), respectivamente.

$$\begin{aligned} h_t &= \delta_h(W_h x_t + U_h h_{t-1} + bh) \\ y_t &= \delta_y(W_y h_t + by) \end{aligned} \tag{2.13}$$

$$\begin{aligned} h_t &= \delta_h(W_h x_t + U_h y_{t-1} + bh) \\ y_t &= \delta_y(W_y h_t + by) \end{aligned} \tag{2.14}$$

onde:

x_t é o vetor de entrada

h_t é o vetor de unidades de ativação ocultas

y_t é o vetor de saída

W, U são as matrizes de parâmetros com os pesos a serem aprendidos

b é o vetor de parâmetro com bias

δ_h, δ_y são funções de ativação

De acordo com Bullinaria (2015), tais equações definem pares de matrizes não-lineares de um sistema dinâmico. De forma geral, um estado de um sistema dinâmico concentra informações do passado do sistema, necessárias para prover informações únicas para predições futuras, à parte dos efeitos de qualquer fator externo. No caso de Redes Neurais Recorrentes, o estado é definido por um conjunto de unidades de ativação ocultas h_t.

Sendo assim, Redes Neurais Recorrentes podem ter diferentes formas. Enquanto um dos tipos mais comuns consiste em um modelo de redes neurais de multicamadas com laços adicionados, aumentando a capacidade de memória, outras possuem estruturas uniformes, potencialmente com cada neurônio conectado com outros e podendo também ter funções de ativação estocásticas.

Para arquiteturas mais simples e para funções de ativação determinísticas, o aprendizado pode ser obtido através do uso de procedimentos similares ao gradiente descendente, levando ao uso do algoritmo de *Retropropagation* para redes de alimentação adiante (BULLINARIA, 2015).

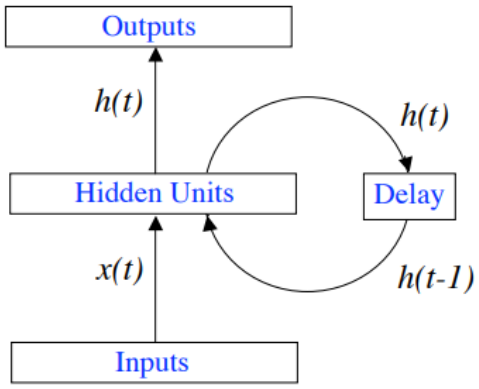
As seções a seguir abordam, em detalhes, os tipos mais importantes de Redes Neurais Recorrentes.

2.3.1.2.4 Redes Recorrentes Totalmente Conectadas

Redes Neurais Recorrentes Totalmente Conectadas são RNA que podem aprender sequências temporais através de pacotes de dados ou de forma on-line, consistindo de camadas de entrada e de saída de unidades lineares e não-lineares, respectivamente. Cada unidade da camada de entrada é totalmente conectada com cada camada da unidade de saída através de pesos ajustáveis, na qual cada unidade possui funções de ativação de valores reais variantes conforme o tempo. A camada de saída possui conhecimento de suas ativações passadas, enviando feedbacks para a ativação das unidades da camada de entrada (TRUPTI, 2018).

De acordo com Bullinaria (2015), a forma mais simples de uma Rede Recorrente Totalmente Conectada é uma rede de multicamadas com um conjunto de unidades de ativação alimentando a rede de forma retroativa através das entradas. Como a escala do tempo é variada e pode mudar de acordo com a quantidade de passos para a execução do processo, uma unidade de atraso precisa ser inclusa para manter as ativações até que elas sejam processadas na próxima etapa.

Figura 6 – Modelo de uma Rede Neural Recorrente Totalmente Conectada, onde h_t representa um novo estado, h_{t-1} representa o estado anterior, e x_t representa a entrada atual.



Fonte: Bullinaria (2015)

2.3.1.2.5 Redes de Memória de Longo Prazo

A arquitetura de Redes de Memória de Longo Prazo (LTSM) foi introduzida em 1997 (HOCHREITER; SCHMIDHUBER, 1997) e é um dos tipos de rede mais utilizadas por conta de sua performance superior ao modelar dados de curto e de longo prazo de forma acurada. Tal rede é capaz de superar intervalos de tempo superiores a 1000 intervalos, mesmo no caso de sequências de ruídos incompressíveis na entrada, sem perda de capacidade de atraso de curto tempo (MIKAMI, 2016).

Redes de Memória de Longo Prazo solucionam parcialmente o problema do desaparecimento do gradiente em RNA treinadas com *Retropropagation*, que ocorre quando o gradiente tende a zero ou ao infinito devido a erros de precisão e arredondamento durante seu processo de computação. Isso é feito através da não-imposição de viés através de observações recentes, mantendo o fluxo de erro constante através de estados internos de uma unidade especial conhecida como “célula de memória” (TRUPTI, 2018).

Enquanto existem diferentes tipos de arquiteturas de Redes de Memória de Longo Prazo, as mais comuns incluem o uso de uma célula de memória e três reguladores denominados como portões de entrada, de saída e de esquecimento. O portão de esquecimento é a primeira das três etapas que compõem o estado de uma célula, definido através da Equação (2.15). Sua função é de decidir quais informações são

úteis e quais não são através do uso da função sigmoide.

$$f_t = \delta(W_f * [h_{t-1}, x_t] + bf) \tag{2.15}$$

onde:

δ é uma função de ativação

W é a matriz de parâmetros com os pesos a serem aprendidos

b é o vetor de parâmetro com bias

f_t é o vetor de ativação do portão de esquecimento

x_t é o vetor de entrada da unidade LTSM

h_{t-1} é o vetor da camada oculta do passo anterior

O portão de entrada é composto por duas partes, determinadas pela Equação (2.16). Primeiramente é determinado quais informações serão guardadas na célula de memória. Em seguida, o uso da função tangente hiperbólica permite criar um vetor de novos valores candidatos a serem adicionados ao estado.

$$\begin{aligned} i_t &= \delta(W_i * [h_{t-1}, x_t] + bi) \\ {}_t &= tahn(W_C * [h_{t-1}, x_t] + bC) \end{aligned} \tag{2.16}$$

onde:

δ é uma função de ativação

W é a matriz de parâmetros com os pesos a serem aprendidos

b é o vetor de parâmetro com bias

i_t é o vetor de ativação do portão de entrada

x_t é o vetor de entrada da unidade LTSM

h_{t-1} é o vetor da camada oculta do passo anterior

C_t é o vetor do estado da célula

$tahn$ é função da tangente hiperbólica como função de ativação

O passo seguinte consiste na atualização da célula antiga, C_{t-1} em uma nova célula C_t. Isso é feito ao se multiplicar o antigo estado por f_t para esquecer as informações filtradas pelos passos anteriores e ao somar $i_t * {}_t$, que são os novos valores candidatos escalados por valores de atualização de cada estado.

$$C_t = f_t * C_{t-1} + i_t * {}_t \tag{2.17}$$

onde:

f_t é o vetor de ativação do portão de esquecimento

i_t é o vetor de ativação do portão de entrada

C_t é o vetor do estado da célula

C_{t-1} é o vetor do estado da célula anterior

Por fim, o portão de saída tem a função de determinar o que sairá da célula de memória ao aplicar a função sigmoide nos estados ocultos anteriores e na entrada atual, multiplicando, em seguida, pela função tangente hiperbólica aplicada a nova célula de memória, conforme definido pela Equação (2.18).

$$\begin{aligned} o_t &= \delta(W_o * [h_{t-1}, x_t] + bo) \\ h_t &= o_t * tahn(Ct) \end{aligned} \tag{2.18}$$

onde:

δ é uma função de ativação

W é a matriz de parâmetros com os pesos a serem aprendidos

b é o vetor de parâmetro com bias

x_t é o vetor de entrada da unidade LTSM

h_t é o vetor da camada oculta do passo atual

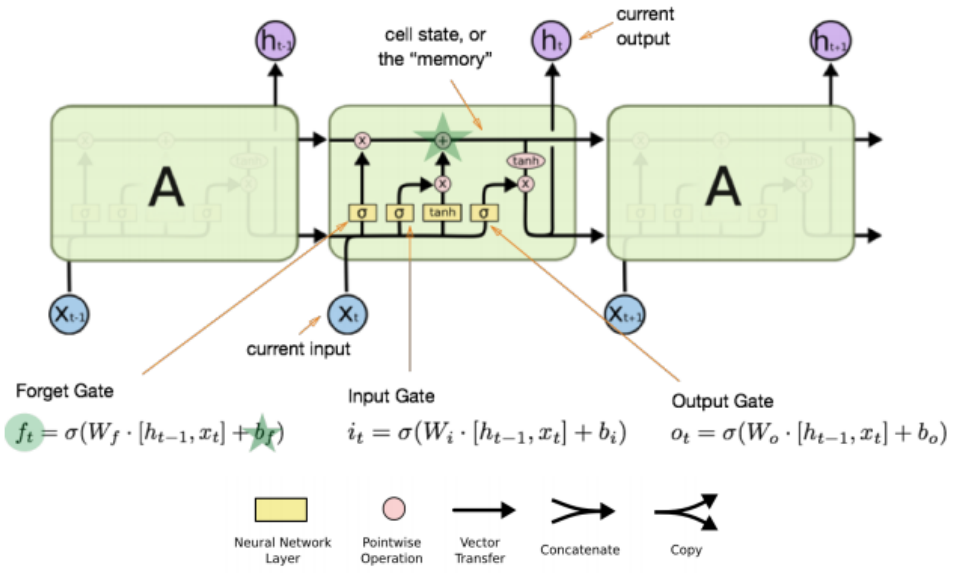
h_{t-1} é o vetor da camada oculta do passo anterior

C_t é o vetor do estado da célula

$tahn$ é função da tangente hiperbólica como função de ativação

o_t é o vetor de ativação do portão de saída

Figura 7 – Estado oculto de uma Rede de Memória de Longo Prazo.



Fonte: Bullinaria (2015)

2.3.1.2.6 Redes de Unidades Recorrentes de Bloqueio

Originalmente proposta por Cho et al. (2014), a arquitetura de Unidades Recorrentes de Bloqueio é semelhante à arquitetura de Memória de Longo Prazo por utilizar portões de esquecimento e de entrada para filtrar os dados previamente. Porém, nessa arquitetura, tais portões são unificados em um único portão para determinar o que uma unidade oculta deve aprender ou esquecer de forma adaptativa.

As Redes de Unidades Recorrentes de Bloqueio utilizam de dois portões, conforme definido pela Equação (2.19). O primeiro é denominado “portão de atualização” e controla o quanto o conteúdo de uma célula corrente deve ser atualizado em relação ao novo estado candidato. O segundo, denominado “portão de reinício”, se fechado, pode efetivamente reiniciar a memória da célula e fazer a unidade agir como se a próxima entrada processada fosse a primeira da sequência.

$$\begin{aligned} \text{portão de atualização} : z_t &= \delta(W_z x[t] + U_z h_{t-1} + bz) \\ \text{portão de reinicio} : r_t &= \delta(W_r x[t] + U_r h_{t-1} + br) \\ \text{portão de saída} : h_t &= (1 - z_t) \cdot h[t-1] + z[t] \cdot \delta(W_h x[t] + U_h(r_t \cdot h_{t-1}) + bh) \end{aligned} \tag{2.19}$$

onde:

δ é uma função de ativação

W, U são matrizes de parâmetros com os pesos a serem aprendidos

b é o vetor de parâmetro com bias

x_t é o vetor de entrada

h_t é o vetor de saída

z_t é o vetor do portão de atualização

r_t é o vetor do portão de reinício

2.3.1.3 Funções de Ativação

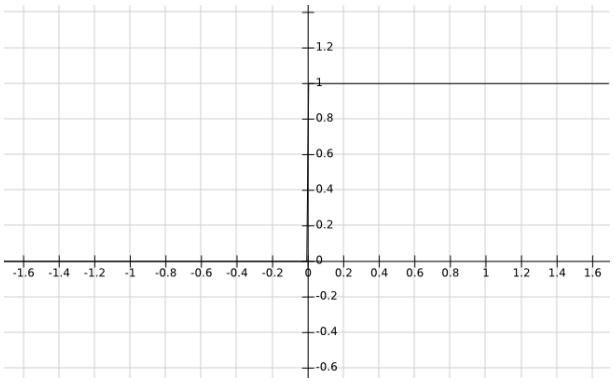
Considerando o aspecto binário do processo de ativação de redes neurais biológicas, as Redes Neurais Artificiais dependem de funções de ativação para acionar um neurônio. Tais funções contam com valores resultantes dentro de um intervalo específico, podendo ser de 0 a 1, -1 a 1 ou outros intervalos, variando entre os demais tipos de funções.

As funções de ativação trazem o aspecto de não-linearidade para o processo de aprendizagem de uma rede neural. Quando uma função de ativação é inexistente, pesos e bias funcionam na forma de uma transformação linear, limitando a rede para soluções mais complexas. Essencialmente, uma rede neural sem função de ativação é apenas um modelo de regressão linear.

De acordo com Haykin (2009), entre as funções de ativação mais básicas estão a Função de Limiar, a Função Linear, e a Função Sigmoide. A Função de Limiar define uma regra simples de ativação ou não do neurônio, considerando se o valor da saída do neurônio anterior está dentro de um limite especificado a partir de valores binários, estabelecida pela Equação (2.20) e demonstrada através do gráfico da imagem 8.

$$f(x) = \begin{cases} 1 & \text{se } x \geq 0 \\ 0 & \text{se } x < 0 \end{cases} \tag{2.20}$$

Figura 8 – Representação gráfica da Equação (2.20).



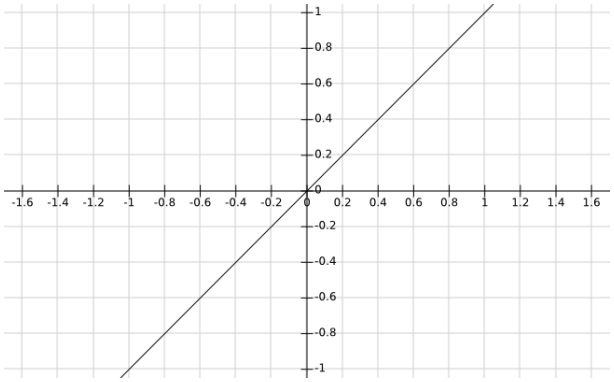
Fonte: O autor

Enquanto a Função de Limiar é a melhor escolha para elaboração de classificadores binários, pelos quais se deve classificar um dado como pertencente ou não a uma única classe, tal função não é capaz de classificar dados em diferentes classes. Ao ser aplicada no processo de back propagation, o gradiente da função resulta no valor 0, impossibilitando que a melhora do modelo seja realizada.

Já a Função Linear é uma função na qual a ativação é proporcional à entrada, definida pela Equação (2.21). Seu uso é comum em camadas de saída de Redes de Arquitetura Multinível ou de Redes Neurais de Regressão. Porém, por incluir características de linearidade para a rede, a Função Linear não provê soluções complexas, especialmente ao tratar de múltiplas classes. Nesse caso, a derivada de uma Função Linear é um valor constante, resultando em um gradiente igual toda vez que o processo de back propagation for realizado.

$$f(x) = ax \tag{2.21}$$

Figura 9 – Representação gráfica da Equação (2.21).

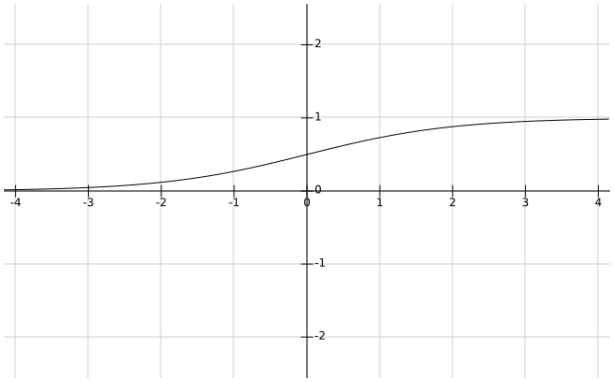


Fonte: O autor

A Função Sigmoide possui o formato de um “S” e é geralmente utilizada em modelos nos quais é necessário predizer a probabilidade da saída em valores definidos entre 0 e 1. Sendo uma função contínua e diferenciável, possui características não-lineares, permitindo seu uso em modelos multiníveis. Por possuir valores altos entre seus extremos, mudanças nos valores de x nesta região farão com que o resultado mude significantemente, criando uma tendência de trazer os valores de y para qualquer extremidade da curva.

$$f(x) = \frac{1}{1 + e^{(} - x)} \tag{2.22}$$

Figura 10 – Representação gráfica da Equação (2.22).

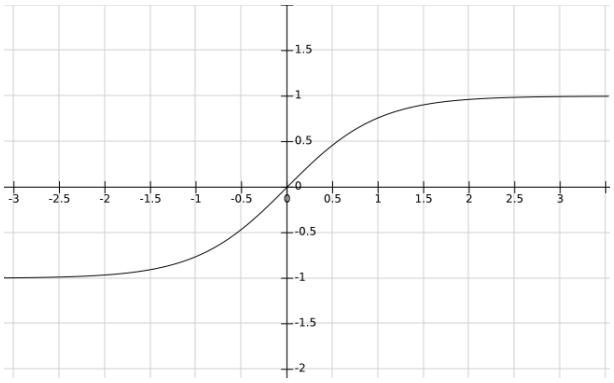


Fonte: O autor

A Função da Tangente Hiperbólica é comumente utilizada em Redes Neurais de Multicamadas, normalizando a entrada entre valores de -1 a 1. Seu uso é alternativo ao da Função Sigmoide por permitir que entradas fortemente negativas resultem em saídas de valores negativos (ao contrário da Função Sigmoide, que tende a ter saídas de valor 0 quando entradas fortemente negativas são aplicadas).

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.23}$$

Figura 11 – Representação gráfica da Equação (2.23).

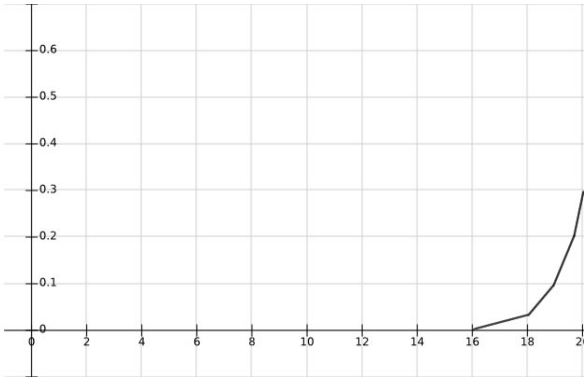


Fonte: O autor

A Função de Softmax é uma função que recebe um vetor de K números reais e o normaliza em uma distribuição probabilística de K probabilidades, resultando em um vetor de componentes variando entre 0 e 1. Sua aplicação em redes neurais é geralmente usada como a camada final de uma rede neural de predição de classes.

$$f(x_i) = \frac{e_i^x}{\sum_{j=1}^{K} e_i^x} \tag{2.24}$$

Figura 12 – Representação gráfica da Equação (2.24).



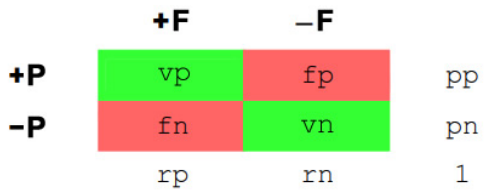
Fonte: O autor

2.3.1.4 Métricas de Avaliação em Problemas de Classificação

A qualidade das predições obtidas pelos algoritmos de aprendizado de máquina é medida através da avaliação dos resultados obtidos pelo modelo definido. Para tal, é necessário o uso de métricas de avaliação, que variam de acordo com o domínio do problema enfrentado. Entre as métricas mais comuns para aprendizado de máquina está a *Accuracy*, a *Precision*, a *Recall* e o *F1-Score*, comumente utilizados em problemas de classificação (POWERS; AILAB, 2011).

A tabela de contingência é um método utilizado para sumarizar as predições de um classificador, representado através de contadores do número de vezes que um rótulo é associado a uma classe ou representado através de termos. Verdadeiros Positivos e Falsos Positivos dizem respeito ao número de Verdadeiros Previstos que foram corretos e incorretos, respectivamente, enquanto Verdadeiros Negativos e Falsos Negativos dizem respeito ao número de Negativos Previstos que foram corretos e incorretos, respectivamente (POWERS; AILAB, 2011).

Figura 13 – Notação da tabela de contingência.



Fonte: Powers e Ailab (2011). Adaptado pelo autor

A partir dos valores obtidos pela tabela de contingência, é possível realizar o cálculo dos outros tipos de métrica, conforme listados a seguir:

- *Recall*: determina o resultado de itens identificados corretamente como positivos a partir do total de positivos verdadeiros.

$$Recall = \frac{\text{PositivosVerdadeiros}}{\text{PositivosVerdadeiros} + \text{NegativosFalsos}} \tag{2.25}$$

- Especificidade ou Taxa de Negativos Verdadeiros: determina o número de itens identificados corretamente como negativos a partir do total de negativos verdadeiros.

$$\text{Especificidade} = \frac{\text{NegativosVerdadeiros}}{\text{NegativosVerdadeiros} + \text{PositivosFalsos}} \tag{2.26}$$

- *Precision*: determina o resultado de itens identificados corretamente como positivos a partir do total de positivos.

$$Precision = \frac{\text{PositivosVerdadeiros}}{\text{PositivosVerdadeiros} + \text{PositivosFalsos}} \tag{2.27}$$

- Taxa de Falsos Positivos ou Erro Tipo 1: determina o número de itens classificados errôneamente como positivos a partir do total de negativos verdadeiros.

$$\text{Taxa de Falsos Positivos} = \frac{\text{PositivosFalsos}}{\text{PositivosFalsos} + \text{NegativosVerdadeiros}} \tag{2.28}$$

- Taxa de Falsos Negativos ou Erro Tipo 2: determina o número de itens classificados errôneamente como negativos a partir do total de positivos verdadeiros.

$$\text{Taxa de Falsos Negativos} = \frac{\text{NegativosFalsos}}{\text{NegativosFalsos} + \text{PositivosVerdadeiros}} \tag{2.29}$$

- *F1-Score*: determina a média harmônica da *Precision* e da *Recall*.

$$\text{F1 Score} = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.30}$$

- *Accuracy*: Porcentagem do total de itens classificados corretamente.

$$Accuracy = \frac{\text{PositivosVerdadeiros} + \text{NegativosVerdadeiros}}{\text{Negativos} + \text{Positivos}} \tag{2.31}$$

2.3.1.5 Transformação de dados categóricos em valores numéricos

Apesar da maioria dos algoritmos de aprendizado de máquina conseguirem trabalhar diretamente com dados categóricos, alguns ainda necessitam que a entrada seja transformada em valores numéricos. Para tal, existem técnicas de representação de dados em forma de valores inteiros, utilizados em aprendizado de máquina e em análise de dados em larga escala, que permitem tal conversão.

O método mais simples de conversão de dados categóricos em inteiros é o *Integer Encoding*, que associa uma palavra diretamente a um valor numérico. Por exemplo, um conjunto das cores vermelho, azul e verde poderia ser representado pelo vetor $[1, 2, 3]$, onde $\text{vermelho} = 1$, $\text{azul} = 2$ e $\text{verde} = 3$.

Entretanto, o uso de Integer Encoding para variáveis categóricas nas quais não existe uma relação ordinal, permitindo que o modelo assuma uma ordem natural entre as categorias, pode resultar em uma performance fraca ou em resultados inesperados. Nesse caso, um método comum para transformação de dados é o *One-Hot Encoding*.

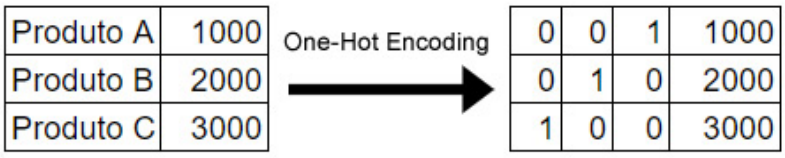
O *One-Hot Encoding* consiste na transformação de valores categóricos em vetores com valores binários. Seja $\{\hat{y}^1, \hat{y}^2, \dots \hat{y}^\mu\}$ um conjunto de padrões de saída

tamanho p, onde cada elemento é definido pela Equação (2.32). Cada vetor deverá conter um total de $p-1$ zeros, enquanto a posição referente à categoria transformada deverá ser marcada com o número 1 (FU et al., 2018).

$$\hat{y}^{\mu}\} = \begin{cases} 1 & \text{se } i = \mu \\ 0 & \text{caso contrário} \end{cases} \tag{2.32}$$

Um exemplo seria a transformação de dados de uma coleção de dados que relacionam produtos a um preço. Após o uso de One-Hot Encoding, os dados categóricos são transformados em vetores com valores binários, conforme visto na Figura (14).

Figura 14 – Transformação de dados por *One-Hot Encoding*.



Fonte: O autor

Entretanto, utilizar o algoritmo de *One-Hot Encoding* em um grande número de palavras pode ser custoso para o modelo em termos de poder computacional. Nesse caso, as técnicas de *Word Embedding*, que transformam palavras individuais em vetores de valores reais dentro de um espaço vetorial pré-definido, garantem uma performance melhor que o *One-Hot Encoding*. Entre as principais técnicas de *Word Embedding*, estão o *Word2Vec* e o *GloVe*.

2.4 Histórico de Navegação

O histórico de navegação de um site consiste nas informações sequenciais das páginas visitadas por um usuário desse determinado site Fu, Budzik e Hammond (2000). Tais informações são relevantes para identificar o caminho que um usuário levou pra chegar em uma determinada página, podendo definir um perfil de interesse baseado nas páginas acessadas anteriormente. Um sistema de comércio eletrônico, por exemplo, pode utilizar de tais informações para checar quais produtos um usuário acessou antes de concluir a sua compra e assim definir um grupo de produtos relacionados que possam satisfazer outros usuários.

Uma das principais ferramentas utilizadas para checar o histórico de navegação de um usuário em um site é o Google Analytics, uma ferramenta que permite agregar estatísticas de visitação em uma página a fim de otimizar campanhas de marketing,

além de armazenar dados como sistema operacional utilizado, localização geográfica do visitante, navegador e resolução da tela utilizada. Para utilizá-lo, basta inserir um código fornecido pelo próprio Google Analytics ao site no qual se deseja que seja realizado o rastreamento, armazenando e enviando as informações de acesso da página diretamente para a ferramenta.

Outra forma de armazenar o histórico do usuário é gravar suas informações nos *Cookies* do navegador. *Cookies* nada mais são do que arquivos enviados ao navegador pelo site acessado, são armazenados no disco rígido do computador do usuário e contêm informações que podem ser utilizadas posteriormente por aquele site. Dados como usuário e senha, de formulários, de e-mail ou de redes sociais, por exemplo, podem ser armazenados nos Cookies dos respectivos sites. Tais sites também determinam o tempo de expiração de um *Cookie*, que pode variar entre dias a anos.

Uma das vantagens do uso do histórico de navegação é a possibilidade de armazenamento de informações de um usuário sem que este tenha feito algum tipo de registro no site. Um e-commerce pode, por exemplo, armazenar o histórico de visitas de um usuário para a recomendação de novos produtos sem que o usuário esteja logado. Da mesma forma, o sistema também pode armazenar tais informações até que o usuário efetue o login, registrando suas preferências na base de dados.

2.5 Estado da Arte

O objetivo desta seção é identificar os principais trabalhos que se relacionam com a temática desta pesquisa a partir de bases indexadas. Para tal, foi desenvolvido um protocolo, detalhado no Apêndice A deste trabalho, contendo o passo a passo para reprodução dos mesmos resultados obtidos.

Em seguida, foi realizada uma leitura preliminar para filtrar os resultados de acordo com os objetivos desta pesquisa e os critérios de inclusão e de exclusão foram estabelecidos, identificando assim os trabalhos que possuem maior afinidade com o estudo em questão, conforme apresentado no Apêndice B.

2.5.1 Uso de Redes Neurais para Sistemas de Recomendação

O uso de Redes Neurais em Sistemas de Recomendação se dá pela necessidade de um sistema mais genérico como alterativa para empresas de pequeno porte, uma vez que Sistemas de Recomendação customizados se mostram como uma solução de alto custo (KOROTAEV; LYADOVA, 2018). Em seu trabalho, Korotaev e Lyadova (2018) comparam as decisões existentes para o desenvolvimento de um Sistema de Recomendação através de cinco critérios de avaliação. Cada decisão é listada pelos

autores da seguinte forma:

- Serviços com o módulo de recomendações para um certo domínio de dados;
- Projetos de pesquisa direcionados para a criação de uma plataforma universal capaz de funcionar com diferentes domínios de dados;
- Diferentes bibliotecas usadas por desenvolvedores para a criação de seus próprios sistemas de recomendação;
- O sistema de recomendação universal, no caso o Prediction IO, da Apache, como um produto de software completo;

Na Tabela (4), cada decisão é avaliada em valores de 0, 1 e 2, em que 0 indica que a decisão não realiza a função esperada; 1 indica que a decisão realiza parcialmente a função esperada; e 2 indica que a decisão realiza completamente a função esperada. O resultado aponta que, apesar do Prediction IO ter a melhor avaliação, ele não possui módulos para armazenamento e para análise de dados necessários para a criação de Sistemas de Recomendação.

Tabela 4 – Comparação das decisões existentes para o desenvolvimento de Sistemas de Recomendação.

Critério de Avaliação	**SR como parte dos sistemas completos**	**Protótipo de Pesquisa**	**Bibliotecas**	**Prediction IO**
Universialidade	0	2	1	2
Habilidade de integração	0	0	0	1
Coleta de dados	2	0	0	0
Análise de dados	1	0	0	0
Acessibilidade	0	0	2	2

Fonte: (KOROTAEV; LYADOVA, 2018).

Sistemas de Recomendação exigem a necessidade de armazenamento de informação para permitir uma recomendação com melhor acurácia, o que é uma limitação de Redes Neurais convencionais. Entretanto, Redes Neurais Recorrentes são flexíveis e permitem o processamento de dados sequenciais, além de não necessitarem do recálculo do modelo quando é preciso obter novas recomendações (KOROTAEV; LYADOVA, 2018).

2.5.2 Sessão de Usuário como insumo de Redes Neurais Recorrentes

Os dados de uma sessão consistem em uma sequência de ações do usuário, sendo que estas sessões podem variar de tamanho e tais ações têm chances de serem dependentes. Esse formato de dado se adequa às Redes Neurais Recorrentes, pelo qual o modelo tem a capacidade de identificar as dependências entre os itens através do uso dos recursos de memória de uma rede (RUOCCO; SKREDE; LANGSETH, 2017). Tal solução é definida por Zhou et al. (2018) como um Session-Based Prediction (SBP), podendo ser estabelecida em qualquer sistema que contenha a sequência de comportamento do usuário disponível.

Os trabalhos analisados nesta revisão apresentam o uso de sessões como insumo de Redes Neurais Recorrentes em diferentes domínios. O trabalho de Heinz, Bracher e Vollgraf (2017) identifica o histórico de compras de um usuário dentro de um sistema de e-commerce de roupas, enquanto Chu et al. (2017) utilizam o histórico de filmes assistidos por um usuário para recomendar um próximo filme e, no caso de Kong, Li e Lv (2018), os autores utilizam dados de trânsito em horários específicos para predizer condições do trânsito.

Enquanto cada domínio possui propriedades particulares para cada item, diferentes tipos de dados podem ser adicionados no modelo para um possível aumento de precisão da recomendação. Zhou et al. (2018) sugerem a inclusão do tempo de permanência em uma página como entrada para a rede neural, de tal forma que a base de dados seja uma tupla contendo um identificador do item e um valor numérico que represente sua permanência, considerando que esse dado é essencial para determinar o engajamento do usuário e para reduzir o potencial do modelo de se adaptar demais ao conjunto de dados de treinamento. Além desses dados, Ruocco, Skrede e Langseth (2017) também sugerem o uso de informações como categoria do item, da imagem, da descrição textual e da geolocalização para aumento da precisão da predição.

Entretanto, a forma como os dados de uma sessão influencia na predição de novos itens varia de acordo com seu domínio. Por exemplo, em um portal de publicação de notícias, se um usuário consumiu notícias de uma determinada categoria em seu histórico, possivelmente ele consumirá mais notícias daquela mesma categoria. Enquanto em um e-commerce, se um usuário já adicionou itens de uma mesma

categoria em seu carrinho, ele provavelmente não irá querer adicionar novos itens da mesma categoria (RUOCCO; SKREDE; LANGSETH, 2017).

Considerando o formato categórico dos valores de insumo para sistemas de predição baseados em sessão, existe a necessidade de transformá-los em valores numéricos para uso em algoritmos. A maioria dos trabalhos revisados que especificam a forma como transformaram seus dados, utilizam a técnica de *One-Hot Vector* para obter um vetor de inteiros para cada valor de entrada. No caso de Fu et al. (2018), ainda é proposto pelo autores o uso de *Word Embedding* para evitar vetores de tamanhos longos.

Já Ruocco, Skrede e Langseth (2017) explicam que a técnica de *One-Hot Vector*, apesar de ser a mais convencional, é custosa para o processo de incorporação dos dados para um vetor. No lugar, foi utilizada a incorporação dos itens diretamente, considerando que ambas as técnicas são similares matematicamente, exceto pelo fato de que o uso de *One-Hot Vector*, quando o número de itens é muito grande, requer mais memória para a criação de pequenos conjuntos de vetores.

2.5.3 Arquiteturas de Redes Neurais Recorrentes

As Redes Neurais Recorrentes são conhecidas por superar as limitações de redes neurais tradicionais ao se tratar de dados sequenciais, memorizando as informações da sequência anterior e aplicando-as no cálculo da saída da sequência atual. Entretanto, Redes Neurais Recorrentes não podem usar efetivamente qualquer tamanho de dado histórico devido ao problema da explosão do gradiente e do desaparecimento do gradiente (KONG; LI; LV, 2018).

Para tratar tal problema, os demais trabalhos identificados nesta revisão utilizam diferentes modelos arquiteturais de Redes Neurais Recorrentes. Entre os modelos de rede de maior frequência, estão as Redes de Memória de Longo Prazo e as Redes de Unidades Recorrentes de Bloqueio.

O trabalho de Anil et al. (2018) avalia a melhor arquitetura para transformar análises em avaliações para um sistema de recomendação de produtos através de testes em três modelos de Redes Neurais Recorrentes: Redes de Memória de Longo Prazo, Redes de Unidades Recorrentes de Bloqueio e uma Rede Híbrida que unifica ambas para identificar os benefícios de tal arquitetura.

Ainda de acordo com Anil et al. (2018), a análise de sentimento é vital para sistemas e-commerce, e sentimentos podem ser expressados através de avaliações que, por usa vez, são utilizados por sistemas de recomendação para identificar relações entre usuários e produtos. Modelos de aprendizado profundo permitem realizar essa transformação das palavras em avaliações e as arquiteturas de Redes de Memória

de Longo Prazo e de Redes de Unidades Recorrentes de Bloqueio se mostram muito eficientes nesse processo.

O método proposto utiliza um sistema de recomendação por filtragem colaborativa baseada em memória, na qual a entrada dada ao sistema de recomendação é a avaliação dada por usuários sobre produtos da base de avaliações da *Amazon Fine Food*. Tais dados foram regularizados e utilizados nas três arquiteturas de Redes Neurais Recorrentes mencionadas, avaliando os resultados em três métricas: acurácia; entropia cruzada categórica; e erro quadrático médio.

Tal experimento mostrou que as três redes neurais recorrentes são comparáveis em termos de acurácia, porém a solução híbrida, que unifica uma rede de memória de longo prazo e uma rede de unidades recorrentes de bloqueio, apresenta maior acurácia com a menor perda, enquanto o tempo para treiná-la é menor do que as demais arquiteturas testadas. Para Anil et al. (2018), ao utilizar a solução híbrida, as avaliações podem ser mais intuitivas.

2.5.3.1 Redes de Memória de Longo Prazo

A arquitetura de Redes de Memória de Longo Prazo aparece em grande parte das propostas obtidas durante a revisão deste trabalho. Segundo Kong, Li e Lv (2018), esse modelo se destaca pela possibilidade de realizar predições com longos intervalos de tempo, se sobressaindo sobre às demais Redes Neurais Recorrentes em relação à dependência de longas sequências de dados.

Em sua pesquisa, Yuan, Yang e Bao (2017) propõem um modelo que utiliza Redes Neurais Recorrentes em paralelo com Redes de Crença Profunda a fim de melhorar a condição de *Cold-Start* em sistemas de recomendação que utilizem comentários longos, uma vez que tais comentários demandam um poder de processamento mais alto.

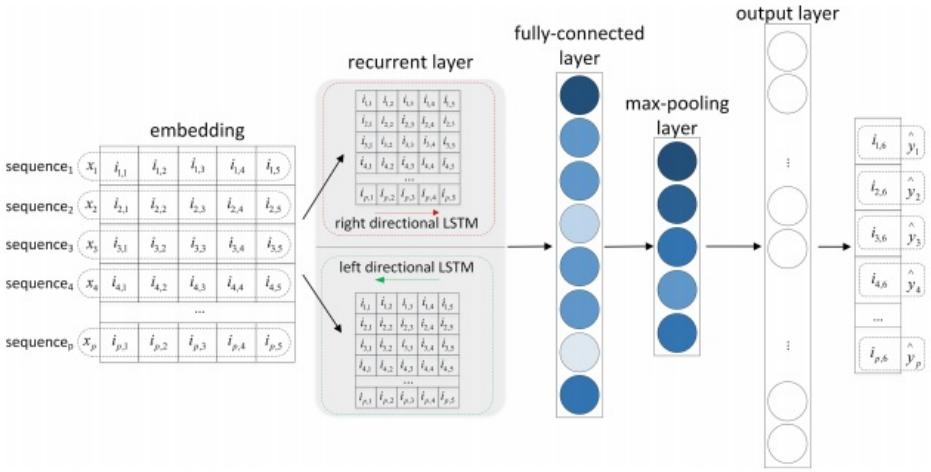
Enquanto o papel do uso de Redes de Crença Profunda é de extrair os principais fatores de comentários de um item, pelo qual cada comentário é transformado em um vetor de palavras, as Redes Neurais Recorrentes são utilizadas para extrair os fatores a partir do contexto semântico dos comentários, considerando a sequência das palavras através do vetor. Isso é feito através do método *Word2Vec*, gerando vetores capazes de representar não só informações semânticas, mas também sintáticas.

O modelo de Yuan, Yang e Bao (2017) utiliza Redes de Memória de Longo Prazo para, em seguida, obter a média das sequências da entrada através do método de *Mean-Pooling*. Por fim, uma camada totalmente conectada é utilizada para obter os fatores latentes dos itens. Através da métrica do Erro Médio Quadrático, os autores compararam os resultados de seu modelo em relação a outros três modelos, que

incluem a média global das avaliações, o método de Fatorização de Matrizes e o uso de Redes de Crença Profunda em paralelo. De acordo com Yuan, Yang e Bao (2017), o modelo proposto de Redes Neurais Recorrentes em paralelo com Redes de Crença Profunda se mostra superior através das demais bases de dados utilizadas.

Já Fu et al. (2018) propõem o uso de uma Rede de Memória de Longo Prazo Bidirecional que, além de capturar os recursos sequenciais dos dados, ainda escala linearmente de acordo com o número de objetivos (tanto para usuários quanto para itens). A proposta consiste no uso de cinco camadas que compõem o sistema de recomendação: incorporação, estrutura recorrente, camada totalmente conectada, camada de pooling e camadas de saída.

Figura 15 – Arquitetura do sistema de recomendação por Redes Neurais de Memória de Longo Prazo.



Fonte: Fu et al. (2018)

A camada de incorporação utiliza as últimas sequências de itens visitados pelo usuário como recurso e o último item como um rótulo para construir um modelo de classificação múltipla de aprendizado supervisionado. Para isso, os recursos precisam ser convertidos em vetores e mapeados com os rótulos. Fu et al. (2018) citam que, em vez do uso da representação vetorial em *One-Hot*, que se torna inviável em bases de dados grandes, o modelo de *Word Embedding*, utilizado em estudos de Processamento de Linguagens Naturais, foi adotado.

A camada totalmente conectada é utilizada para o aprendizado dos interesses

do usuário através do uso de uma Rede Neural de Memória de Longo Prazo bidirecional. Seja $h_b(I_i)$ um vetor com valores reais que representam o interesse do usuário antes de consumir um item I_i; $h_a(I_i)$ um vetor com valores reais que representam o interesse do usuário depois de consumir um item I_i; $W^{(b)}$ uma matriz que transforma a camada oculta na próxima camada oculta; $W^{(eb)}$ uma matriz que é usada para combinar o interesse do item atual com o interesse anterior do próximo item; e δ uma função de ativação não-linear, os vetores de interesse são calculados a partir da Equação (2.33) e da Equação (2.34)

$$h_b(I_i) = \delta(W^{(b)}h_b(I_{i-1}) + W^{(eb)}e(I_{i-1})) \tag{2.33}$$

$$h_a(I_i) = \delta(W^{(a)}h_a(I_{i+1}) + W^{(ea)}e(I_{i+1})) \tag{2.34}$$

O status temporário do interesse do usuário quando o este assume o comportamento de I_i é definido na Equação (2.35) ao concatenar o status temporário anterior de interesse $h_b(I_i)$ antes de consumir o item I_i, o comportamento incorporado I, o item consumido $e(I_i)$ e o status temporário subsequente de interesse $h_a(I_i)$ após o consumo do item I_i.

$$x_i = [h_b(I_i); e(I_i); h_a(I_i)] \tag{2.35}$$

Por fim, é aplicado uma transformação linear através da função de ativação da tangente hiperbólica em x_i, conforme definido na Equação (2.36).

$$y_i^{(2)} = tanh(W^{(2)}x_i + b^{(2)}) \tag{2.36}$$

A camada de *pooling* é responsável por aplicar um filtro de maximização nas sub-regiões não sobrepostas da representação superior, determinado pela Equação (2.37).

$$y^{(3)} = \max_{i=1}^{n} y_i^{(2)} \tag{2.37}$$

A última parte do modelo inclui a camada de saída definida pela Equação (2.38). A função de ativação *softmax* também é aplicada em $y^{(4)}$ para converter os valores em probabilidades dos próximos itens.

$$y^{(4)} = W^{(4)}y^{(3)} + b^{(4)} \tag{2.38}$$

O modelo foi avaliado através das métricas de *Accuracy*, de *Precision*, de *Recall* e de *F1-Score*, utilizando os dados de um banco de dados de um *site* Chinês de *live streaming* e comparado com outros quatro modelos recorrentes do Estado da Arte: predição de popularidade, filtragem colaborativa por item, filtragem colaborativa por usuários e Redes de Memória de Longo Prazo convencionais. Os resultados apontam que o modelo proposto por Fu et al. (2018) se mostra superior em quase todas as métricas avaliadas, tendo pequena diferença apenas quando comparado com Redes de Memória de Longo Prazo convencionais.

Heinz, Bracher e Vollgraf (2017) também utilizaram em seu trabalho um modelo de Rede de Memória de Longo Prazo para predizer itens de interesse para um usuário de um sistema de *e-commerce* de moda, levando em consideração seu histórico de compras. No processo, Heinz, Bracher e Vollgraf (2017) avaliaram três modelos de perda para identificar qual o melhor uso de acordo com o resultado: perda de entropia-cruzada por sigmoide, perda por *rank* de sigmoide e perda por *softmax*. A comparação entre os modelos identificou resultados parecidos entre a perda de entropia-cruzada por sigmoide e a perda por *rank* de sigmoide, enquanto que a perda por *softmax* se mostrou o pior modelo.

2.5.3.2 Redes de Unidades Recorrentes de Bloqueio

Redes de Unidades Recorrentes de Bloqueio se mostram mais simples e menos redundantes que as Redes de Memória de Longo Prazo, tendo seu tempo de treinamento reduzido entre 20% a 30%. Apesar de existirem demonstrações de que não há diferença de qualidade dependendo da forma como as Redes de Memória de Longo Prazo são inicializadas, as Redes de Unidades Recorrentes de Bloqueio ainda garantem uma performance superior (KOROTAEV; LYADOVA, 2018).

Korotaev e Lyadova (2018) definiram uma Rede de Unidades Recorrentes de Bloqueio tendo como entrada a atual sessão do usuário, com o objetivo de predizer sua próxima sessão. Para otimizar o processo de memória, a entrada foi codificada utilizando o algoritmo de "*1-of-N*"e normalizada. Contudo, os autores Korotaev e Lyadova (2018) não realizam nenhuma comparação com outros métodos para avaliar a eficácia do modelo.

Já Chu et al. (2017) propuseram o uso de Redes de Unidades Recorrentes de Bloqueio com *Retropropagation* para a recomendação de vídeos, calculando a probabilidade de um vídeo que um usuário possa gostar levando em consideração os vídeos anteriores que foram assistidos. Como as Redes de Unidades Recorrentes de Bloqueio possuem a mesma camada de saída que as Redes de *Retropropagation*, as camadas de saída de ambas as redes foram unificadas para formar um único resultado.

No modelo utilizado, a Rede de Unidades Recorrentes de Bloqueio teve as

camadas ocultas definidas pela Equação (2.39), enquanto a camada de saída foi definida pela Equação (2.40).

$$s_t = tanh(Ux_t + Ws_{t-1}) \tag{2.39}$$

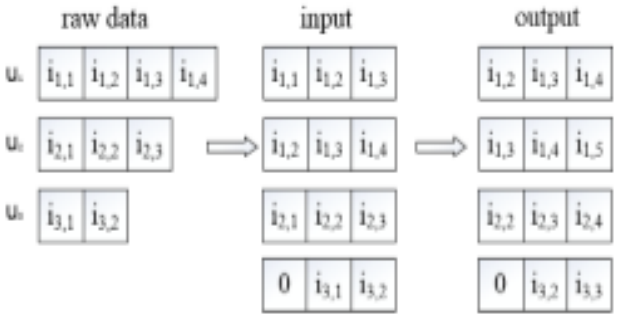
$$o_t = softmax(Vs_t) \tag{2.40}$$

A entropia cruzada é utilizada como a função de perda definida pela Equação (2.41), onde o_t representa o filme correto no tempo t e $\hat{o}_t$ representa a precisão utilizada no modelo.

$$E_t(o_t, \hat{o}_t) = -o_t \log \hat{o}_t \tag{2.41}$$

A redução do erro é calculada através do uso do algoritmo de *Retropropagation* para calcular os gradientes. Para resolver o problema de quando a quantidade de filmes assistidos por um usuário é diferente do tamanho dos filmes da base de treinamento, a entrada da rede foi adaptada conforme apresentada na Figura (16).

Figura 16 – Adaptação da entrada da rede para resolver o problema de quando aa quantidade de filmes assistidos por um usuário é diferente do tamanho do filmes da base de treinamento.



Fonte: Fu et al. (2018)

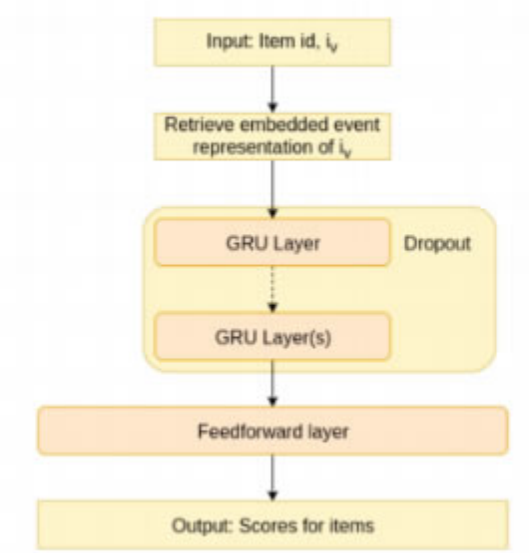
Para a avaliação do modelo, a base de dados do *MovieLens* foi utilizada para obter 100.000 avaliações de 943 usuários em 1.682 filmes, uma vez cada usuário avaliou pelo menos 20 filmes. Para a entrada na rede, apenas os ids dos filmes foram considerados. O vetor de entrada possui o tamanho do número de filmes assistidos por um usuário u_j, marcando o filme i_j assistido como 1 e os demais como 0.

Em comparação com redes com números de camadas variando entre 3 e 7, Chu et al. (2017) identificaram que a precisão do modelo é maior quando utilizado com quatro camadas, sendo superior se comparado com a precisão de outros modelos.

Os autores Ruocco, Skrede e Langseth (2017) propõem um modelo que utiliza duas Redes de Unidades Recorrentes de Bloqueio com a proposta de processar a sessão atual de um usuário em uma Rede Neural Recorrente para, em seguida, processar o histórico de sessões do mesmo usuário em uma segunda Rede Neural Recorrente. Dessa forma, a segunda Rede Neural Recorrente atua como uma camada oculta da primeira.

A primeira rede conta com camadas de Unidades Recorrentes de Bloqueio conectadas a uma camada de *Feedfoward*, gerando um vetor de valores entre 0 e 1. Nesse caso, é esperado que todos os itens do vetor de saída tenham valor 0, exceto pelo item mais relevante, representado pelo valor 1. Dessa forma, o problema de recomendação é tratado como um problema de classificação.

Figura 17 – Arquitetura da Rede Neural Recorrente utilizando camadas de Unidades Recorrentes de Bloqueio.



Fonte: Ruocco, Skrede e Langseth (2017)

A segunda rede também utiliza uma camada de Unidades Recorrentes de Bloqueio para processar as demais sessões do usuário, provendo um histórico prévio para a primeira rede proposta pelo modelo. O modelo foi avaliado através das métricas de *Recall* e de Média de Classificação Recíproca, além de comparado com outros modelos de recomendação, inclusive de Redes Neurais Recorrentes tradicionais. Os resultados

apontam valores superiores nas métricas avaliadas para a arquitetura proposta por Ruocco, Skrede e Langseth (2017).

2.5.3.3 Outros modelos de arquitetura

Ao longo do trabalho de Li et al. (2018), é apresentada uma arquitetura de Redes Neurais Recorrentes que utiliza *Retropropagation* através do modelo simplificado de Adagrad (DUCHI; HAZAN; SINGER, 2011) que, por sua vez, utiliza apenas a taxa inicial do aprendizado como parâmetro.

Os dados foram coletados da base do MovieLens, contando com mais de um milhão de avaliações dadas por 6.040 usuários para 3.706. A rede conta com duas camadas ocultas, enquanto a rede foi truncada com um valor fixo de 50 passos. Para avaliação do modelo, os resultados foram comparados com outros quatro métodos utilizados em sistemas de recomendação: Filtragem Colaborativa Baseada em Itens; Filtragem Colaborativa Baseada em Usuários; Decomposição de Valor Singular; e Fatorização de Matriz para avaliações implícitas, avaliando cada método través das métricas de *Precision*, de *Recall* e de *F1-Score*.

Os resultados foram medidos através dos dez itens de melhor avaliação para cada método, inclusive o modelo proposto no trabalho. De acordo com os resultados apresentados por Li et al. (2018), o modelo proposto garante um aumento de 1,17% de *Precision*, 4,02% de *Recall* e 1,81% de *F1-Score*.

2.5.4 Consolidação do Estado da Arte

Os trabalhos revisados nesta seção apresentam diferentes soluções para o uso de redes recorrentes em sistemas de recomendação, mostrando também a viabilidade do uso de dados de sessão do usuário como insumo para os modelos propostos. Considerando os estudos como os de Heinz, Bracher e Vollgraf (2017); Chu et al. (2017); e Kong, Li e Lv (2018), que utilizam os dados de sessão de maneira simplificada (repassando o mínimo de dados como entrada da rede), este trabalho considera o mesmo modelo a fim de permitir maior adaptação da arquitetura em diferentes domínios.

Foi também identificada uma tendência no uso de dois modelos arquiteturais para Redes Neurais Recorrentes nos trabalhos da revisão: Redes de Memória de Longo Prazo e Redes de Unidades Recorrentes de Bloqueio. Enquanto trabalhos como o de Fu et al. (2018) deixam claro a melhoria de performance de uma Rede de Memória de Longo Prazo Bidirecional se comparado com seu modelo tradicional, os trabalhos que utilizam Redes de Unidades Recorrentes de Bloqueio não deixam explícita a melhoria de performance de seus modelos adaptados em relação ao modelo original, ou apresentam pouca melhoria.

Além disso, os demais trabalhos apresentam poucos ou nenhum comparativo entre Redes de Memória de Longo Prazo e Redes de Unidades Recorrentes de Bloqueio. Dessa forma, este trabalho utiliza o modelo de Rede de Memória de Longo Prazo para comparar a performance da predição em relação a uma Rede de Unidades Recorrentes de Bloqueio tradicional.

Também é realizada uma comparação com o modelo híbrido que utiliza os conceitos arquiteturais apresentados no trabalho de Anil et al. (2018). Como métricas de comparação, são utilizados os métodos de *Accuracy*, de *Recall*, de *Precision* e de *F1-Score*, presentes na maioria dos trabalhos desta revisão, além das métricas convencionais para sistemas de recomendação.

A Tabela (5) apresenta a relação dos trabalhos desta revisão e seus vínculos com este trabalho.

Tabela 5 – Relação de trabalhos revisados com vínculo no trabalho proposto

Trabalho	**Vínculo**
D. Anil, A. Vembar, S. Hiriyannaiah, S. GM and K. Srinivasa, "Performance Analysis of Deep Learning Architectures for Recommendation Systems"	Proposta de modelo híbrido de Redes de Memória de Longo Prazo e Redes de Unidades Recorrentes de Bloqueio
W. Yuan, Y. Yang and X. Bao, "Parallel DNNs for users and items modeling and recommendation using comments"	Uso de métodos comparativos entre Redes Neurais Recorrentes e Sistemas de Recomendação
Massimiliano Ruocco, Ole Steinar Lillestøl Skrede, and Helge Langseth. 2017. Inter-Session Modeling for Session-Based Recommendation	Uso de proposta arquitetural de uma Rede de Unidades Recorrentes de Bloqueio

Kong, F., Li, J., Lv, Z. Construction of intelligent traffic information recommendation system based on long short-term memory	Uso de proposta arquitetural de uma Rede de Memória de Longo Prazo
Korotaev, A., Lyadova, L. Method for the development of recommendation systems, customizable to domains, with deep GRU network	Uso de proposta arquitetural de uma Rede de Unidades Recorrentes de Bloqueio
Zhou, T., Qian, H., Shen, Z., Zhang, C., Wang, C., Liu, S., Ou, W. JUMP: A joint predictor for user click and dwell time	Uso de bases de dados e modelos de comparação de performance entre modelos
Fu, H., Li, J., Chen, J., Tang, Y., Zhu, J. Sequence-based recommendation with bidirectional LSTM network	Uso de proposta arquitetural de uma Rede de Memória de Longo Prazo
Chu, Y., Huang, F., Wang, H., Li, G., Song, X. Short-term recommendation with recurrent neural networks	Uso de proposta arquitetural de uma Rede de Memória de Longo Prazo

Fonte: O autor.

3 DEFINIÇÃO DO EXPERIMENTO

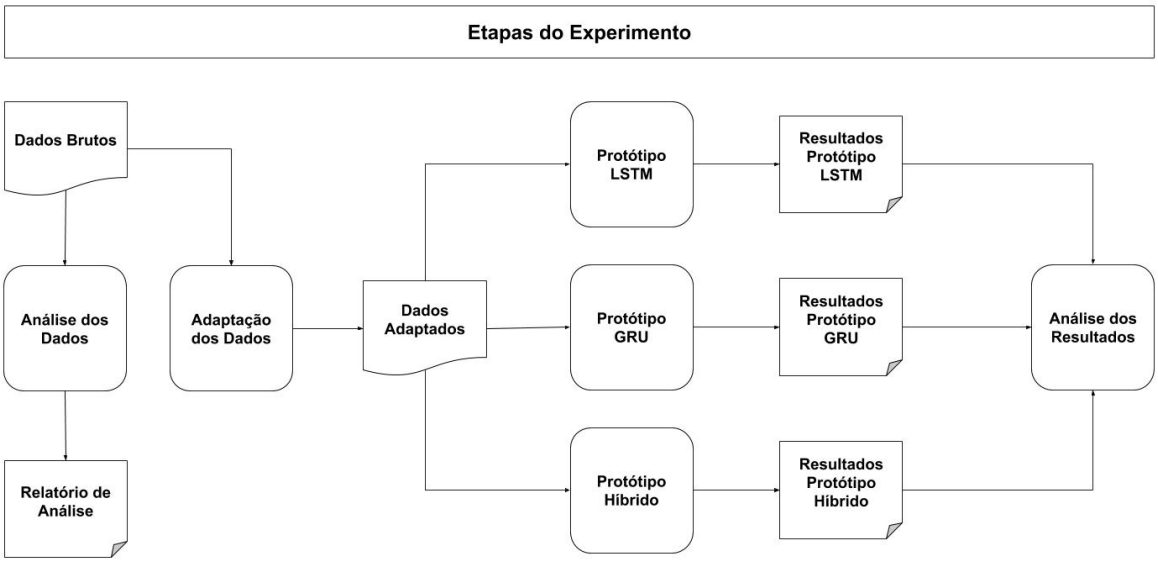
Esta seção tem como objetivo apresentar os detalhes sobre a base de dados e a configuração utilizada no experimento para este projeto. Para que seja possível realizar a avaliação da recomendação entre diferentes tipos de arquiteturas de Redes Neurais Recorrentes, foi proposta a construção de três protótipos que utilizam as arquiteturas de Redes de Memória de Longo Prazo, de Redes de Unidades Recorrentes de Bloqueio e de Redes Híbridas.

Conforme identificado no Estado da Arte desta pesquisa, apesar das três arquiteturas citadas aparecerem com frequência como as arquiteturas de Redes Neurais Recorrentes utilizadas para se lidar com dados sequenciais, como o histórico de navegação do usuário, são inexistentes os trabalhos que comparam os resultados das arquiteturas entre si. Comparar os resultados dos trabalhos de forma independente pode aumentar o viés dos resultados finais, uma vez que as fontes de dados e parâmetros utilizados pelos demais autores são diferentes.

Desta forma, um dos principais objetivos deste experimento é utilizar os mesmos parâmetros e a mesma base de dados nas três arquiteturas de Redes Neurais Recorrentes, de tal forma que seja possível avaliar sua performance evitando viés durante o processo. Isto é feito através da realização de diferentes experimentos que variam conforme o parâmetro avaliado, obtendo os valores de *Accuracy*, de *Precision*, de *Recall* e de *F1-Score*.

O experimento contou com diferentes etapas. Primeiramente, os dados passaram por um processo de análise, identificando os picos de acesso e os casos que podem interferir no resultado final do trabalho. A partir da análise, os dados foram adaptados para a execução dos protótipos, divididos em uma base para treinamento e outra para testes. Em seguida, a construção dos protótipos foi realizada com base nos dados coletados através do Estado da Arte, definido os parâmetros de entrada para cada um e os parâmetros específicos de cada arquitetura. Por fim, os três protótipos foram executados de forma independente utilizando os dados adaptados e gerando seus resultados em diretórios específicos.

Figura 18 – Etapas do Experimento.



Fonte: O autor.

As seções a seguir explicam, em detalhes, cada etapa da execução do experimento.

3.1 Análise dos dados de entrada

Os dados utilizados neste experimento representam o histórico de navegação de um site chinês de *live streaming*, conforme utilizado no trabalho de Fu et al. (2018) e disponibilizado publicamente através do site Kaggle (KAGGLE,).

Cada linha do arquivo de dados representa uma sequência de navegação de um usuário, contendo identificadores numéricos de cada página visitada em ordem cronológica. Para analisar os dados foi criado um *script*, disponível no Apêndice C, utilizando a linguagem Python na versão 2.7.15, com o objetivo de quantificar os acessos de páginas por usuário e a frequência de acesso. Esse *script* foi executado em um sistema Ubuntu versão 18.04.2 LTS com 16 GB de RAM e processador do modelo Intel Core i7-7700.

O objetivo deste *script* é a contabilizar e agrupar valores de métricas que permitam a análise dos dados antes de adaptá-los. De forma iterativa, cada linha do arquivo foi processada, contabilizando a quantidade de sessões por usuário, total de páginas visitadas por todos os usuários e a quantidade de páginas visitadas por sessão. Além disto, também foi considerada a frequência em que uma página é acessada e a quantidade de vezes em que um usuário acessa a mesma página. Em seguida, foi

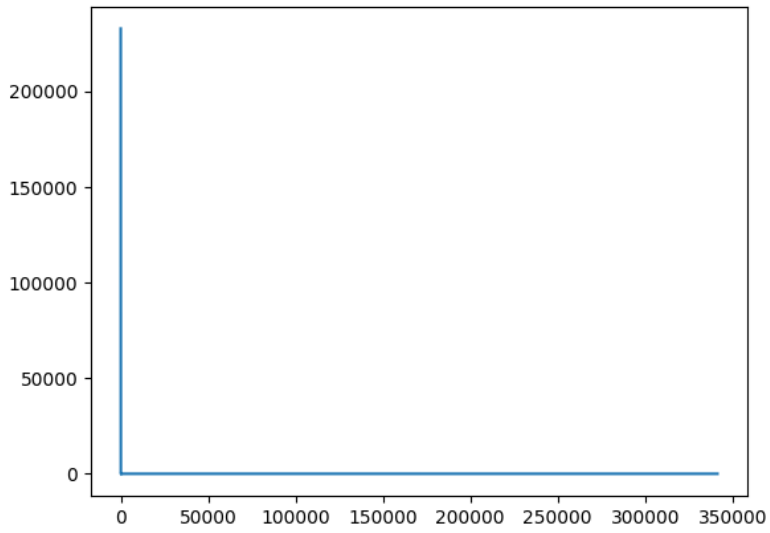
calculada a média de acessos por usuário, a maior frequência de acessos em uma única página, a menor frequência de acessos em uma única página, a maior sequência de páginas que um usuário acessou e a menor sequência de páginas que um usuário acessou.

No total, o arquivo de dados contabiliza 1.806.204 linhas representando usuários únicos, 18.927.700 de acessos a páginas e 453.423 páginas distintas acessadas. O número de acessos varia entre 1 e 1.060 páginas em uma mesma sessão, com uma média de 10 acessos diferentes para cada usuário, na qual a página menos visitada teve pelo menos 1 acesso de um usuário e a mais visitada teve acesso de 341.471 usuários.

Para o cálculo da frequência de acessos de cada página, foi contabilizada a quantidade de vezes que cada página foi acessada para, em seguida, ser agrupada a quantidade de páginas por quantidade de acessos. Já para o cálculo da frequência de usuários por quantidade de acessos por página, foi contabilizada a quantidade de páginas acessadas por cada usuário para, em seguida, agrupar a quantidade de usuários pela quantidade de acessos por página.

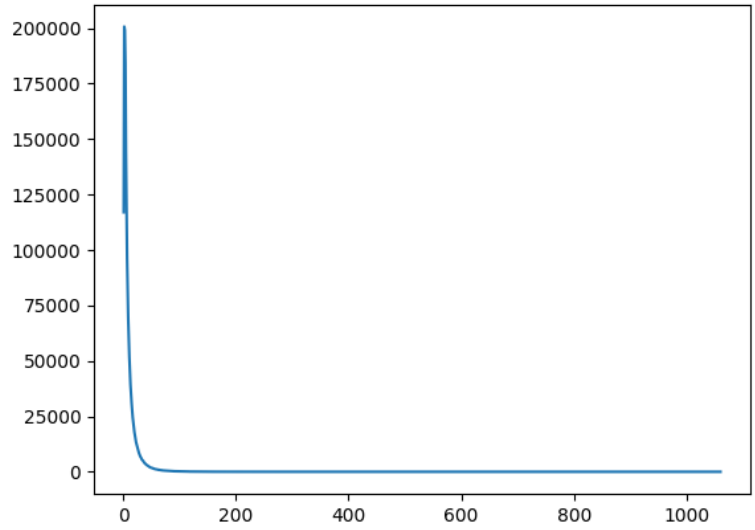
Os resultados apresentaram picos na quantidade de páginas que tiveram apenas um acesso, ou usuários cuja sessão conta apenas com um acesso, conforme apresentado nas Figuras (3.1 e 3.1). A biblioteca *matplotlib* do Python foi utilizada com o propósito de gerar os gráficos para a análise das frequências dos dados.

Figura 19 – Frequência de acessos de cada página pela quantidade de vezes que a página foi acessada.



Fonte: O autor

Figura 20 – Frequência de acessos de cada página pela quantidade de usuários.



Fonte: O autor

3.2 Adaptação dos dados de entrada

Considerando que sessões com acesso a uma única página não possuem valor para esta pesquisa, uma vez que tais sessões não possuem valor de aprendizado em relação a trajetória do usuário para a página-destino, foram descartadas da base de entrada. Logo, foram consideradas apenas sessões que possuem pelo menos duas páginas acessadas por um mesmo usuário.

Além da remoção das sequências com apenas um único acesso de um mesmo usuário, não houve necessidade de outras alterações dos dados de entrada para uso nos protótipos. Inicialmente, os dados foram divididos em dois grupos contendo 80% dos dados para a base de treinamento e 20% para a base de testes seguindo o Princípio de Pareto Robert (1987).

Durante a execução do experimento, o modelo parou seu treinamento devido ao alto uso de memória para armazenamento das variáveis. Este foi um dos grandes desafios encontrados durante a execução deste experimento, uma vez que houve uma limitação física da memória máquina disponível ao utilizar todo o conjunto de dados de entrada. Isto se deu devido a grande quantidade de memória necessária para realizar o *One-Hot Encoding* dos dados, processo fundamental para que os dados possam ser tratados pelas redes. Dessa forma, o volume de dados foi reduzido iterativamente até atingir a proporção de 10% de seu total, permitindo a execução do experimento sem falhas de memória.

3.3 Preparação do Experimento

A construção dos protótipos deste projeto tem como objetivo permitir a análise da performance do uso do histórico de navegação do usuário em Redes Neurais Recorrentes para recomendação de itens. Por permitir agilidade e praticidade para se criar as redes nas arquiteturas propostas, a biblioteca Keras (CHOLLET et al., 2015) foi utilizada através da linguagem Python versão 2.7.15. Os passos para a configuração do ambiente de desenvolvimento estão disponíveis no Apêndice E.

Após a execução do *script* definido na seção anterior, a base de dados foi dividida em dados de treinamento e em dados de teste. Durante a criação desses protótipos, executados em um sistema Ubuntu versão 18.04.2 LTS com 16 GB de RAM e processador do modelo Intel Core i7-7700, os modelos pararam a execução devido a alto uso de memória da máquina.

Considerando as arquiteturas de Redes Neurais Recorrentes a serem utilizadas neste experimento, foi criado um *script* para treinamento de uma Rede de Memória de Longo Prazo, de uma Rede de Unidades Recorrentes de Bloqueio e de uma Rede Híbrida que unificasse ambos os modelos. Isso é definido pela biblioteca Keras ao configurar o modelo como sequencial e determinado através de uma variável no *script* desenvolvido, disponível no Apêndice D.

Os modelos de treinamento também determinam um limite máximo do número de sequências para cada linha do arquivo de entrada. Tal limite foi tratado como uma variável do modelo, sendo alterado para identificar a mudança dos resultados e o tempo de execução do modelo de treinamento, conforme realizado por Fu et al. (2018).

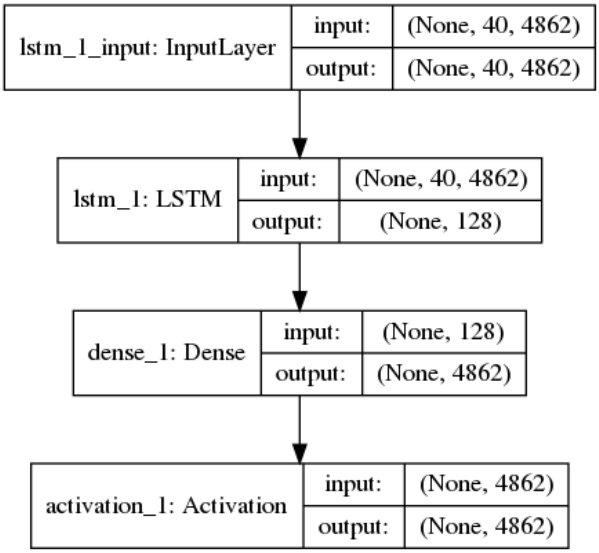
Para uso dos dados no modelo, cada linha do arquivo de entrada foi separada em dois conjuntos de dados, onde o conjunto X representa a sequência de páginas percorrida por cada usuário e o conjunto y representa a última página acessada por cada usuário. Em seguida, os dados foram transformados utilizando o algoritmo de *One-Hot Encoding*, permitindo o uso de dados categóricos como dados numéricos. Nesse processo, o número de sequências e o número de páginas acessadas foram armazenados para identificação, como o tamanho do número de sentenças e o tamanho do vocabulário do sistema para comparação dos resultados.

Cada arquitetura varia de acordo com as principais camadas adicionadas ao modelo, levando em consideração as arquiteturas propostas nos trabalhos de Anil et al. (2018); Ruocco, Skrede e Langseth (2017); Kong, Li e Lv (2018); Korotaev e Lyadova (2018); Fu et al. (2018); e Chu et al. (2017).

No protótipo utilizando a arquitetura de uma Rede de Memória de Longo Prazo, foi criado um modelo sequencial com três camadas: uma camada de Memória de Longo Prazo com 128 unidades, uma camada Densa com ativação a partir da função

linear $a(x) = x$ e uma camada de ativação variando a função de ativação utilizada.

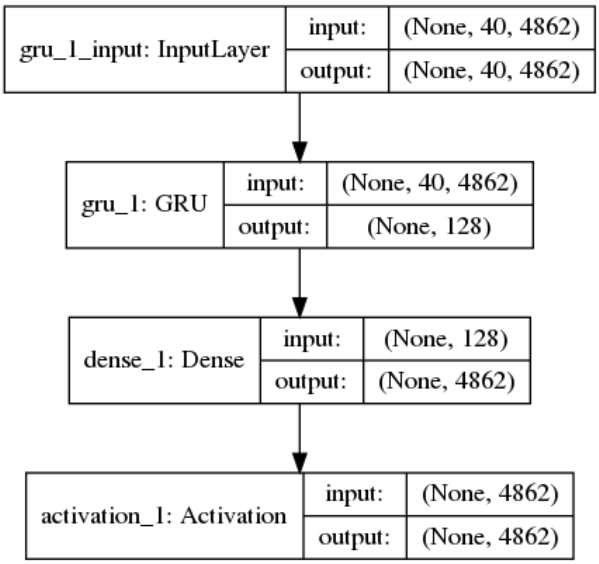
Figura 21 – Representação gráfica da arquitetura do protótipo de Memória de Longo Prazo.



Fonte: O autor

No protótipo utilizando a arquitetura de uma Rede de Unidades Recorrentes de Bloqueio, também foi criado um modelo sequencial com três camadas: uma camada de Unidades Recorrentes de Bloqueio com 128 unidades, uma camada Densa com ativação a partir da função linear $a(x) = x$ e uma camada de ativação variando a função de ativação utilizada.

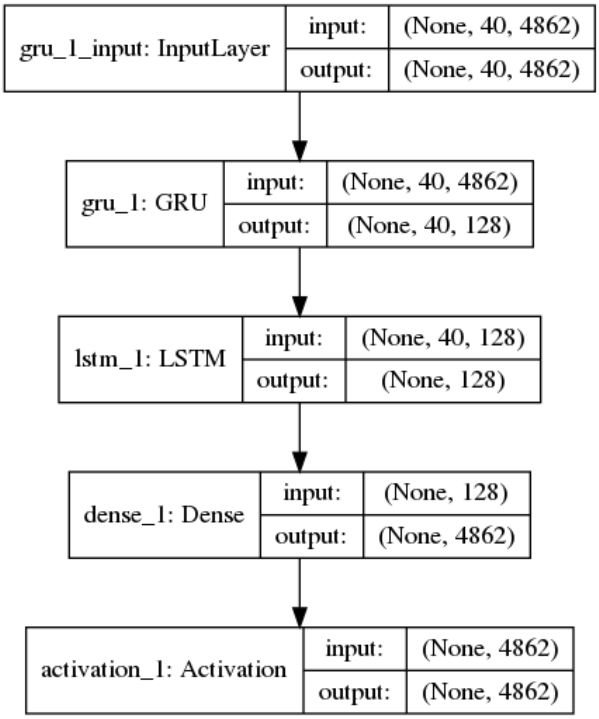
Figura 22 – Representação gráfica da arquitetura do protótipo de Unidades Recorrentes de Bloqueio.



Fonte: O autor

Já no protótipo utilizando a arquitetura de uma rede híbrida, foi criado um modelo sequencial com quatro camadas: uma camada de Memória de Longo Prazo com 128 unidades, uma camada de Unidades Recorrentes de Bloqueio com 128 unidades, uma camada Densa com ativação a partir da função linear $a(x) = x$ e uma camada de ativação variando a função de ativação utilizada.

Figura 23 – Representação gráfica da arquitetura do protótipo com a Rede Híbrida.



Fonte: O autor

A tabela 6 apresenta as variáveis e o conjunto de valores utilizados durante o experimento deste trabalho

Tabela 6 – Lista de variáveis e valores utilizados no experimento

Variável	**Valores**
Função de ativação	softmax, tanh e sigmoid
Otimizador	SGD, Adam, RMSprop e Adagrad
Tamanho do *batch*	64, 128, 256
Épocas	20, 50, 100

Fonte: O autor.

No total, foram gerados 108 experimentos a partir da combinação das variáveis. A lista de cada experimento e suas respectivas variáveis está disponível no Apêndice F.

3.4 Execução do Experimento

Os protótipos definidos na subseção anterior foram executados conforme definido no Apêndice D, a partir de uma instância do tipo t2.2xlarge da Amazon, com sistema operacional Linux, 32 GB de RAM e processador do modelo Intel de 3,0 GHz. A execução dos protótipos ocorreu de forma simultêna, treinando as 3 arquiteturas de rede ao mesmo tempo.

A performance durante o treinamento das redes variou conforme a arquitetura utilizada e os parâmetros de entrada. As redes LSTM e GRU tiveram tempos equivalentes de treinamento, enquanto a rede híbrida demorou mais para finalizar sua execução. Da mesma forma, notou-se um aumento no tempo de execução de cada época das redes com a alteração do tamanho do *batch*, enquanto os parâmetros como função de ativação e otimizador utilizado pouco variaram na performance durante o treinamento.

4 RESULTADOS DO EXPERIMENTO

Esta seção apresenta os resultados obtidos através da execução do experimento definido na seção anterior. As subseções a seguir apresentam os resultados da execução dos protótipos a partir dos demais cenários especificados, considerando as métricas de *Accuracy*, de *Precision*, de *Recall* e de *F1-Score*. O relatório completo dos resultados está disponível no Apêndice G.

4.1 Avaliação Geral dos Resultados

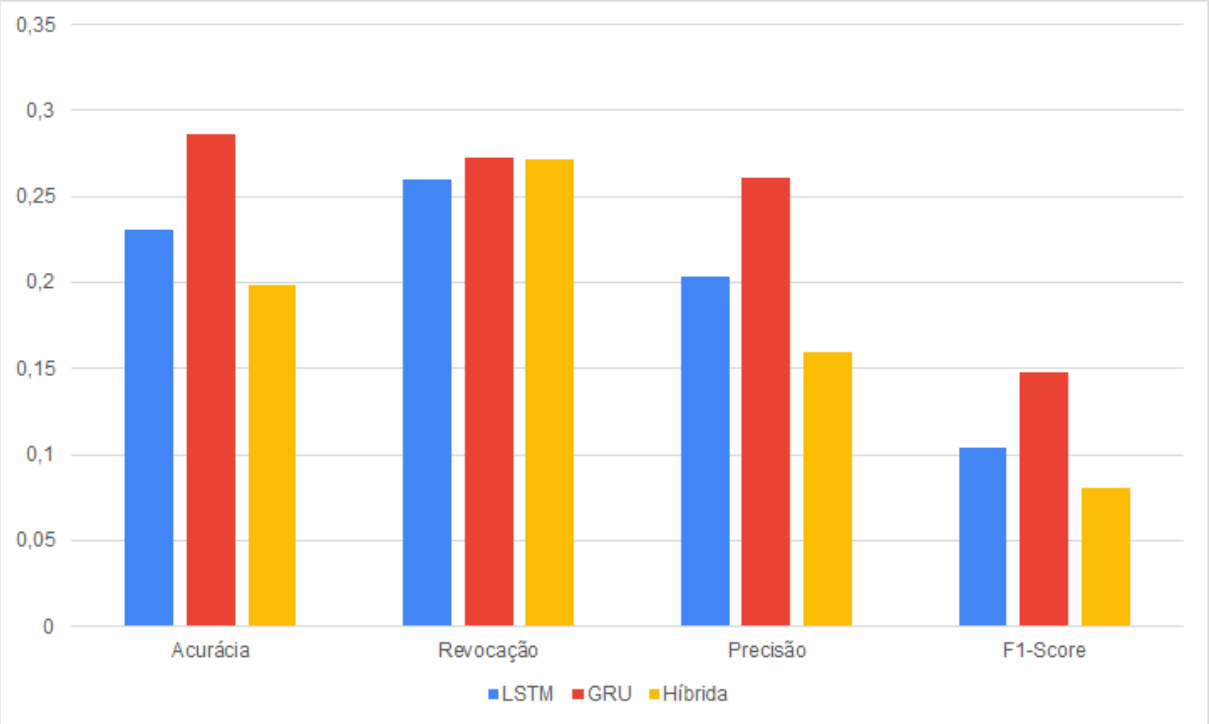
A partir dos resultados gerados, foi calculada a média geral de cada uma das métricas de avaliação, conforme representado na Tabela 7 e na Imagem 24. Os dados demonstram uma maior generalização da arquitetura de GRU quanto às demais variáveis no modelo, tendo um bom desempenho em uma maior quantidade de cenários que as demais arquiteturas – enquanto a arquitetura de Rede Híbrida possui um bom desempenho em uma quantidade menor de cenários.

Tabela 7 – Média geral das métricas por arquitetura de rede

Arquitetura	***Accuracy***	***Recall***	***Precision***	**F1-Score**
LSTM	0,23048	0,26033	0,20371	0,10427
GRU	0,28590	0,27280	0,26073	0,14817
Híbrida	0,19879	0,27212	0,15960	0,08083

Fonte: O autor.

Figura 24 – Gráfico da média geral das métricas por arquitetura de rede.



Fonte: O autor

Analisando os resultados a partir da arquitetura de rede utilizada, percebe-se um desempenho superior da arquitetura de GRU, obtendo valores maiores de *Accuracy* e *F1-Score* com maior frequência se comparado às demais arquiteturas. A arquitetura de LSTM se mostra superior em casos específicos, enquanto as Rede Híbrida não demonstra uma performance superior em nenhum caso se comparado utilizando as métricas de avaliação citadas.

Entretanto, a performance das arquiteturas varia conforme a combinação dos parâmetros utilizados. A arquitetura de GRU possui maiores valores de *Accuracy* e *F1-Score* quando utilizado em conjunto com o otimizador *RMSProp* e função de ativação *softmax*, tendo também valores mais elevados conforme o tamanho do *batch*. Ou seja: quanto mais elevado o tamanho do *batch* ao utilizar Redes GRU juntamente com o otimizador *RMSProp*, melhor será sua performance.

Da mesma forma, a arquitetura de LSTM teve um desempenho superior quando comparado as demais redes quando utilizada em conjunto com a função de ativação *sigmoid* e o otimizador *RMSProp*, indiferente ao tamanho do *batch* nestes casos.

4.2 Avaliação dos Resultados a partir da função de ativação

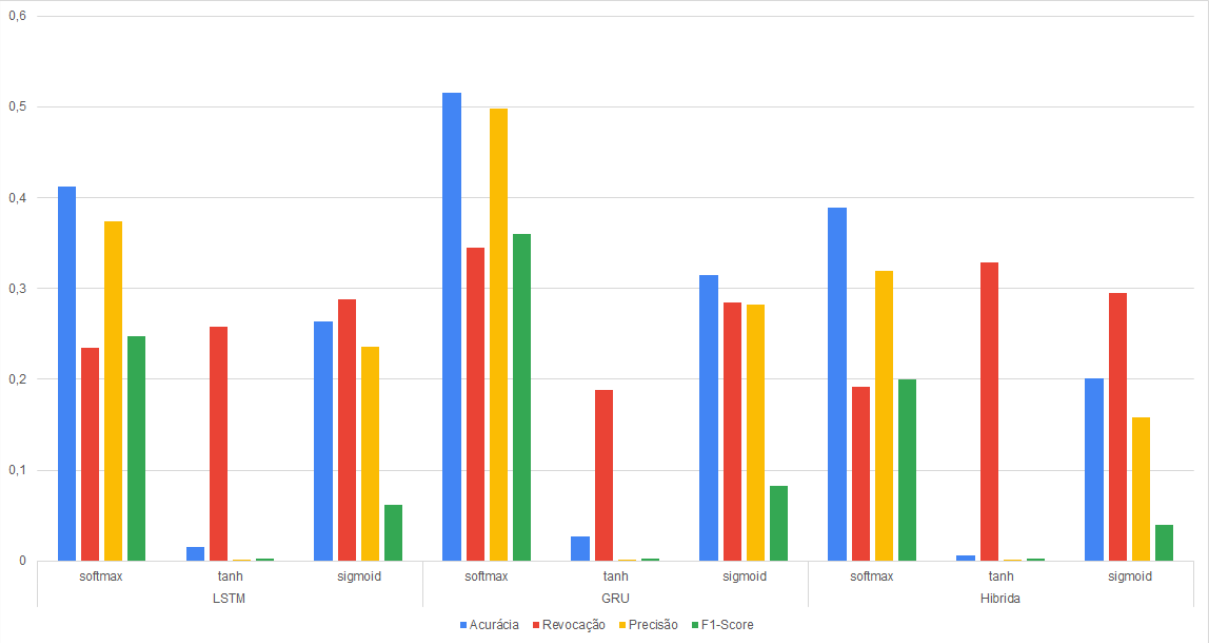
Agrupando os resultados a partir da função de ativação, é possível analisar a performance das arquiteturas a partir das funções utilizadas no treinamento do modelo deste experimento, conforme apresentado na Tabela 8 e na Imagem 25. A função *softmax* apresenta melhor desempenho em comparação com as demais funções utilizadas, enquanto a tangente hiperbólica possui o menor desempenho entre as demais funções utilizadas.

Tabela 8 – Média geral das métricas por arquitetura de rede e função de ativação

Arquitetura	**Ativação**	***Accuracy***	***Recall***	***Precision***	**F1-Score**
LSTM	softmax	0,41270	0,23449	0,373887	0,24795
	tanh	0,01521	0,25799	0,00125	0,00245
	sigmoid	0,26354	0,28850	0,23601	0,06241
GRU	softmax	0,51600	0,345299	0,49862	0,35953
	tanh	0,02739	0,18848	0,00131	0,00220
	sigmoid	0,31430	0,28463	0,28226	0,08276
Híbrida	softmax	0,38940	0,19212	0,31979	0,19942
	tanh	0,00578	0,32912	0,00141	0,00274
	sigmoid	0,20120	0,29513	0,15761	0,04033

Fonte: O autor.

Figura 25 – Gráfico da média geral das métricas por arquitetura de rede e função de ativação.



Fonte: O autor

As funções de ativação têm menor influência na melhoria dos valores avaliados do resultado do que os demais parâmetros utilizados. Observa-se, no entanto, que as métricas de *Accuracy*, de *Precision* e de *F1-Score* são maiores quando utilizada a função *softmax* do que a *sigmoid*. Já a função *tanh* influencia diretamente nos baixos valores das métricas observadas, independente dos valores dos demais parâmetros.

4.3 Avaliação dos Resultados a partir do número de épocas

Agrupando os resultados a partir do número de épocas, é possível analisar a performance das arquiteturas a partir da quantidade de épocas utilizadas no treinamento do modelo deste experimento, conforme apresentado na Tabela 9 e na Imagem 26.

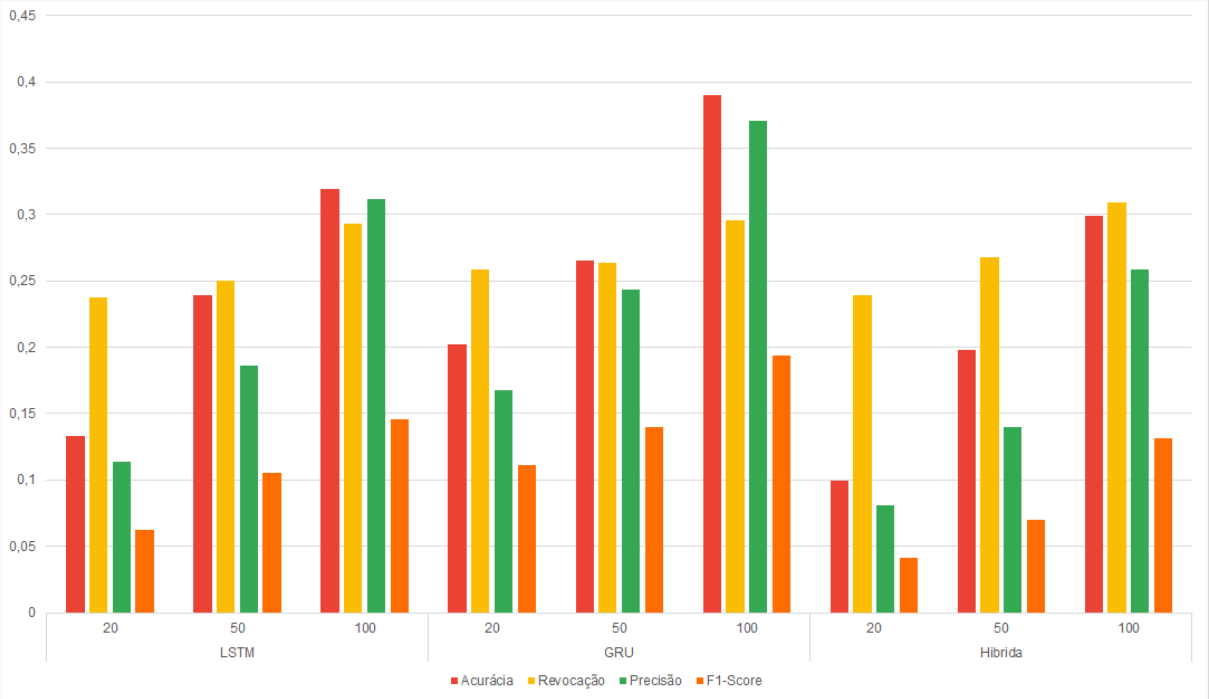
Tabela 9 – Média geral das métricas por arquitetura de rede e quantidade de épocas utilizadas

Arquitetura	**Nº de épocas**	***Accuracy***	***Recall***	***Precision***	**F1-Score**
LSTM	20	0,13286	0,23715	0,11341	0,06205
	50	0,23896	0,25046	0,18596	0,10495
	100	0,31963	0,29336	0,31177	0,14581
GRU	20	0,20235	0,25852	0,16794	0,11092

	50	0,26509	0,26406	0,24347	0,13949
	100	0,39026	0,29583	0,37079	0,19408
Híbrida	20	0,09924	0,23904	0,08087	0,04140
	50	0,19824	0,26825	0,13955	0,06982
	100	0,29891	0,30908	0,25838	0,13128

Fonte: O autor.

Figura 26 – Gráfico da média geral das métricas por arquitetura de rede e quantidade de épocas utilizada.



Fonte: O autor

Ao analisar os resultados a partir do número de épocas utilizado, percebe-se que quanto maior o número de épocas, maior a qualidade dos resultados. Isto acontece independente dos demais parâmetros utilizados, porém o valor tem pouca influência no aumento dos valores das métricas, melhorando discretamente os resultados.

4.4 Avaliação dos Resultados a partir do otimizador

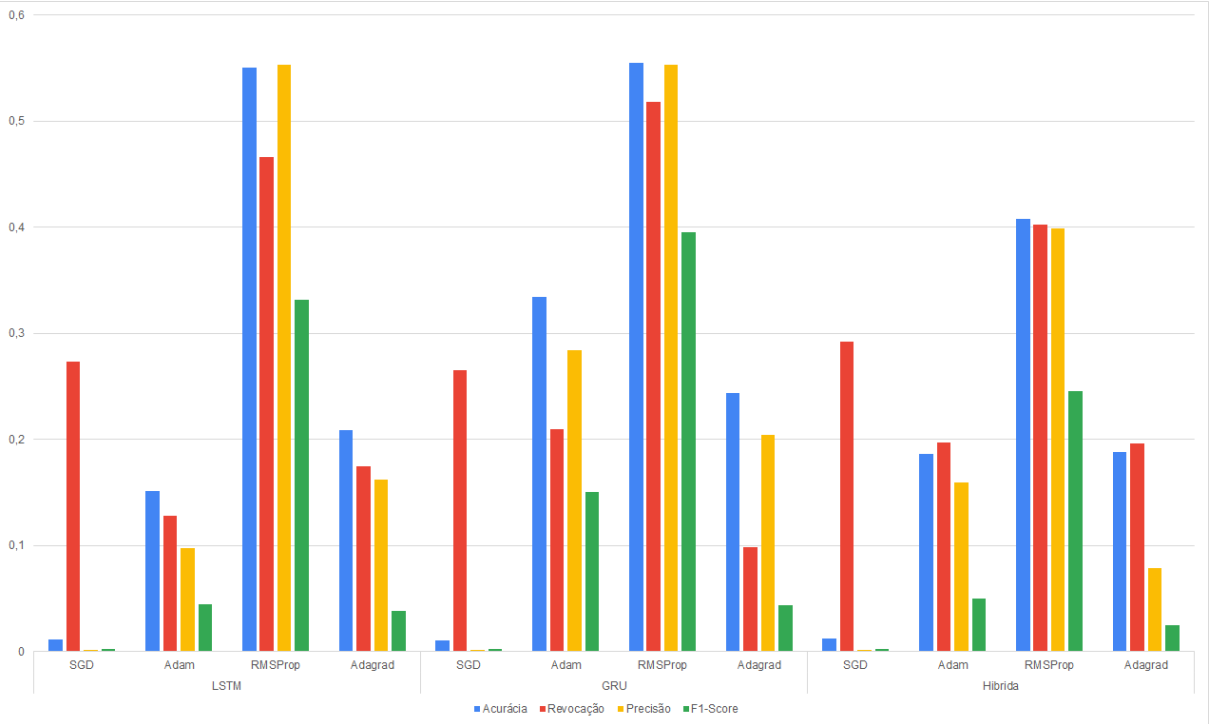
Agrupando os resultados a partir do otimizador, é possível analisar a performance das arquiteturas a partir otimizador utilizado no treinamento do modelo deste experimento, conforme apresentado na Tabela 10 e na Imagem 27.

Tabela 10 – Média geral das métricas por arquitetura de rede e otimizador utilizado

Arquitetura	Otimizador	*Accuracy*	*Recall*	*Precision*	F1-Score
LSTM	SGD	0,01158	0,27333	0,00141	0,00277
	Adam	0,15127	0,12786	0,09755	0,04428
	RMSProp	0,55057	0,46581	0,55344	0,33127
	Adagrad	0,20851	0,17431	0,16245	0,03876
GRU	SGD	0,01084	0,26497	0,00142	0,00276
	Adam	0,33471	0,20965	0,28363	0,15068
	RMSProp	0,55462	0,51780	0,55350	0,39510
	Adagrad	0,24343	0,09878	0,20437	0,04412
Híbrida	SGD	0,01269	0,29208	0,00135	0,00261
	Adam	0,18640	0,197493	0,15936286	0,05042538
	RMSProp	0,40807	0,40255	0,39889	0,24514
	Adagrad	0,18802	0,19637	0,07881	0,02516

Fonte: O autor.

Figura 27 – Gráfico da média geral das métricas por arquitetura de rede e otimizador utilizado.



Fonte: O autor

Os otimizadores foram os valores que mais apresentaram infuência nas métricas avaliadas. Os resultados com maiores valores estão relacionados ao uso do otimizador *RMSProp*, especialmente quando utilizado em conjunto com a arquitetura de GRU. Entretanto, os resultados relacionados a arquitetura de LSTM combinada ao uso do otimizador *RMSProp* mostrou resultados maiores quando utilizado um menor número de *batch*.

Os otimizadores *Adam* e *Adagrad* se mostraram fortemente ligados ao uso do número de épocas utilizado, onde quanto maior o número de épocas, maiores os valores das métricas avaliadas. Entretanto, quando comparados entre si, o *Adam* apresenta valores maiores das métricas avaliadas em relação ao *Adagrad*. Já os experimentos que utilizaram o otimizador *SGD* apresentaram os menores resultados independente da arquitetura utilizada.

4.5 Avaliação dos Resultados a partir do tamanho do *batch*

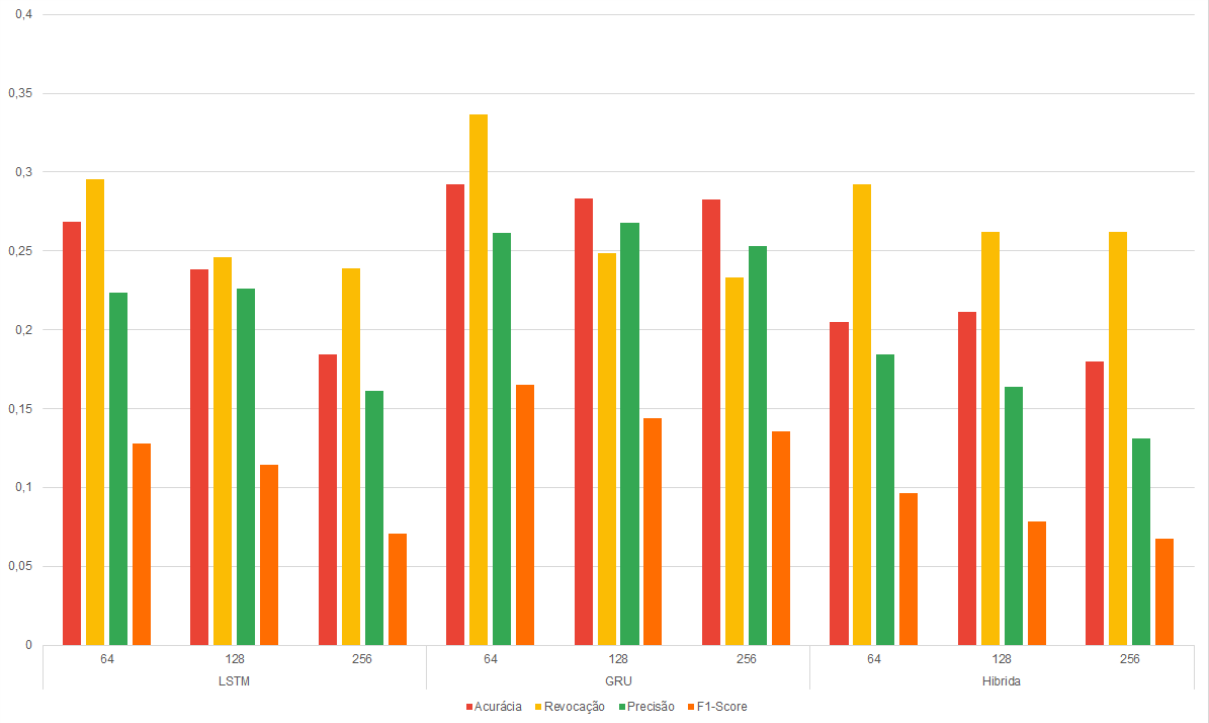
Agrupando os resultados a partir do tamanho do *batch*, é possível analisar a performance das arquiteturas a partir do tamanho do *batch* utilizado no treinamento do modelo deste experimento, conforme apresentado na Tabela 11 e na Imagem 28.

Tabela 11 – Média geral das métricas por arquitetura de rede e tamanho do *batch*

Arquitetura	***batch***	***Accuracy***	***Recall***	***Precision***	**F1-Score**
LSTM	64	0,26879	0,29560	0,2233	0,12796
	128	0,23852	0,24612	0,22624	0,11421
	256	0,18414	0,23926	0,16154	0,07065
GRU	64	0,29205	0,33642	0,26115	0,16489
	128	0,28314	0,24883	0,26781	0,14397
	256	0,28251	0,23316	0,25324	0,13564
Híbrida	64	0,20502	0,29228	0,18422	0,09613
	128	0,21142	0,26182	0,16354	0,07853
	256	0,17994	0,26227	0,13105	0,06785

Fonte: O autor.

Figura 28 – Gráfico da média geral das métricas por arquitetura de rede e tamanho do *batch*.



Fonte: O autor

A variação do tamanho do tamanho do *batch* apresentou comportamentos distintos nos resultados avaliados, de acordo com os valores utilizados como parâmetros de entrada. Ao utilizar a arquitetura de LSTM com o otimizador *RMSProp*, as métricas analisadas diminuiram conforme o aumento do tamanho do *batch*, enquanto o uso da arquitetura de GRU teve pouco impacto nos resultados utilizando os mesmos parâmetros. Porém, o uso de outros otimizadores a partir dos mesmos parâmetros, variando o tamanho do *batch*, teve o mesmo comportamento independente da arquitetura utilizada.

4.6 Comparação dos resultados com outros autores

Para avaliar a qualidade dos resultados obtidos, os valores das métricas analisadas foram comparadas com as mesmas métricas de trabalhos citados no Estado da Arte. Os demais trabalhos utilizam diferentes fontes de dados, e aplicam arquiteturas customizadas a partir das arquiteturas clássicas avaliadas neste trabalho.

O trabalho de Anil et al. (2018) apresenta uma arquitetura de uma rede híbrida que compõe a arquitetura de LSTM e GRU, tal qual a construída neste experimento. Apesar de Anil et al. (2018) terem utilizado apenas 10 épocas para treinar sua rede, e não terem mencionadoo uso de otimizadores, os valores de *Accuracy* variam entre

0,3328 e 0,4606, sendo inferior aos valores observados no treinamento da rede híbrida deste trabalho, que obteve o *Accuracy* de 0,9978 ao utilizar a função de ativação *softmax* combinada com o uso de 50 épocas para o treinamento, o otimizador *RMSprop* e o tamanho do *batch* como 64.

Já Ruocco, Skrede e Langseth (2017) treinaram redes com a arquitetura de GRU e uma solução adaptada da mesma arquitetura utilizando o tamanho do *batch* como 100 e variando o número de épocas entre 5, 10 e 20. Os resultados avaliaram apenas a métrica de *Recall*, obtendo os valores entre 0,5022 e 0,6537 na melhor combinação de atributos de uma de suas redes customizadas, enquanto a arquitetura GRU convencional obteve valores entre 0,3660 e 0,5118 com os mesmos parâmetros. A rede GRU treinada neste trabalho possui valores equivalentes na maior parte dos casos, sendo superiores quando utilizado o otimizador *RMSProp* ou o *Adam* combinado ao uso de 100 épocas durante o treinamento. Korotaev e Lyadova (2018) também utilizam da arquitetura de GRU em seu trabalho, porém não apresentam resultados para comparação.

Fu et al. (2018) utilizam uma rede LSTM bidirecional com a mesma base de dados utilizada neste experimento. Porém, os autores obtiveram apenas 10 mil sessões do arquivo de entrada de maneira randômica, utilizando o tamanho do *batch* como 32 e 100 épocas para treinamento, sem especificar o uso de um otimizador. Nesta configuração, os valores de *Precision* variam entre 0,1895 e 0,1165, os de *Recall* variam entre 0,1165 e 0,379, e os de *F1-Score* variam entre 0,036095 e 0,1165. Neste experimento, a arquitetura de LSTM obteve valores de *Precision* como 0,99712, *Recall* como 0,9813 e *F1-Score* como 0,98913 ao utilizar 20 épocas no treinamento combinado ao uso do otimizador de *RMSProp* e tamanho do *batch* como 64. Tais valores também são superiores aos apresentados no trabalho de Chu et al. (2017), que não citam o uso de otimizadores em sua arquitetura.

5 CONCLUSÃO

Este estudo constituiu uma avaliação de diferentes arquiteturas de Redes Neurais Recorrentes utilizando dados do Histórico de Navegação do usuário como insumo, a fim de proporcionar uma alternativa para Sistemas de Recomendação em situação de *Cold-Start Problem*. Para tal, foram abordados, neste trabalho, os principais conceitos envolvendo Sistemas de Recomendação, *Cold-Start Problem*, Inteligência Artificial, Aprendizado Profundo, Redes Neurais e Redes Neurais Recorrentes.

Uma revisão sistemática foi realizada para identificar os trabalhos relevantes sobre o tema abordado e para a extração de outras informações importantes para a pesquisa. Os trabalhos de Anil et al. (2018); Ruocco, Skrede e Langseth (2017); Kong, Li e Lv (2018); Korotaev e Lyadova (2018); Fu et al. (2018); e Chu et al. (2017) foram utilizados como base para a identificação das arquiteturas de Redes de Memória de Longo Prazo (LSTM), de Redes de Unidades Recorrentes de Bloqueio (GRU) e de Redes Híbridas como as mais frequentes entre as utilizadas no contexto de Redes Neurais Recorrentes com uso de dados sequenciais.

Entretanto, ao consolidar os resultados da revisão sistemática, não foi possível identificar comparações satisfatórias entre as arquiteturas utilizadas, uma vez que os autores não comparam o resultado das redes com as arquiteturas LSTM, GRU e Híbridas. Sendo assim, este trabalho propôs a avaliação dessas três arquiteturas através da construção de protótipos que utilizam a mesma base de dados e as mesmas métricas de avaliação. Para tal, as métricas de *Accuracy*, de *Precision*, de *Recall* e de *F1-Score* foram utilizadas, conforme presentes na maioria dos trabalhos obtidos através da revisão sistemática.

Foram separados 108 cenários a partir da combinação das variáveis utilizadas no modelo, conforme descrito na Tabela 6 e no Apêndice F, enquanto o resultado completo da execução de cada protótipo está disponível no Apêndice G. Um dos maiores desafios encontrados durante a execução do experimento foi garantir o funcionamento dos protótipos sem exceder o limite de memória da máquina utilizada dado o uso do *One-Hot Encoding*. Para tal, os dados de entrada tiveram que ser reduzidos iterativamente em proporções menores até atingir 10% de seu total.

De forma geral, os resultados apresentam melhor performance da arquitetura de Unidades Recorrentes de Bloqueio na maior parte dos cenários em comparação às outras duas arquiteturas. Enquanto a medida de *Recall* aparece com valores similares na maioria dos cenários para as três arquiteturas, sendo a medida de *Accuracy* e a de *Precision* são consideravelmente maiores no uso de GRU, resultando em um *F1-Score*

maior. Já o modelo Híbrido apresentou resultados inferiores às demais arquiteturas entre todas as métricas de avaliação.

Também foi possível identificar a performance das arquiteturas a partir das variáveis utilizadas no modelo. A função *softmax* se mostrou mais eficaz que as demais funções de ativação utilizadas no modelo, obtendo altos valores de *Accuracy* e de *Precision* quando utilizada, enquanto a função da tangente hiperbólica teve resultados expressivamente menores que as demais funções de ativação.

Da mesma forma, o uso do otimizador RMSProp garantiu resultados superiores aos demais otimizadores, enquanto o uso de SGD rendeu resultados inferiores aos demais obtidos. Já a variação do número de épocas no treinamento dos modelos apresentou uma melhoria conforme o crescimento da variável, enquanto a mudança do tamanho do batch teve pouca influência nos resultados.

Quando comparados aos resultados dos trabalhos revisados no Estado da Arte desta pesquisa, os resultados obtidos através da execução dos protótipos construídos foram superiores aos demais quando utilizadas combinações específicas de parâmetros. Foi possível observar que os trabalhos revisados no Estado da Arte não citaram o uso de otimizadores em seus modelos, enquanto a variação de otimizadores durante o treinamento das redes desta pesquisa mostrou uma grande diferença no aumento das métricas avaliadas.

Sendo assim, dentro do contexto deste experimento, o uso de Redes de Unidades Recorrentes de Bloqueio, combinadas ao uso de RMSProp como otimizador e o uso da *softmax* como função de ativação do modelo, é sugerido para o uso do histórico de navegação como alternativa para Sistemas de Recomendação em situação de Cold-Start Problem, em consequência de sua performance observada nos resultados se comparada às arquiteturas identificadas na revisão sistemática deste trabalho.

5.1 Trabalhos Futuros

Considerando a limitação realizada na base de entrada para evitar falha de execução do modelo por falta de memória da máquina, sugere-se a execução deste experimento em máquinas com memória física superior a 32 GB ou com execução a partir da GPU. Dessa forma, é sugerido o uso da base de entrada em sua totalidade para identificar alterações nos resultados.

Também se recomenda a modificação das variáveis utilizadas, podendo adicionar novos valores ou adicionar novas variáveis ao modelo com o propósito de aprimorá-lo. Além das variáveis estipuladas, outras arquiteturas de Redes Neurais Recorrentes podem ser testadas sob as mesmas condições do experimento deste trabalho com a finalidade de verificar sua performance quanto às arquiteturas utilizadas.

Em especial, o uso de outros otimizadores a partir dos valores padrões sugeridos pelos autores, uma vez que eles se mostraram com maior influência na qualidade dos resultados.

A base de entrada deste estudo foi escolhida por ser uma base pública com dados do histórico de navegação do usuário que se adequam ao modelo deste experimento. Outras bases de dados contendo o histórico de navegação do usuário podem ser testadas no modelo, variando o tamanho, o número de sequências ou o último estado da sessão do usuário. Diferentes bases de dados podem ser utilizadas para identificar a mudança dos resultados conforme o contexto da base de entrada.

Durante a execução dos protótipos, foi observado a convergência de alguns valores de *Accuracy* para 1 dada uma certa combinação de atributos. Sugere-se, também, a adição de uma camada de *Dropout* para evitar que os dados sofram de *overfit*.

REFERÊNCIAS

ACADEMY, D. S. **Deep Learning Book**. 2019. Disponível em: <http://www.deeplearningbook.com.br>.

AGGARWAL, C. C. **An Introduction to Recommender Systems**. Cham: Springer International Publishing, 2016. 1–28 p. ISBN 978-3-319-29659-3. Disponível em: <https://doi.org/10.1007/978-3-319-29659-3_1>.

AGUIAR, F. G. **Utilização de redes neurais artificiais para detecção de padrões de vazamento em dutos**. São Carlos: [s.n.], 2010.

AL-BASHIRI, H. et al. An improved memory-based collaborative filtering method based on the topsis technique. **PLOS ONE**, Public Library of Science, v. 13, n. 10, p. 1–26, 10 2018. Disponível em: <https://doi.org/10.1371/journal.pone.0204434>.

AL-MASHHADANY, Y. **Recurrent Neural Network with Human Simulator Based Virtual Reality**. [S.l.: s.n.], 2012. ISBN 978-953-51-0409-4.

ANIL, D. et al. Performance analysis of deep learning architectures for recommendation systems. p. 129–136, 12 2018.

ANáSTACIO, R. dos S. **Utilização de informações de redes sociais em algoritmos de recomendação baseados em filtragem colaborativa**. São Paulo: [s.n.], 2018.

BISHOP, C. M. **Pattern Recognition and Machine Learning (Information Science and Statistics)**. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN 0387310738.

BOBADILLA, J. et al. A collaborative filtering approach to mitigate the new user cold start problem. **Knowledge-Based Systems**, v. 26, p. 225 – 238, 2012. ISSN 0950-7051. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0950705111001882>.

BULLINARIA, J. A. Recurrent neural networks. **Neural Computation: Lecture12**, 2015.

CHO, K. et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation. **CoRR**, abs/1406.1078, 2014. Disponível em: <http://arxiv.org/abs/1406.1078>.

CHOLLET, F. et al. **Keras**. 2015. <https://keras.io>.

CHOU, P.-H. et al. Integrating web mining and neural network for personalized e-commerce automatic service. **Expert Systems with Applications**, v. 37, p. 2898–2910, 2010.

Chu, Y. et al. Short-term recommendation with recurrent neural networks. In: **2017 IEEE International Conference on Mechatronics and Automation (ICMA)**. [S.l.: s.n.], 2017. p. 927–932. ISSN 2152-744X.

DENG, S.; HUANG, L.; XU, G. Social network-based service recommendation with trust enhancement. **Expert Systems with Applications**, v. 41, n. 18, p. 8075—8084, 2014.

DO, M.-P. T.; NGUYEN, D. V.; NGUYEN, A. N. of L. Model-based approach for collaborative filtering. **The 6th International Conference on Information Technology for Education**, 08 2010.

DONGES, N. **Gradient Descent in a Nutshell**. 2018. Disponível em: <https://towardsdatascience.com/gradient-descent-in-a-nutshell-eaf8c18212f0>.

DUCHI, J. C.; HAZAN, E.; SINGER, Y. Adaptive subgradient methods for online learning and stochastic optimization. **Journal of Machine Learning Research**, v. 12, p. 2121–2159, 07 2011.

EKSTRAND, M. D.; RIEDL, J. T.; KONSTAN, J. A. Collaborative filtering recommender systems. **Found. Trends Hum.-Comput. Interact.**, Now Publishers Inc., Hanover, MA, USA, v. 4, n. 2, p. 81–173, fev. 2011. ISSN 1551-3955. Disponível em: <http://dx.doi.org/10.1561/1100000009>.

ELMAN, J. L. Finding structure in time. **Cognitive Science**, v. 14, n. 2, p. 179–211, 1990.

FOGEL, D.; LIU, D.; KELLER, J. M. **Recurrent Neural Networks**. [S.l.: s.n.], 2016. 77-100 p.

FU, H. et al. **Sequence-Based Recommendation with Bidirectional LSTM Network: 19th Pacific-Rim Conference on Multimedia, Hefei, China, September 21-22, 2018, Proceedings, Part III**. [S.l.: s.n.], 2018. 428-438 p. ISBN 978-3-030-00763-8.

FU, X.; BUDZIK, J.; HAMMOND, K. J. Mining navigation history for recommendation. ACM, New York, NY, USA, p. 106–112, 2000. Disponível em: <http://doi.acm.org/10.1145/325737.325796>.

GOLDBERG, Y.; HIRST, G. **Neural Network Methods in Natural Language Processing**. [S.l.]: Morgan and Claypool Publishers, 2017. ISBN 1627052984, 9781627052986.

GOMEZ-URIBE, C. A.; HUNT, N. The netflix recommender system: Algorithms, business value. **ACM Trans. Manage. Inf. Syst.**, v. 6, n. 4, 2015.

GOODFELLOW, Y. B. I.; COURVILLE, A. Deep learning. Book in preparation for MIT Press. 2016. Disponível em: <http://www.deeplearningbook.org>.

HAYKIN, S. S. **Neural networks and learning machines**. Third. Upper Saddle River, NJ: Pearson Education, 2009.

HEINZ, S.; BRACHER, C.; VOLLGRAF, R. An lstm-based dynamic customer model for fashion recommendation. **CoRR**, abs/1708.07347, 2017. Disponível em: <http://arxiv.org/abs/1708.07347>.

HIDASI, B. et al. Session-based recommendations with recurrent neural networks. 11 2015.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. **Neural Comput.**, MIT Press, Cambridge, MA, USA, v. 9, n. 8, p. 1735–1780, nov. 1997. ISSN 0899-7667. Disponível em: <http://dx.doi.org/10.1162/neco.1997.9.8.1735>.

HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. **Proceedings of the National Academy of Sciences**, v. 79, n. 8, p. 2554–2558, 1982. Disponível em: <https://www.pnas.org/content/79/8/2554>.

HOU, L.; PAN, X.; LIU, K. Balancing the popularity bias of object similarities for personalised recommendation. **The European Physical Journal B**, v. 91, n. 3, p. 47, Mar 2018. ISSN 1434-6036. Disponível em: <https://doi.org/10.1140/epjb/e2018-80374-8>.

JORDAN, M. I. Serial order: A parallel distributed processing approach. North-Holland, v. 121, p. 471 – 495, 1997. ISSN 0166-4115. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0166411597801112>.

KAGGLE. **LiveStreaming dataset**. Disponível em: <https://www.kaggle.com/hailinfu/livestreaming/version/1>.

KHATWANI, S.; CHANDAK, M. Building personalized and non personalized recommendation systems. **International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)**, 2016.

KONG, F.; LI, J.; LV, Z. Construction of intelligent traffic information recommendation system based on long short-term memory. **Journal of Computational Science**, v. 26, p. 78 – 86, 2018. ISSN 1877-7503. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877750317314084>.

KOREN, Y.; BELL, R. Advances in collaborative filtering. Springer Science+Business Media, p. 107–144, 2011.

KOROTAEV, A.; LYADOVA, L. Method for the development of recommendation systems, customizable to domains, with deep gru network. p. 231–236, 01 2018.

KUMAR, R. et al. Recommendation systems: a probabilistic analysis. **Proceedings 39th Annual Symposium on Foundations of Computer Science**, 1998.

LI, C. et al. Capturing temporal dynamics of users' preferences from purchase history big data for recommendation system. p. 5372–5374, 12 2018.

LIKA, B.; KOLOMVATSOS, K.; HADJIEFTHYMIADES, S. Facing the cold start problem in recommender systems. **Expert Syst. Appl.**, Pergamon Press, Inc., Tarrytown, NY, USA, v. 41, n. 4, p. 2065–2073, 2014. ISSN 0957-4174. Disponível em: <http://dx.doi.org/10.1016/j.eswa.2013.09.005>.

LOPS, P.; GEMMIS, M. de; SEMERARO, G. Content-based recommender systems: State of the art and trends. Springer Science+Business Media, p. 73–100, 2011.

MADHUKAR, M. Challenges and limitation in recommender systems. **International Journal of Latest Trends in Engineering and Technology (IJLTET)**, 2014.

MASTHOFF, J. Group recommender systems: Combining individual models. Springer Science+Business Media, p. 677–702, 2011.

MIKAMI, A. **Long Short-Term Memory Recurrent Neural Network Architectures for Generating Music and Japanese Lyrics**. Tese (Doutorado) — Ph. D., Computer Science Department, Boston College, 2016.

MINSKY, M.; PAPERT, S. **Perceptrons - an introduction to computational geometry**. [S.l.]: MIT Press, 1987.

MITCHELL, T. M. **Machine Learning**. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN 0070428077, 9780070428072.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos sobre aprendizado de máquina. **Sistemas Inteligentes-Fundamentos e Aplicações**, v. 1, n. 1, p. 32, 2003.

NADIMI-SHAHRAKI, M.-H.; BAHADORPOUR, M. Cold-start problem in collaborative recommender systems: Efficient methods based on ask-to-rate technique. **Journal of Computing and Information Technology**, p. 105/113, 2014.

NETCRAFT. **June 2018 Web Server Survey**. 2018. Disponível em: <https://news.netcraft.com/archives/2018/06/13/june-2018-web-server-survey.html>.

PANDEY, A. K.; RAJPOOT, D. S. Resolving cold start problem in recommendation system using demographic approach. **2016 International Conference on Signal Processing and Communication (ICSC)**, 2016.

PERDANAPUTRA, A. Greedy implementation of neural networks gradient descent. **Strategi Algoritma**, 2017.

POWERS, D.; AILAB. Evaluation: From precision, recall and f-measure to roc, informedness, markedness and correlation. **J. Mach. Learn. Technol**, v. 2, p. 2229–3981, 01 2011.

RICCI, F. et al. **Recommender Systems Handbook**. [S.l.]: Springer Science+Business Media, 2011.

ROBERT, S. The pareto principle: Its use and abuse. MCB UP Ltd, v. 1, n. 2, p. 37–40, Jan 1987. ISSN 0887-6045. Disponível em: <https://doi.org/10.1108/eb024706>.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, p. 65–386, 1958.

RUDER, S. An overview of gradient descent optimization algorithms. **CoRR**, abs/1609.04747, 2016. Disponível em: <http://arxiv.org/abs/1609.04747>.

RUOCCO, M.; SKREDE, O. S. L.; LANGSETH, H. Inter-session modeling for session-based recommendation. **CoRR**, abs/1706.07506, 2017. Disponível em: <http://arxiv.org/abs/1706.07506>.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3rd. ed. Upper Saddle River, NJ, USA: Prentice Hall Press, 2009. ISBN 0136042597, 9780136042594.

SHAH, K. et al. Recommender systems: An overview of different approaches to recommendations. **International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)**, 2017.

SHANI, G.; GUNAWARDANA, A. Evaluating recommendation systems. Springer Science+Business Media, p. 257–297, 2011.

Sá, D. F. S. de. **MODELOS BASEADOS EM REDES NEURAIS ARTIFICIAIS COM APLICAÇÃO EM CONTROLE INDIRETO DE TEMPERATURA**. Dissertação (Mestrado), 2015. Engenharia. Disponível em: <http://tedebc.ufma.br:8080/jspui/handle/tede/295>.

TAN, Y. K.; XU, X.; LIU, Y. Improved recurrent neural networks for session-based recommendations. ACM, p. 17–22, 2016. Disponível em: <http://doi.acm.org/10.1145/2988450.2988452>.

TRUPTI, K. Recurrent neural network and its various architecture types. **International Journal of Research and Scientific Innovation**, v. 5, n. 3, p. 124–129, 2018.

WEI, J. et al. Collaborative filtering and deep learning based recommendation system for cold start items. **Expert Systems with Applications**, v. 69, p. 29 – 39, 2017. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417416305309>.

Willmott, C.; Matsuura, K. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. **Climate Research**, v. 30, p. 79–82, 2005.

YAO, L. et al. Unified collaborative and content-based web service recommendation. **IEEE TRANSACTIONS ON SERVICES COMPUTING**, v. 8, p. 453–466, 2015.

Yuan, W.; Yang, Y.; Bao, X. Parallel dnns for users and items modeling and recommendation using comments. p. 1–6, Aug 2017.

ZHANG, R. sheng et al. Collaborative filtering for recommender systems. **2014 Second International Conference on Advanced Cloud and Big Data**, 2014.

ZHOU, T. et al. Jump: a jointly predictor for user click and dwell time. p. 3704–3710, 07 2018.

APÊNDICE A – PROTOCOLO DE PESQUISA PARA ESTADO DA ARTE

A.1 Objetivo

Realizar uma revisão na literatura para identificar trabalhos que utilizem redes neurais recorrentes em sistemas de recomendação

A.2 Questão de Pesquisa

- Como utilizar dados sequenciais em redes neurais recorrentes para recomendação de itens em sistemas de recomendação;

A.3 Seleção de Fontes

A.3.1 Critério de Definição de Fontes

Mecanismos de busca de maior relevância para a área científica com possibilidade de acesso via web e com possibilidade de uso de strings de busca para filtros avançados.

A.3.2 Listagem das Fontes

As seguintes fontes foram selecionadas devido seu grau de relevância na área científica e por contar com bases de trabalhos ligadas à área de Engenharia de Software

- IEEE Digital Library (http://ieeexplore.ieee.org/Xplore/);
- ACM Digital Library (http://portal.acm.org);
- SCOPUS Digital Library (http://www.scopus.com);

A.3.3 Palavras-Chave

"Redes Neurais Recorrentes", ou sua versão na língua inglesa "Recurrent Neural Networks", e "Sistemas de Recomendação", ou sua versão na língua inglesa "Recommendation Systems". Operadores lógicos OR e AND também foram utilizados para combinação entre as palavras chaves.

A.4 Seleção de Fontes

Os seguintes critérios de inclusão ou exclusão foram definidos para escolher os trabalhos a serem aceitos por esta revisão.

A.4.1 Critérios de Inclusão

- I-1: Trabalhos publicados entre 2013 em diante;
- I-2: Trabalhos publicados em inglês ou português;
- I-3: Trabalhos disponíveis na web em formato eletrônico;
- I-4: Trabalhos de natureza experimental ou revisões sistemáticas de literatura;
- I-5: Estudos primários ou secundários;

Para a seleção inicial dos estudos, foi realizada uma leitura dos títulos e resumos para determinar a utilidade do trabalho.

A.4.2 Critérios de Exclusão

- E-1: Estudos repetidos;
- E-2: Trabalhos que o resumo não atenda ao objetivo especificado;
- E-3: Versão completa não disponibilizada para ser acessada em formato digital;
- E-4: Não aborda modelos arquiteturais de redes neurais recorrentes;

A.5 Estratégia de Seleção de Dados

Através de buscas testes por meio das palavras-chaves apontadas, foram criadas strings de busca específicas para cada fonte selecionada. Buscando um resultado mais assertivo, as buscas foram feitas em cima das palavras-chaves dos trabalhos, extraindo os seguintes dados dos resultados: título, resumo, palavras-chaves, nome dos autores, fonte e ano de publicação.

As strings de busca consideraram a derivação das palavras-chaves em sinônimos, abreviação ou pluralidade para garantir a equivalência nas bases de busca. Também foram consideradas strings de busca nas demais línguas listadas neste protocolo, porém as palavras de natureza inglesa foram mantidas na língua original.

- Strings na Língua Portuguesa: ((((“sistemas"OR "sistema”) AND (“recomendação” OR "recomendações")) OR ("cold-start"OR "cold start")) AND (("redes"OR "rede") AND ("recorrente"OR "recorrentes")));
- Strings na Língua Inglesa: ((((“systems"OR "system”) AND (“recommendation” OR "recommendations")) OR ("cold-start"OR "cold start")) AND (("network"OR "networks") AND ("recurrent")) AND ("RNN"));

Por fim, um resumo foi realizado após a leitura integral dos trabalhos incluídos, destacando as informações relevantes de cada trabalho resultante.

APÊNDICE B – EXECUÇÃO DO PROTOCOLO DE PESQUISA PARA ESTADO DA ARTE

Os demais tópicos desta seção apresentam a execução e os resultados das buscas realizadas nas demais fontes apontadas no procolo de pesquisa definido no Apêndice A

B.1 IEEE Explore

B.1.1 Formulário

- Fonte: IEEE Xplore Digital Library
- Data de Busca: 20/02/2019
- Strings Utilizadas: Língua Portuguesa: (("Abstract":"rede neural recorrente"OR "Abstract":"rede recorrente"OR "Abstract":"redes neurais recorrentes") AND ("Abstract":"sistemas de recomendação"OR "Abstract":"sistema de recomendação") OR "Abstract":"cold-start problem") OR "Abstract":"cold start problem")); Língua Inglesa: (("Abstract":"recurrent neural network"OR "Abstract":"recurrent network"OR "Abstract":"RNN"OR "Abstract":"recurrent neural networks") AND ("Abstract":"recommendation systems"OR "Abstract":"recommendation system"OR "Abstract":"cold-start problem"OR "Abstract":"cold start problem")).
- Campos Pesquisados: Palavras-chave (Keywords)
- Período Considerado: 2012 à Março de 2019
- Filtros Utilizados: “Metadata only”

B.1.2 Lista de resultados

- IEE-1 - D. Anil et al., "Performance Analysis of Deep Learning Architectures for Recommendation Systems", 2018.
- IEE-2 - Z. Qu, et al., "Converged Recommendation System Based on RNN and BP Neural Networks", 2018.
- IEE-3 - M. Jiang, Z. Yang e C. Zhao, "What to play next? A RNN-based music recommendation system", 2017.
- IEE-4 - T. Saito e Y. Watanobe, "Learning Path Recommender System based on Recurrent Neural Network", 2018.

- IEE-5 - R. Lopes Rosa et al., "A Knowledge-Based Recommendation System that includes Sentiment Analysis and Deep Learning".
- IEE-6 - Q. Lin et al., "Heterogeneous Knowledge-Based Attentive Neural Networks for Short-Term Music Recommendations", 2018.
- IEE-7 - G. Preethi et al., "Application of Deep Learning to Sentiment Analysis for recommender system on cloud", 2017.
- IEE-8 - C. Li et al., "Capturing Temporal Dynamics of Users' Preferences from Purchase History Big Data for Recommendation System", 2018.
- IEE-9 - T. Yang, C. G. Brinton e C. Joe-Wong, "Predicting Learner Interactions in Social Learning Networks", 2018
- IEE-10 - S. Li et al., "A Method of Emotional Analysis of Movie Based on Convolution Neural Network and Bi-directional LSTM RNN", 2017.
- IEE-11 - Q. Cui et al., "MV-RNN: A Multi-View Recurrent Neural Network for Sequential Recommendation".
- IEE-12 - W. Yuan, Y. Yang e X. Bao, "Parallel DNNs for users and items modeling and recommendation using comments", 2017.
- IEE-13 - Z. Zhao et al., "A Memory-Efficient Approach to the Scalability of Recommender System With Hit Improvement", 2018.

Tabela 12 – Lista de inclusão e exclusão dos trabalhos

Item	Critérios de Inclusão	Critérios de Exclusão	Status
IEE-1	(I-1), (I-2), (I-3) (I-4), (I-5)		Aceito
IEE-2	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
IEE-3	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
IEE-4	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
IEE-5	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
IEE-6	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
IEE-7	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
IEE-8	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
IEE-9	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
IEE-10	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
IEE-11	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
IEE-12	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
IEE-13	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado

Fonte: O autor.

B.2 ACM Digital Library

B.2.1 Formulário

- Fonte: ACM Digital Library
- Data de Busca: 20/02/2019
- Strings Utilizadas: Língua Portuguesa: recordAbstract:((+rede +neural +recorrente) OR (+rede +recorrente) OR (+redes +neurais +recorrentes)) AND ((+sistemas +de +recomendação) OR (+sistema +de +recomendação) OR (+cold +start) OR (+cold-start)) ; Língua Inglesa: recordAbstract:((+recurrent +neural +network) OR (+recurrent +network) OR (+RNN) OR (+recurrent +neural +networks)) AND ((+recommendation +systems) OR (+recommendation +system) OR (+cold +start) OR (+cold-start)) .
- Período Considerado: 2012 à Março de 2019
- Filtros Utilizados: “owners.owner=HOSTED"

B.2.2 Lista de resultados

- ACM-1 - Halder K., Poddar L. e Kan M, 2018, "Cold Start Thread Recommendation as Extreme Multi-label Classification"
- ACM-2 - Bansal T., Belanger D. e McCallum A, 2016, "Ask the GRU: Multi-task Learning for Deep Text Recommendations"
- ACM-3 - Ruocco M., Skrede O. e Langseth H, 2017, "Inter-Session Modeling for Session-Based Recommendation"
- ACM-4 - Moreira G., Ferreira F. e Cunha A., 2018, "News Session-Based Recommendations using Deep Neural Networks"
- ACM-5 - Kim J. et al., 2018, "Towards Seed-Free Music Playlist Generation: Enhancing Collaborative Filtering with Playlist Title Information"
- ACM-6 - Khattar D. et al., 2018, "HRAM: A Hybrid Recurrent Attention Machine for News Recommendation"
- ACM-7 - Manotumruksa J., Macdonald C. e Ounis I., 2017, "A Deep Recurrent Collaborative Filtering Framework for Venue Recommendation"

- ACM-8 - Sonie O., Sarkar S. e Kumar S., 2018, "Concept to code: learning distributed representation of heterogeneous sources for recommendation"
- ACM-9 - Xu X. et al., 2018, "Hulu video recommendation: from relevance to reasoning"
- ACM-10 - Suglia A. et al., 2017, "A Deep Architecture for Content-based Recommendations Exploiting Recurrent Neural Networks"
- ACM-11 - Bharadhwaj H., 2019, "Explainable recommender system that maximizes exploration"
- ACM-12 - Park K., Lee J. e Choi J., 2017, "Deep Neural Networks for News Recommendations"
- ACM-13 - Bharadhwaj H., Park H. e Lim B., 2018, "RecGAN: recurrent generative adversarial networks for recommendation systems"
- ACM-14 - Quadrana M. et al., 2017, "Personalizing Session-based Recommendations with Hierarchical Recurrent Neural Networks"
- ACM-15 - Tan Y., Xu X. e Liu Y., 2016, "Improved Recurrent Neural Networks for Session-based Recommendations"
- ACM-16 - Twardowski B., 2016, "Modelling Contextual Information in Session-Aware Recommender Systems with Neural Networks"
- ACM-17 - Bersini R., 2017, "Long and Short-Term Recommendations with Recurrent Neural Networks"
- ACM-18 - Baral R. et al., 2018, "CLoSe: Contextualized Location Sequence Recommender"
- ACM-19 - Donkers T., Loepp B. e Ziegler J., 2017, "Sequential User-based Recurrent Neural Network Recommendations"
- ACM-20 - Wang S., 2017, "Audience Activity Recommendation Using Stacked-LSTM Based Sequence Learning"
- ACM-21 - Vassøy B. et al., 2019, "Time is of the Essence: A Joint Hierarchical RNN and Point Process Model for Time and Item Predictions"
- ACM-22 - Vasile E., 2017, "Contextual Sequence Modeling for Recommendation with Recurrent Neural Networks"
- ACM-23 - Hidasi B. et al., 2016, "Parallel Recurrent Neural Network Architectures for Feature-rich Session-based Recommendations"

- ACM-24 - A H., Mohamed Hassan, Giuseppe Sansonetti, Fabio Gasparetti, and Alessandro Micarelli, "2018"
- ACM-25 - Yu F. et al., 2016, "A Dynamic Recurrent Model for Next Basket Recommendation"
- ACM-26 - Huang J. et al., 2018, "Improving Sequential Recommendation with Knowledge-Enhanced Memory Networks"
- ACM-27 - Almahairi A. et al., 2015, "Learning Distributed Representations from Reviews for Collaborative Filtering"
- ACM-28 - Ludewig D., 2017, "When Recurrent Neural Networks meet the Neighborhood for Session-Based Recommendation"
- ACM-29 - Lian J. et al., 2018, "xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems"
- ACM-30 - Sun F. et al., 2017, "A recommendation algorithm of we-media articles"
- ACM-31 - Beutel A. et al., 2018, "Latent Cross: Making Use of Context in Recurrent Recommender Systems"
- ACM-32 - Phuong T., 2017, "3D Convolutional Networks for Session-based Recommendation with Content Features"
- ACM-33 - Okura S. et al., 2017, "Embedding-based News Recommendation for Millions of Users"
- ACM-34 - Wang Q. et al., 2018, "Neural Memory Streaming Recommender Networks with Adversarial Training"
- ACM-35 - Caverlee J., 2019, "Recurrent Recommendation with Local Coherence"
- ACM-36 - Lin X. et al., 2018, "K-plet Recurrent Neural Networks for Sequential Recommendation"
- ACM-37 - Hassan H., 2017, "Personalized Research Paper Recommendation using Deep Learning"
- ACM-38 - Chen X. et al., 2018, "Sequential Recommendation with User Memory Networks"
- ACM-39 - Dai H. et al., 2016, "Recurrent Coevolutionary Latent Feature Processes for Continuous-Time Recommendation"

- ACM-40 - Monti D. et al., 2018, "An Ensemble Approach of Recurrent Neural Networks using Pre-Trained Embeddings for Playlist Completion"
- ACM-41 - Liu K. et al., 2016, "Temporal learning and sequence modeling for a job recommender system"
- ACM-42 - Wu C. et al., 2017, "Recurrent Recommender Networks"
- ACM-43 - P S., Chatzis, Panayiotis Christodoulou, and Andreas S, "Andreou"
- ACM-44 - Wang L. et al., 2018, "Supervised Reinforcement Learning with Recurrent Neural Network for Dynamic Treatment Recommendation"
- ACM-45 - Lee H. et al., 2016, "Quote Recommendation in Dialogue using Deep Neural Network"
- ACM-46 - Musto C. et al., 2018, "Deep Content-based Recommender Systems Exploiting Recurrent Neural Networks and Linked Open Data"
- ACM-47 - Yang L. et al., 2018, "Characterizing User Skills from Application Usage Traces with Hierarchical Attention Recurrent Networks"
- ACM-48 - Greenstein-Messica A., Rokach L. e Friedman M., 2017, "Session-Based Recommendations Using Item Embedding"
- ACM-49 - Fang T., 2017, "Neural Citation Network for Context-Aware Citation Recommendation"
- ACM-50 - Lin A., 2018, "The Evolution of Content Analysis for Personalized Recommendations at Twitter"
- ACM-51 - Sun Z. et al., 2018, "Recurrent knowledge graph embedding for effective recommendation"
- ACM-52 - Ku W., 2017, "Recommendation-based Smart Indoor Navigation: Poster Abstract"
- ACM-53 - Sun P., Wu L. e Wang M., 2018, "Attentive Recurrent Social Recommendation"
- ACM-54 - Gao J., Zhang T. e Xu C., 2017, "A Unified Personalized Video Recommendation via Dynamic Recurrent Neural Networks"
- ACM-55 - Ping Q., 2018, "Video recommendation using crowdsourced time-sync comments"

- ACM-56 - Jiang W., Pardos Z. e Wei Q., 2019, "Goal-based Course Recommendation"
- ACM-57 - Karatzoglou B., 2018, "Recurrent Neural Networks with Top-k Gains for Session-based Recommendations"
- ACM-58 - Sachdeva N. et al., 2019, "Sequential Variational Autoencoders for Collaborative Filtering"
- ACM-59 - Christakopoulou K. et al., 2018, "QandR: A Two-Stage Approach toward Interactive Recommendation"
- ACM-60 - Yang C. et al., 2017, "A Neural Network Approach to Jointly Modeling Social Networks and Mobile Trajectories"
- ACM-61 - Bai T. et al., 2018, "An Attribute-aware Neural Attentive Model for Next Basket Recommendation"
- ACM-62 - Song W. et al., 2019, "Session-Based Social Recommendation via Dynamic Graph Attention Networks"
- ACM-63 - Pardos S., 2017, "Personalized Behavior Recommendation: A Case Study of Applicability to 13 Courses on edX"
- ACM-64 - Manotumruksa J., Macdonald C. e Ounis I., 2018, "A Contextual Attention Recurrent Architecture for Context-Aware Venue Recommendation"
- ACM-65 - Zhao S. et al., 2017, "Geo-Teaser: Geo-Temporal Sequential Embedding Rank for Point-of-interest Recommendation"
- ACM-66 - Yin Z., Chang K. e Zhang R., 2017, "DeepProbe: Information Directed Sequence Understanding and Chatbot Design via Recurrent Neural Networks"
- ACM-67 - Li P. et al., 2017, "Neural Rating Regression with Abstractive Tips Generation for Recommendation"
- ACM-68 - A Z., Pardos, Steven Tang, Daniel Davis, and Christopher Vu Le, "2017"
- ACM-69 - Liu Q. et al., 2018, "STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation"
- ACM-70 - Kousias K. et al., 2018, "HINDSIGHT: an R-based framework towards long short term memory (LSTM) optimization"
- ACM-71 - Zhang S., 2017, "Collaborative Sequence Prediction for Sequential Recommender"

- ACM-72 - Atassi I., Azami E. e Sadiq A., 2018, "New GRU from convolutional neural network and gated recurrent unit"
- ACM-73 - Park K., Cha M. e Rhim E., 2018, "Positivity Bias in Customer Satisfaction Ratings"
- ACM-74 - Gao Q. et al., 2018, "Trajectory-based social circle inference"
- ACM-75 - Feng C. et al., 2018, "Attentive Encoder-based Extractive Text Summarization"
- ACM-76 - Hengyang Z. et al., 2015, "In Proceedings of the First International Conference on Data Science, E-learning and Information Systems (DATA '18)"
- ACM-77 - Pei W. et al., 2017, "Interacting Attention-gated Recurrent Networks for Recommendation"
- ACM-78 - Ding W. et al., 2016, "Audio and face video emotion recognition in the wild using deep neural networks and small datasets"
- ACM-79 - Ko S., Son J. e Jung H., 2018, "Sign language recognition with recurrent neural network using human keypoint detection"
- ACM-80 - Katumba A., Bienstman P. e Dambre J., 2015, "Photonic reservoir computing approaches to nanoscale computation"
- ACM-81 - Knottenbelt J., 2015, "Neural NILM: Deep Neural Networks Applied to Energy Disaggregation"
- ACM-82 - Wu X. et al., 2019, "Neural Tensor Factorization for Temporal Interaction Learning"
- ACM-83 - Chen C. et al., 2018, "Fast Distributed Deep Learning via Worker-adaptive Batch Sizing"
- ACM-84 - Sadri A. e D F., Salim, Yongli Ren, Wei Shao, John C, "Krumm, and Cecilia Mascolo"
- ACM-85 - Soleymani M. et al., Caro, Florian Eyben, Konstantin Markov, Björn W, "Schuller, Remco Veltkamp, Felix Weninger, and Frans Wiering"
- ACM-86 - Chakraborttii C., Sinha V. e Litz H., 2018, "SSD QoS Improvements through Machine Learning"
- ACM-87 - Chiba D., 2017, "A Neural Language Model for Query Auto-Completion"
- ACM-88 - Islam Y., 2018, "Question-Question Similarity in Online Forums"

- ACM-89 - López B. et al., 2018, "Knowledge representation and machine learning on wearable sensor data: a study on gait monitoring"
- ACM-90 - Assem H., Buda T. e O'sullivan D., 2017, "RCMC: Recognizing Crowd-Mobility Patterns in Cities Based on Location Based Social Networks Data"
- ACM-91 - Chen C. et al., 2018, "Predictive Analysis by Leveraging Temporal User Behavior and User Embeddings"
- ACM-92 - Meijer E., 2018, "Behind every great deep learning framework is an even greater programming languages concept (keynote)"
- ACM-93 - Cheng J. e A L., Adamic, Jon M, "Kleinberg, and Jure Leskovec"
- ACM-94 - Zhang T., 2017, "Deep Learning for Intelligent Video Analysis"
- ACM-95 - Costa R., 2011, "Using feedback in a regulatory network computational device"
- ACM-96 - Mertikopoulos P., Papadimitriou C. e Piliouras G., 2018, "Cycles in adversarial regularized learning"
- ACM-97 - Costa R., 2013, "Genetic programming with genetic regulatory networks: genetic programming"
- ACM-98 - Hasenbein D., 2003, "Weak instability in stochastic and fluid queueing networks"
- ACM-99 - Shamma G., 2014, "Optimization despite chaos: convex relaxations to complex limit sets via Poincaré; recurrence"
- ACM-100 - Barbera M. et al., 2013, "Signals from the crowd: uncovering social relationships through smartphone probes"
- ACM-101 - Huggett M., 2007, "Information-behaviour modeling with external cues"

Tabela 13 – Lista de inclusão e exclusão dos trabalhos

Item	Critérios de Inclusão	Critérios de Exclusão	Status
ACM-1	(I-1), (I-2), (I-3) (I-4), (I-5)		Aceito
ACM-2	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
ACM-3	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
ACM-4	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-4)	Rejeitado
ACM-5	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-4)	Rejeitado
ACM-6	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-4)	Rejeitado

ACM-7	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-8	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-4)	Rejeitado
ACM-9	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
ACM-10	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-3)	Rejeitado
ACM-11	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-12	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
ACM-13	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-14	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
ACM-15	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
ACM-16	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-4)	Rejeitado
ACM-17	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
ACM-18	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-19	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
ACM-20	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-21	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-22	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-23	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-24	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-25	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-26	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-27	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-28	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-4)	Rejeitado
ACM-29	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-30	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-31	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-32	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-33	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-34	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-35	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-36	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-37	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-4)	Rejeitado
ACM-38	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-39	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-40	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-41	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-42	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
ACM-43	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-44	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-45	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado

ACM-46	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-47	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-48	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-49	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-50	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-51	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-52	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-53	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-54	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-55	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-56	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-57	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-58	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-59	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-60	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-61	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-62	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-63	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-64	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-65	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-66	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-67	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-68	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-69	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-70	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-71	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-72	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-73	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-74	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-75	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-76	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-77	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-78	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-79	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-80	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-81	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-82	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-83	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-84	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado

ACM-85	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-86	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-87	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-88	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-89	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-90	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-91	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-92	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-93	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-94	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-95	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-96	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-97	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-98	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-99	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-100	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
ACM-101	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado

Fonte: O autor.

B.3 Scopus

B.3.1 Formulário

- Fonte: Scopus
- Data de Busca: 22/02/2019
- Strings Utilizadas: Língua Portuguesa: TITLE-ABS-KEY (("redes neurais recorrente"OR "rede recorrente"OR "rnn"OR "rede neural recorrente") AND ("sistemas de recomendação"OR "sistema de recomendação"OR "cold start"OR "cold-start")) ; Língua Inglesa: TITLE-ABS-KEY (("recurrent neural network"OR "recurrent network"OR "rnn"OR "recurrent neural network") AND ("recommendation systems"OR "recommendation system"OR "cold start"OR "cold-start")) .
- Período Considerado: 2012 à Março de 2019

B.3.2 Lista de resultados

- SCO-1: Gao X., Chen J. e Huai N., 2019, "Meta-Circuit machine: Inferencing human collaborative relationships in heterogeneous information networks"

- SCO-2: Jiang W., Pardos A., Wei Q., 2019, "Goal-based course recommendation"
- SCO-3: Anil D. et al., 2019, "Performance analysis of deep learning architectures for recommendation systems"
- SCO-4: Li C. et al., 2019, "Capturing Temporal Dynamics of Users' Preferences from Purchase History Big Data for Recommendation System"
- SCO-5: Huang R. et al., 2018, "An attention-based recommender system to predict contextual intent based on choice histories across and within sessions"
- SCO-6: Saito T. e Watanobe, Y., 2018, "Learning Path Recommender System based on Recurrent Neural Network"
- SCO-7: Khattar D. et al., 2018, "HRAM: A hybrid recurrent attention machine for news recommendation"
- SCO-8: Gui Y. e Xu Z., 2018, "Training Recurrent Neural Network on Distributed Representation Space for Session-based Recommendation"
- SCO-9: Yang T., Brinton C. e Joe-Wong C., 2018, "Predicting Learner Interactions in Social Learning Networks"
- SCO-10: Moreira G., Ferreira F. e Cunha A, 2018, "News Session-Based Recommendations using Deep Neural Networks"
- SCO-11: Kim J. et al., 2018, "Towards seed-free music playlist generation enhancing collaborative filtering with playlist title information"
- SCO-12: Kumar A. et al., 2018, "Supported matrix factorization using distributed representations for personalised recommendations on twitter"
- SCO-13: Bharadhwaj H., Park H. e Lim B., 2018, "RecGan: Recurrent generative adversarial networks for recommendation systems"
- SCO-14: Sonie O., Sarkar S. e Kumar S., 2018, Concept to code: Learning distributed representation of heterogeneous sources for recommendation"
- SCO-15: Moreira G., 2018, "Chameleon - A deep learning meta-architecture for news recommender systems"
- SCO-16: Lopes R. et al., 2018, "A Knowledge-Based Recommendation System that includes Sentiment Analysis and Deep Learning"
- SCO-17: Wang L. et al., 2018, "Supervised reinforcement learning with recurrent neural network for dynamic treatment recommendation"

- SCO-18: Christakopoulou K., et al., 2018, "Q&R: A two-stage approach toward interactive recommendation"
- SCO-19: Benabderrahmane S., "Mellouli N. e Lamolle M., 2018, On the predictive analysis of behavioral massive job data using embedded clustering and deep recurrent neural networks"
- SCO-20: Lin X. et al., 2018, "K-plet recurrent neural networks for sequential recommendation"
- SCO-21: Manotumruksa J., Macdonald C., Ounis I., 2018, "A contextual attention recurrent architecture for context-aware venue recommendation"
- SCO-22: Yuan W., Yang Y., Bao X., 2018, "Parallel DNNs for users and items modeling and recommendation using comments"
- SCO-23: Qu Z. et al., 2018, "Converged Recommendation System Based on RNN and BP Neural Networks"
- SCO-24: Kong F., Li J. e Lv Z., 2018, "Construction of intelligent traffic information recommendation system based on long short-term memory"
- SCO-25: Jiang M., Yang Z., Zhao C., 2018, "What to play next? A RNN-based music recommendation system"
- SCO-26: Chen C. et al, 2018, "PERSON - Personalized Expert Recommendation System for Optimized Nutrition"
- SCO-27: Korotaev A. e Lyadova L., 2018, "Method for the development of recommendation systems, customizable to domains, with deep GRU network"
- SCO-28: Cui Q., et al., 2018, "MV-RNN: A Multi-View Recurrent Neural Network for Sequential Recommendation"
- SCO-29: Zhou T. et al., 2018, "JUMP: A joint predictor for user click and dwell time"
- SCO-30: Niu J. e Guo Y., 2018, "A multi-information fusion model for shop recommendation based on deep learning"
- SCO-31: Fu H. et al., 2018, "Sequence-based recommendation with bidirectional LSTM network"
- SCO-32: Zhao Z. et al., 2018, "A Memory-Efficient Approach to the Scalability of Recommender System with Hit Improvement"

- SCO-33: Lin Q. et al. "Heterogeneous knowledge-based attentive neural networks for short-term music recommendations"
- SCO-34: Petrova S., 2018, "Migration to graph data structures for Big Data analysis in causal model of personal curricula"
- SCO-35: Lenz D., Schulze C. e Guckert M., 2018, "Real-time session-based recommendations using LSTM with neural embeddings"
- SCO-36: Su Y. et al., 2018, "Exercise-enhanced sequential modeling for student performance prediction"
- SCO-37: Manotumruksa J., Macdonald C. e Ounis I., 2017, "A deep recurrent collaborative filtering framework for venue recommendation"
- SCO-38: Gao J., Zhang T. e Xu C., 2017, "A unified personalized video recommendation via dynamic recurrent neural networks"
- SCO-39: Preethi G., et al., 2017, "Application of Deep Learning to Sentiment Analysis for recommender system on cloud"
- SCO-40: Ruocco, M., Skrede O. e Langseth, H., 2017, "Inter-session modeling for session-based recommendation"
- SCO-41: Chu Y. et al., 2017, "Short-term recommendation with recurrent neural networks"
- SCO-42: Yin Z., Chang K. e Zhang R., 2017, "DeepProbe: Information directed sequence understanding and chatbot design via recurrent neural networks"
- SCO-43: Li S. et al., 2017, "A Method of Emotional Analysis of Movie Based on Convolution Neural Network and Bi-directional LSTM RNN"
- SCO-44: Li P. et al., 2017, "Neural rating regression with abstractive tips generation for recommendation"
- SCO-45: Hassan, H.A., 2017, "Personalized research paper recommendation using deep learning"
- SCO-46: Tang S., Wu Z. e Chen K., 2017, "Movie recommendation via BLSTM"
- SCO-47: Heinz S., Bracher C. e Vollgraf R., 2017, "An LSTM-Based dynamic customer model for fashion recommendation"
- SCO-48: Liu K. et al., 2016, "Natarajan, P. Temporal learning and sequence modeling for a job recommender system"

- SCO-49: Bansal T., Belanger D. e McCallum, A., 2016, "Ask the GRU: Multi-task learning for deep text recommendations"
- SCO-50: Zhao W.X. et al., 2016, "Connecting Social Media to E-Commerce: Cold-Start Product Recommendation Using Microblogging Information"
- SCO-51: Liu Q. et al., 2016, "Predicting the next location: A recurrent model with spatial and temporal contexts"
- SCO-52: 2016, "12th International Conference on Advanced Data Mining and Applications ADMA 2016"
- SCO-53: 2014, 2014 IEEE 7th International Workshop on Computational Intelligence and Applications, IWCIA 2014 - Proceedings
- SCO-54: Abramson M. e Gore S., 2013, "Associative patterns of web browsing behavior"
- SCO-55: 2012, "Proceedings - UKSim-AMSS 6th European Modelling Symposium, EMS 2012"
- SCO-56: Lin X. et al. 2012, "Privacy preserving reverse nearest-neighbor queries processing on road network"

Tabela 14 – Lista de inclusão e exclusão dos trabalhos

Item	Critérios de Inclusão	Critérios de Exclusão	Status
SCO-1	(I-1), (I-2), (I-3) (I-4), (I-5)	(E-2)	Rejeitado
SCO-2	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-3	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-4	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-3)	Rejeitado
SCO-5	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
SCO-6	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-7	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-8	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
SCO-9	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-10	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-11	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-12	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
SCO-13	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
SCO-14	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-15	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
SCO-16	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado

SCO-17	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-18	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-19	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
SCO-20	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-21	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-22	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
SCO-23	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-24	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
SCO-25	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-26	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
SCO-27	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
SCO-28	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-29	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
SCO-30	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
SCO-31	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
SCO-32	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-33	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-34	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
SCO-35	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-4)	Rejeitado
SCO-36	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
SCO-37	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-38	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-39	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-40	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-41	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
SCO-42	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-43	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-44	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
SCO-45	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-46	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
SCO-47	(I-1), (I-2), (I-3), (I-4), (I-5)		Aceito
SCO-48	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-49	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-1)	Rejeitado
SCO-50	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
SCO-51	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
SCO-52	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
SCO-53	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
SCO-54	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado
SCO-55	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado

SCO-56	(I-1), (I-2), (I-3), (I-4), (I-5)	(E-2)	Rejeitado

Fonte: O autor.

APÊNDICE C – CÓDIGO PARA ANÁLISE DOS DADOS DE ENTRADA

```
import operator
import matplotlib.pylab as matplotlib
import os
from datetime import datetime
import json

FILE_INPUT_PATH = "files/user_behavior.txt"
RESULTS_PARENT_DIRECTORY = "results"

numberOfLines = 0
numberOfPages = 0

def countPagesFromLine(pagesLine, pagesCounterDict):
    global numberOfPages
    pages = pagesLine.split()

    for page in pages:
        numberOfPages = numberOfPages + 1
        pagesCounterDict[page] = pagesCounterDict[page] + 1 if page in
            pagesCounterDict else 1

def countPagesFromDataset(dataset):
    global numberOfLines
    pagesCounterDict = dict()

    for line in dataset:
        numberOfLines = numberOfLines + 1
        countPagesFromLine(line, pagesCounterDict)

    return pagesCounterDict

def countFrequencyOfPages(pagesCounterDict):
    frequencyCounterDict = dict()

    for page in pagesCounterDict:
        frequencyCounterDict[pagesCounterDict[page]] =
            frequencyCounterDict[pagesCounterDict[page]] + 1 if
            pagesCounterDict[page] in frequencyCounterDict else 1
```

```
    return frequencyCounterDict

def countPagesCountPerUserFromDataset(dataset):
    pagesCounterArray = []

    for line in dataset:
        pages = line.split()
        pagesCounterArray.append(len(pages))

    return pagesCounterArray

def countAccessFrequencyPerUser(pagesCounterArray):
    frequencyCounterDict = dict()

    for pageCount in pagesCounterArray:
        frequencyCounterDict[pageCount] = frequencyCounterDict[pageCount] + 1 if
            pageCount in frequencyCounterDict else 1

    return frequencyCounterDict

print('Trying to read %s' % FILE_INPUT_PATH)
dataset = open(FILE_INPUT_PATH,"r")

print('Starting to count number of words...')
pagesCounter = countPagesFromDataset(dataset)

print('Starting to count number of words per user...')
dataset.seek(0)
pagesCountPerUserCounter = countPagesCountPerUserFromDataset(dataset)

print('Calculating most visited pages...')
mostVisitedPage = max(pagesCounter.iteritems(), key=operator.itemgetter(1))[0]

print('Calculating min visited pages...')
leastVisitedPage = min(pagesCounter.iteritems(), key=operator.itemgetter(1))[0]

print('Calculating most visits a user made...')
mostsVisitsByUser = max(pagesCountPerUserCounter)

print('Calculating least visits a user made...')
leastVisitsByUser = min(pagesCountPerUserCounter)
```

```
print('Calculating frequency of pages count...')
pageAccessFrequencyCounter = countFrequencyOfPages(pagesCounter)
pageAccessFrequencyCounter = sorted(pageAccessFrequencyCounter.items())

print('Calculating frequency of pages access per user count...')
pageAccessFrequencyPerUserCounter =
    countAccessFrequencyPerUser(pagesCountPerUserCounter)
pageAccessFrequencyPerUserCounter =
    sorted(pageAccessFrequencyPerUserCounter.items())

if not os.path.exists(RESULTS_PARENT_DIRECTORY):
    os.makedirs(RESULTS_PARENT_DIRECTORY)

print('Trying to create results directory...')
currentDatetime = datetime.now()
resultsDirectory = "%s/analysis-%s" % (RESULTS_PARENT_DIRECTORY,
    currentDatetime.strftime("%d-%m-%Y-%H:%M:%S"))

try:
    os.mkdir(resultsDirectory)
except OSError:
    print ("Creation of the directory %s failed" % resultsDirectory)
    exit()
else:
    print ("Successfully created the directory %s " % resultsDirectory)

print('Trying to create summary file...')
resultsFile = open("%s/%s" % (resultsDirectory, "summary.txt") ,"w+")

resultsFile.write('Number of users is %i\n' % numberOfLines)
resultsFile.write('Number of pages is %i\n' % len(pagesCounter))
resultsFile.write('Number of visits on pages is %i\n' % numberOfPages)
resultsFile.write('Average visited pages per user: %i\n' % (numberOfPages /
    numberOfLines))
resultsFile.write('The most visited page is %s (%i)\n' % (mostVisitedPage,
    pagesCounter[mostVisitedPage]))
resultsFile.write('The least visited page is %s (%i)\n' % (leastVisitedPage,
    pagesCounter[leastVisitedPage]))
resultsFile.write('The most visits a user made is %i\n' % mostsVisitsByUser)
resultsFile.write('The least visits a user made is %i\n' % leastVisitsByUser)
resultsFile.close()
```

```
print('Trying to create page access frequency log...')
pageAccessFrequencyFile = open("%s/%s" % (resultsDirectory,
    "page_access_frequency.txt") ,"w+")
pageAccessFrequencyFile.write(json.dumps(pageAccessFrequencyCounter))
pageAccessFrequencyFile.close()

print('Gererating frequency of pages count graph...')
x, y = zip(*pageAccessFrequencyCounter)
pageAccessFrequencyImage = matplotlib.figure()
matplotlib.plot(x, y)
pageAccessFrequencyImage.savefig("%s/%s" % (resultsDirectory,
    '/pageAccessFrequencyImage.png'), dpi=pageAccessFrequencyImage.dpi)

print('Trying to create page access frequency by user log...')
pageAccessFrequencyFileByUser = open("%s/%s" % (resultsDirectory,
    "page_access_frequency_by_user.txt") ,"w+")
pageAccessFrequencyFileByUser.write(json.dumps(pageAccessFrequencyPerUserCounter))
pageAccessFrequencyFileByUser.close()

print('Gererating frequency of pages count by user graph...')
x, y = zip(*pageAccessFrequencyPerUserCounter)
pageAccessFrequencyPerUserImage = matplotlib.figure()
matplotlib.plot(x, y)
pageAccessFrequencyPerUserImage.savefig("%s/%s" % (resultsDirectory,
    '/pageAccessFrequencyImagePerUser.png'),
    dpi=pageAccessFrequencyPerUserImage.dpi)

dataset.close()
```

APÊNDICE D – CÓDIGO PARA CONSTRUÇÃO E TREINAMENTO DAS REDES

```
import numpy as np
np.random.seed(42)
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import GRU, LSTM
from keras.layers import Dropout
from keras.layers.core import Dense, Activation, Dropout, RepeatVector
from keras.utils import np_utils
from keras.optimizers import RMSprop
from keras.optimizers import SGD
from keras.optimizers import Adagrad
from keras.optimizers import Adam
import metrics
import pickle
import json
import os
import itertools as it
from pandas import DataFrame
from keras.utils import plot_model


#CONFIG_FILE_PATH = "files/config-test.json"
CONFIG_FILE_PATH = "files/config.json"
RESULTS_DIRECTORY = "results"

DATASET_PATH = "processed_data/0.001.txt"
MAX_SEQUENCE_LENGTH = 40
NUMBER_OF_UNITS = 128
RNN_TYPE = 'lstm-gru'

def buildExperimentsCombination(configJson):
    experiments = list()

    allValues = list(configJson)
    combinations = it.product(*(config[configKey] for configKey in allValues))
    experimentNumber = 1
    combinationList = list(combinations)
```

```
    for combination in combinationList:
        experiment = dict()
        experiment['experimentId'] = 'experiment-%i' % experimentNumber
        attributeIndex = 0

        for attribute in configJson:
            experiment[attribute] = combination[attributeIndex]
            attributeIndex += 1

        experimentNumber += 1
        experiments.append(experiment)

    return experiments

def getOptimizerFromName(optimizer):
    if optimizer == 'SGD':
        return SGD(learning_rate=0.01, momentum=0.0, nesterov=False)
    elif optimizer == 'Adagrad':
        return Adagrad(learning_rate=0.01)
    elif optimizer == 'Adam':
        return Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, amsgrad=False)
    else:
        return RMSprop(lr=0.01)

def createRNN(type, experimentId, X, y, chars, maxSequenceLength,
    numberOfUnits, activationFunction, epochs, optimizerName, batchSize,
    dataFrameDict):
    model = Sequential()
    if(type == 'lstm'):
        model.add(LSTM(numberOfUnits, input_shape=(maxSequenceLength,
            len(chars))))
    elif(type == 'gru'):
        model.add(GRU(numberOfUnits, input_shape=(maxSequenceLength,
            len(chars))))
    elif(type == 'lstm-gru'):
        model.add(GRU(numberOfUnits, input_shape=(maxSequenceLength,
            len(chars)), return_sequences=True))
        model.add(LSTM(numberOfUnits, input_shape=(maxSequenceLength,
            len(chars))))
    model.add(Dense(len(chars)))
    model.add(Activation(activationFunction))
```

```
    optimizer = getOptimizerFromName(optimizerName)
    model.compile(loss='categorical_crossentropy', optimizer=optimizer,
        metrics=['accuracy', metrics.recall, metrics.precision,
        metrics.f1_score])
    model.summary()

    plot_model(model, show_shapes=True, show_layer_names=True,
        to_file='model-lstm-gru.png')
    from IPython.display import Image
    Image(retina=True, filename='model.png')

    history = model.fit(X, y, validation_split=0.2, batch_size=batchSize,
        epochs=epochs, shuffle=True).history

    resultsDirectory = '%s/lstm-gru' % RESULTS_DIRECTORY

    if not os.path.exists(resultsDirectory):
        os.makedirs(resultsDirectory)
        print('created directory %s' % resultsDirectory)

    resultsFile = open("%s/%s.txt" % (resultsDirectory, experimentId) ,"w+")
    resultsFile.write('accuracy %f\n' % history['accuracy'][epochs - 1])
    resultsFile.write('recall %f\n' % history['recall'][epochs - 1])
    resultsFile.write('precision %f\n' % history['precision'][epochs - 1])
    resultsFile.write('f1_score %f\n' % history['f1_score'][epochs - 1])
    resultsFile.close()

    dataFrameDict['experimentId'].append(experimentId)
    dataFrameDict['type'].append('lstm')
    dataFrameDict['accuracy'].append(history['accuracy'][epochs - 1])
    dataFrameDict['recall'].append(history['recall'][epochs - 1])
    dataFrameDict['precision'].append(history['precision'][epochs - 1])
    dataFrameDict['fonescore'].append(history['f1_score'][epochs - 1])

    model.summary()

    model.save('%s/%s.h5' % (resultsDirectory, experimentId))
    pickle.dump(history, open("%s/%s.p" % (resultsDirectory, experimentId),
        "wb"))


config = json.load(open(CONFIG_FILE_PATH, "r"))
```

```
if not os.path.exists(RESULTS_DIRECTORY):
    os.makedirs(RESULTS_DIRECTORY)
    print('created directory %s' % RESULTS_DIRECTORY)

experiments = buildExperimentsCombination(config)

file = open("%s" % DATASET_PATH ,"r")
dataset = file.read()
file.close()

words = dataset.split()

chars = sorted(list(set(words)))
char_indices = dict((c, i) for i, c in enumerate(chars))

print('unique chars: ', {len(chars)})

sentences = []
next_chars = []
for i in range(0, len(words) - MAX_SEQUENCE_LENGTH, 1):
    sentences.append(words[i: i + MAX_SEQUENCE_LENGTH])
    next_chars.append(words[i + MAX_SEQUENCE_LENGTH])

n_patterns = len(sentences)
n_vocab = len(chars)

print('n_patterns:', n_patterns)
print('n_vocab:', n_vocab)

#HOT ENCODING
X = np.zeros((n_patterns, MAX_SEQUENCE_LENGTH, n_vocab), dtype=np.bool)
y = np.zeros((len(sentences), len(chars)), dtype=np.bool)
for i, sentence in enumerate(sentences):
   for t, char in enumerate(sentence):
       X[i, t, char_indices[char]] = 1
   y[i, char_indices[next_chars[i]]] = 1


dataFrameDict = {
    'experimentId': [],
    'type': [],
```

```
    'accuracy': [],
    'recall': [],
    'precision': [],
    'fonescore': []
}

for experiment in experiments:
    print("starting %s" % experiment['experimentId'])
    print(experiment)
    createRNN(
        RNN_TYPE,
        experiment['experimentId'],
        X,
        y,
        chars,
        MAX_SEQUENCE_LENGTH,
        NUMBER_OF_UNITS,
        experiment['activation_function'],
        experiment['epochs'],
        experiment['optimizer'],
        experiment['batch_size'],
        dataFrameDict
    )

    print('==========================')

dataFrame = DataFrame(dataFrameDict, columns= ['experimentId', 'type',
    'accuracy', 'recall', 'precision', 'fonescore'])
dataFrame.to_csv('%s/results-lstm-gru.csv' % RESULTS_DIRECTORY, index = None,
    header=True)
```

APÊNDICE E – CONFIGURAÇÃO DO AMBIENTE DE DESENVOLVIMENTO

O desenvolvimento dos protótipos deste projeto foram realizados em um sistema Ubuntu versão 18.04.2 LTS com 16 Gb de RAM e processador do modelo Intel Core i7-7700 através da linguagem Python, na versão 2.7.15. Juntamente com o pacote de desenvolvimento do Python, outras demais bibliotecas foram utilizadas para a criação das redes neurais neste experimento. A tabela abaixo lista a relação das bibliotecas instaladas, em ordem, juntamente com o comando executado no terminal da máquina utilizada.

Tabela 15 – Bibliotecas utilizadas para o desenvolvimento dos protótipos

Nome	**Comando para instalar**
Python 2.7	sudo apt-get install python2.7
pip (gerenciador de pacotes do Python)	sudo apt install python-pip
OpenBLAS	sudo apt-get install build-essential cmake git unzip pkg-config libopenblas-dev liblapack-dev
NumPy	sudo apt-get install python-numpy
SciPy	sudo apt-get install python-scipy
matplotlib	sudo apt-get install python-matplotlib python-yaml pip install matplotlib –user
HDF5	sudo apt-get install libhdf5-serial-dev python-h5py
Graphviz	sudo apt-get install graphviz
pydot-ng	pip install pydot-ng –user
OpenCV	sudo apt-get install python-opencv
TensorFlow	pip install tensorflow –user

Keras	pip install keras –user
Pandas	pip install pandas –user

Fonte: O autor.

APÊNDICE F – COMPOSIÇÃO DE CENÁRIOS PARA O EXPERIMENTO

A partir da combinação das variáveis inseridas no modelo do experimento deste trabalho, foram gerados 108 diferentes cenários para avaliação dos resultados em cada um dos protótipos construídos. A Tabela 16 apresenta os diferentes cenários e os respectivos valores utilizados nas variáveis.

Tabela 16 – Cenários identificados a partir da combinação das variáveis do modelo

ID do experimento	Função de Ativação	Número de Épocas	Otimizador *batch*	Tamanho do
experiment-1	softmax	20	SGD	64
experiment-2	softmax	20	SGD	128
experiment-3	softmax	20	SGD	256
experiment-4	softmax	20	Adam	64
experiment-5	softmax	20	Adam	128
experiment-6	softmax	20	Adam	256
experiment-7	softmax	20	RMSprop	64
experiment-8	softmax	20	RMSprop	128
experiment-9	softmax	20	RMSprop	256
experiment-10	softmax	20	Adagrad	64
experiment-11	softmax	20	Adagrad	128
experiment-12	softmax	20	Adagrad	256
experiment-13	softmax	50	SGD	64
experiment-14	softmax	50	SGD	128
experiment-15	softmax	50	SGD	256
experiment-16	softmax	50	Adam	64
experiment-17	softmax	50	Adam	128
experiment-18	softmax	50	Adam	256
experiment-19	softmax	50	RMSprop	64
experiment-20	softmax	50	RMSprop	128
experiment-21	softmax	50	RMSprop	256
experiment-22	softmax	50	Adagrad	64
experiment-23	softmax	50	Adagrad	128
experiment-24	softmax	50	Adagrad	256
experiment-25	softmax	100	SGD	64
experiment-26	softmax	100	SGD	128

experiment-27	softmax	100	SGD	256
experiment-28	softmax	100	Adam	64
experiment-29	softmax	100	Adam	128
experiment-30	softmax	100	Adam	256
experiment-31	softmax	100	RMSprop	64
experiment-32	softmax	100	RMSprop	128
experiment-33	softmax	100	RMSprop	256
experiment-34	softmax	100	Adagrad	64
experiment-35	softmax	100	Adagrad	128
experiment-36	softmax	100	Adagrad	256
experiment-37	tanh	20	SGD	64
experiment-38	tanh	20	SGD	128
experiment-39	tanh	20	SGD	256
experiment-40	tanh	20	Adam	64
experiment-41	tanh	20	Adam	128
experiment-42	tanh	20	Adam	256
experiment-43	tanh	20	RMSprop	64
experiment-44	tanh	20	RMSprop	128
experiment-45	tanh	20	RMSprop	256
experiment-46	tanh	20	Adagrad	64
experiment-47	tanh	20	Adagrad	128
experiment-48	tanh	20	Adagrad	256
experiment-49	tanh	50	SGD	64
experiment-50	tanh	50	SGD	128
experiment-51	tanh	50	SGD	256
experiment-52	tanh	50	Adam	64
experiment-53	tanh	50	Adam	128
experiment-54	tanh	50	Adam	256
experiment-55	tanh	50	RMSprop	64
experiment-56	tanh	50	RMSprop	128
experiment-57	tanh	50	RMSprop	256
experiment-58	tanh	50	Adagrad	64
experiment-59	tanh	50	Adagrad	128
experiment-60	tanh	50	Adagrad	256
experiment-61	tanh	100	SGD	64
experiment-62	tanh	100	SGD	128
experiment-63	tanh	100	SGD	256
experiment-64	tanh	100	Adam	64
experiment-65	tanh	100	Adam	128

experiment-66	tanh	100	Adam	256
experiment-67	tanh	100	RMSprop	64
experiment-68	tanh	100	RMSprop	128
experiment-69	tanh	100	RMSprop	256
experiment-70	tanh	100	Adagrad	64
experiment-71	tanh	100	Adagrad	128
experiment-72	tanh	100	Adagrad	256
experiment-73	sigmoid	20	SGD	64
experiment-74	sigmoid	20	SGD	128
experiment-75	sigmoid	20	SGD	256
experiment-76	sigmoid	20	Adam	64
experiment-77	sigmoid	20	Adam	128
experiment-78	sigmoid	20	Adam	256
experiment-79	sigmoid	20	RMSprop	64
experiment-80	sigmoid	20	RMSprop	128
experiment-81	sigmoid	20	RMSprop	256
experiment-82	sigmoid	20	Adagrad	64
experiment-83	sigmoid	20	Adagrad	128
experiment-84	sigmoid	20	Adagrad	256
experiment-85	sigmoid	50	SGD	64
experiment-86	sigmoid	50	SGD	128
experiment-87	sigmoid	50	SGD	256
experiment-88	sigmoid	50	Adam	64
experiment-89	sigmoid	50	Adam	128
experiment-90	sigmoid	50	Adam	256
experiment-91	sigmoid	50	RMSprop	64
experiment-92	sigmoid	50	RMSprop	128
experiment-93	sigmoid	50	RMSprop	256
experiment-94	sigmoid	50	Adagrad	64
experiment-95	sigmoid	50	Adagrad	128
experiment-96	sigmoid	50	Adagrad	256
experiment-97	sigmoid	100	SGD	64
experiment-98	sigmoid	100	SGD	128
experiment-99	sigmoid	100	SGD	256
experiment-100	sigmoid	100	Adam	64
experiment-101	sigmoid	100	Adam	128
experiment-102	sigmoid	100	Adam	256
experiment-103	sigmoid	100	RMSprop	64
experiment-104	sigmoid	100	RMSprop	128

experiment-105	sigmoid	100	RMSprop	256
experiment-106	sigmoid	100	Adagrad	64
experiment-107	sigmoid	100	Adagrad	128
experiment-108	sigmoid	100	Adagrad	256

Fonte: O autor.

APÊNDICE G – RESULTADO DA EXECUÇÃO DOS PROTÓTIPOS

G.1 Protótipo de Memória de Longo Prazo (LSTM)

Tabela 17 – Cenários identificados a partir da combinação das variáveis do modelo

ID do experimento	*Accuracy*	*Recall*	*Precision*	F1-Score
experiment-1	0,0186	0,0	0,0	0,0
experiment-2	0,0171	0,0	0,0	0,0
experiment-3	0,0164	0,0	0,0	0,0
experiment-4	0,0808	0,0	0,0	0,0
experiment-5	0,0329	0,0	0,0	0,0
experiment-6	0,0279	0,0	0,0	0,0
experiment-7	0,9914	0,9813	0,9971	0,9891
experiment-8	0,8475	0,6665	0,9255	0,7725
experiment-9	0,3951	0,0329	0,5931	0,0618
experiment-10	0,1510	0,0	0,0	0,0
experiment-11	0,1173	0,0	0,0	0,0
experiment-12	0,0873	0,0	0,0	0,0
experiment-13	0,0186	0,0	0,0	0,0
experiment-14	0,0186	0,0	0,0	0,0
experiment-15	0,0128	0,0	0,0	0,0
experiment-16	0,5461	0,0028	0,1363	0,0055
experiment-17	0,1424	0,0	0,0	0,0
experiment-18	0,0773	0,0	0,0	0,0
experiment-19	0,9821	0,9710	0,9953	0,9827
experiment-20	1,0	1,0	1,0	1,0
experiment-21	0,9713	0,9578	0,9869	0,9720
experiment-22	0,4702	0,0021	0,1363	0,0041
experiment-23	0,3042	0,0	0,0	0,0
experiment-24	0,3350	0,0	0,0	0,0
experiment-25	0,0186	0,0	0,0	0,0
experiment-26	0,0186	0,0	0,0	0,0
experiment-27	0,0186	0,0	0,0	0,0
experiment-28	0,9348	0,6393	0,9805	0,7722
experiment-29	0,7766	0,1184	0,9956	0,2102

experiment-30	0,3500	0,0013	0,3333	0,0025
experiment-31	0,9849	0,9792	0,9947	0,9869
experiment-32	1,0	1,0	1,0	1,0
experiment-33	1,0	1,0	1,0	1,0
experiment-34	0,8146	0,0650	0,9757	0,1199
experiment-35	0,6914	0,0199	0,9090	0,0387
experiment-36	0,5862	0,0039	0,4999	0,0077
experiment-37	0,0007	0,1991	0,0032	0,0064
experiment-38	0,0028	0,1279	0,0037	0,0072
experiment-39	0,0035	0,0157	0,0013	0,0024
experiment-40	0,0322	0,0	0,0	0,0
experiment-41	0,0128	0,3874	0,0012	0,0025
experiment-42	0,0629	0,0	0,0	0,0
experiment-43	0,0007	0,3688	0,0007	0,0015
experiment-44	0,0078	0,3084	0,0006	0,0013
experiment-45	0,0035	0,6849	0,0013	0,0027
experiment-46	0,0128	0,4761	0,0012	0,0023
experiment-47	0,0143	0,1786	0,0014	0,0028
experiment-48	0,0365	0,0	0,0	0,0
experiment-49	0,0007	0,2376	0,0026	0,0052
experiment-50	0,0007	0,0839	0,0006	0,0013
experiment-51	0,0007	0,1089	0,0058	0,0110
experiment-52	0,0443	0,0	0,0	0,0
experiment-53	0,0722	0,0	0,0	0,0
experiment-54	0,0909	0,0	0,0	0,0
experiment-55	0,0007	0,5119	0,0010	0,0021
experiment-56	0,0064	0,3994	0,0008	0,0016
experiment-57	0,0007	0,5817	0,0012	0,0024
experiment-58	0,0107	0,4552	0,0010	0,0021
experiment-59	0,0307	0,0	0,0	0,0
experiment-60	0,0100	0,3280	0,0013	0,0026
experiment-61	0,0007	0,3536	0,0011	0,0023
experiment-62	0,0007	0,2587	0,0017	0,0034
experiment-63	0,0007	0,2026	0,0047	0,0093
experiment-64	0,0057	0,4261	0,0009	0,0018
experiment-65	0,0379	0,0	0,0	0,0
experiment-66	0,0200	0,0	0,0	0,0
experiment-67	0,0007	0,2603	0,0005	0,0011
experiment-68	0,0007	0,3291	0,0006	0,0013

experiment-69	0,0057	0,5469	0,0011	0,0023
experiment-70	0,0057	0,6980	0,0014	0,0028
experiment-71	0,0064	0,4009	0,0008	0,0016
experiment-72	0,0021	0,3566	0,0020	0,0040
experiment-73	0,0164	0,6533	0,0014	0,0028
experiment-74	0,0100	0,6258	0,0013	0,0026
experiment-75	0,0042	0,5590	0,0011	0,0023
experiment-76	0,0186	0,3076	0,0066	0,0130
experiment-77	0,0121	0,3165	0,0062	0,0122
experiment-78	0,0186	0,3287	0,0057	0,0113
experiment-79	0,9398	0,0593	0,8755	0,1073
experiment-80	0,7015	0,0742	0,5994	0,1260
experiment-81	0,0236	0,3122	0,0122	0,0235
experiment-82	0,0186	0,2441	0,0221	0,0406
experiment-83	0,0229	0,2989	0,0129	0,0248
experiment-84	0,0214	0,3295	0,0071	0,0140
experiment-85	0,0186	0,6659	0,0015	0,0030
experiment-86	0,0200	0,6553	0,0014	0,0028
experiment-87	0,0178	0,6460	0,0013	0,0027
experiment-88	0,0272	0,0842	0,0477	0,0594
experiment-89	0,0186	0,2551	0,0114	0,0218
experiment-90	0,0186	0,2937	0,0067	0,0132
experiment-91	1,0	0,0672	0,9545	0,1229
experiment-92	1,0	0,0384	1,0	0,0735
experiment-93	1,0	0,0566	1,0	0,1059
experiment-94	0,2949	0,1418	0,2633	0,1809
experiment-95	0,0186	0,2211	0,1050	0,1423
experiment-96	0,0200	0,2501	0,0316	0,0561
experiment-97	0,0186	0,6779	0,0016	0,0033
experiment-98	0,0186	0,6482	0,0015	0,0031
experiment-99	0,0186	0,6598	0,0014	0,0029
experiment-100	0,5633	0,0042	0,0544	0,0054
experiment-101	0,0400	0,0456	0,0332	0,0384
experiment-102	0,0186	0,2409	0,0135	0,0256
experiment-103	0,0007	0,0	0,0	0,0
experiment-104	1,0	0,3296	1,0	0,4937
experiment-105	1,0	0,0582	1,0	0,1094
experiment-106	0,6313	0,1075	0,5811	0,1787
experiment-107	0,5633	0,0716	0,5300	0,1248

experiment-108	0,3514	0,0567	0,3020	0,0949

Fonte: O autor.

G.2 Protótipo de Unidades Recorrentes de Bloqueio (GRU)

Tabela 18 – Cenários identificados a partir da combinação das variáveis do modelo

ID do experimento	***Accuracy***	***Recall***	***Precision***	**F1-Score**
experiment-1	0,0186	0,0	0,0	0,0
experiment-2	0,0143	0,0	0,0	0,0
experiment-3	0,0064	0,0	0,0	0,0
experiment-4	0,1617	0,0	0,0	0,0
experiment-5	0,0637	0,0	0,0	0,0
experiment-6	0,0307	0,0	0,0	0,0
experiment-7	0,9978	0,9978	0,9992	0,9985
experiment-8	0,9964	0,9934	0,9985	0,9959
experiment-9	0,9928	0,9855	0,9980	0,9917
experiment-10	0,1753	0,0007	0,0454	0,0013
experiment-11	0,1374	0,0	0,0	0,0
experiment-12	0,1173	0,0	0,0	0,0
experiment-13	0,0186	0,0	0,0	0,0
experiment-14	0,0178	0,0	0,0	0,0
experiment-15	0,0207	0,0	0,0	0,0
experiment-16	0,9906	0,8058	1,0	0,8915
experiment-17	0,6506	0,0243	0,9848	0,0472
experiment-18	0,1889	0,0	0,0	0,0
experiment-19	0,9964	0,9940	0,9962	0,9951
experiment-20	1,0	1,0	1,0	1,0
experiment-21	1,0	1,0	1,0	1,0
experiment-22	0,5483	0,0072	0,3636	0,0141
experiment-23	0,4302	0,0007	0,0909	0,0014
experiment-24	0,4087	0,0033	0,4999	0,0067
experiment-25	0,0186	0,0	0,0	0,0
experiment-26	0,0186	0,0	0,0	0,0
experiment-27	0,0186	0,0	0,0	0,0
experiment-28	1,0	1,0	1,0	1,0
experiment-29	1,0	0,9942	1,0	0,9971

experiment-30	0,9506	0,3669	0,9979	0,5360
experiment-31	0,9992	0,9992	0,9992	0,9992
experiment-32	1,0	1,0	1,0	1,0
experiment-33	1,0	1,0	1,0	1,0
experiment-34	0,8682	0,1138	0,9967	0,2017
experiment-35	0,8969	0,0824	0,9878	0,1509
experiment-36	0,8210	0,0609	0,9916	0,1142
experiment-37	0,0007	0,4265	0,0013	0,0026
experiment-38	0,0028	0,1146	0,0022	0,0044
experiment-39	0,0007	0,0046	0,0007	0,0013
experiment-40	0,0930	0,0	0,0	0,0
experiment-41	0,0665	0,0	0,0	0,0
experiment-42	0,1503	0,0	0,0	0,0
experiment-43	0,0042	0,6896	0,0014	0,0029
experiment-44	0,0014	0,3978	0,0008	0,0016
experiment-45	0,0007	0,2749	0,0005	0,0011
experiment-46	0,0286	0,0	0,0	0,0
experiment-47	0,0350	0,0	0,0	0,0
experiment-48	0,0665	0,0	0,0	0,0
experiment-49	0,0007	0,4420	0,0009	0,0018
experiment-50	0,0035	0,0697	0,0049	0,0092
experiment-51	0,0007	0,0132	0,0013	0,0024
experiment-52	0,0415	0,0	0,0	0,0
experiment-53	0,0894	0,0	0,0	0,0
experiment-54	0,0665	0,0	0,0	0,0
experiment-55	0,0014	0,4566	0,0009	0,0019
experiment-56	0,0035	0,4431	0,0009	0,0018
experiment-57	0,0007	0,2578	0,0005	0,0010
experiment-58	0,0042	0,5892	0,0014	0,0028
experiment-59	0,0250	0,0	0,0	0,0
experiment-60	0,0128	0,1092	0,0009	0,0019
experiment-61	0,0014	0,1844	0,0061	0,0118
experiment-62	0,0007	0,2224	0,0046	0,0090
experiment-63	0,0021	0,0542	0,0036	0,0068
experiment-64	0,0021	0,5468	0,0011	0,0023
experiment-65	0,0708	0,0	0,0	0,0
experiment-66	0,0909	0,0	0,0	0,0
experiment-67	0,0114	0,6843	0,0014	0,0028
experiment-68	0,0	0,2701	0,0005	0,0011

experiment-69	0,0064	0,5291	0,0011	0,0022
experiment-70	0,0415	0,0	0,0	0,0
experiment-71	0,0393	0,0	0,0	0,0
experiment-72	0,0178	0,0040	0,0091	0,0055
experiment-73	0,0078	0,6297	0,0013	0,0026
experiment-74	0,0093	0,5862	0,0012	0,0024
experiment-75	0,0028	0,5651	0,0011	0,0023
experiment-76	0,0186	0,4033	0,0097	0,0189
experiment-77	0,0186	0,3515	0,0063	0,0125
experiment-78	0,0186	0,3409	0,0057	0,0112
experiment-79	0,9992	0,1571	1,0	0,2675
experiment-80	0,9957	0,1925	0,9960	0,3148
experiment-81	0,9763	0,1985	0,9486	0,3065
experiment-82	0,0200	0,3070	0,0112	0,0216
experiment-83	0,0243	0,3507	0,0077	0,0150
experiment-84	0,0293	0,3377	0,0079	0,0155
experiment-85	0,0186	0,6545	0,0015	0,0029
experiment-86	0,0178	0,6245	0,0013	0,0026
experiment-87	0,0150	0,5954	0,0012	0,0025
experiment-88	0,6993	0,0845	0,5314	0,1404
experiment-89	0,0186	0,2057	0,0735	0,1081
experiment-90	0,0186	0,3521	0,0111	0,0215
experiment-91	0,0007	0,0	0,0	0,0
experiment-92	0,9885	0,1042	1,0	0,1840
experiment-93	1,0	0,1905	1,0	0,3160
experiment-94	0,1245	0,1122	0,0707	0,0846
experiment-95	0,1009	0,1504	0,0811	0,1049
experiment-96	0,0186	0,2151	0,0450	0,0742
experiment-97	0,0186	0,6699	0,0015	0,0031
experiment-98	0,0186	0,6591	0,0015	0,0030
experiment-99	0,0178	0,6374	0,0013	0,0027
experiment-100	0,9985	0,0762	1,0	0,1371
experiment-101	0,9763	0,0532	0,9090	0,0997
experiment-102	0,5612	0,0544	0,1272	0,0443
experiment-103	0,0007	0,0	0,0	0,0
experiment-104	0,0007	0,0	0,0	0,0
experiment-105	1,0	0,1637	1,0	0,2811
experiment-106	0,5819	0,0777	0,3582	0,1256
experiment-107	0,4588	0,0661	0,4867	0,1152

experiment-108	0,5390	0,0783	0,4612	0,1333

Fonte: O autor.

G.3 Protótipo de Redes Híbridas

Tabela 19 – Cenários identificados a partir da combinação das variáveis do modelo

ID do experimento	***Accuracy***	***Recall***	***Precision***	**F1-Score**
experiment-1	0,0186	0,0	0,0	0,0
experiment-2	0,0164	0,0	0,0	0,0
experiment-3	0,0186	0,0	0,0	0,0
experiment-4	0,1195	0,0	0,0	0,0
experiment-5	0,0343	0,0	0,0	0,0
experiment-6	0,0207	0,0	0,0	0,0
experiment-7	0,3278	0,0384	0,7473	0,0722
experiment-8	0,9863	0,9750	0,9956	0,9851
experiment-9	0,0157	0,0	0,0	0,0
experiment-10	0,1374	0,0	0,0	0,0
experiment-11	0,1166	0,0	0,0	0,0
experiment-12	0,0665	0,0	0,0	0,0
experiment-13	0,0186	0,0	0,0	0,0
experiment-14	0,0186	0,0	0,0	0,0
experiment-15	0,0243	0,0	0,0	0,0
experiment-16	0,7637	0,0210	0,8636	0,0408
experiment-17	0,3586	0,0	0,0	0,0
experiment-18	0,0887	0,0	0,0	0,0
experiment-19	0,9978	0,9957	0,9992	0,9975
experiment-20	0,2247	0,0021	0,1363	0,0041
experiment-21	0,9527	0,9261	0,9671	0,9458
experiment-22	0,4831	0,0	0,0	0,0
experiment-23	0,4180	0,0	0,0	0,0
experiment-24	0,2956	0,0	0,0	0,0
experiment-25	0,0186	0,0	0,0	0,0
experiment-26	0,0186	0,0	0,0	0,0
experiment-27	0,0186	0,0	0,0	0,0
experiment-28	0,9928	0,8368	1,0	0,9102
experiment-29	0,8324	0,0943	1,0	0,1716

experiment-30	0,5032	0,0027	0,4999	0,0054
experiment-31	1,0	1,0	1,0	1,0
experiment-32	1,0	1,0	1,0	1,0
experiment-33	1,0	1,0	1,0	1,0
experiment-34	0,7451	0,0173	0,6818	0,0336
experiment-35	0,7530	0,0049	0,4545	0,0098
experiment-36	0,6120	0,0014	0,1666	0,0028
experiment-37	0,0007	0,1362	0,0061	0,0116
experiment-38	0,0007	0,0216	0,0030	0,0054
experiment-39	0,0021	0,0045	0,0015	0,0022
experiment-40	0,0071	0,5080	0,0012	0,0024
experiment-41	0,0121	0,3398	0,0035	0,0070
experiment-42	0,0164	0,0	0,0	0,0
experiment-43	0,0007	0,4741	0,0010	0,0020
experiment-44	0,0050	0,3134	0,0006	0,0013
experiment-45	0,0057	0,7125	0,0015	0,0030
experiment-46	0,0007	0,3579	0,0007	0,0014
experiment-47	0,0007	0,1167	0,0030	0,0059
experiment-48	0,0021	0,2406	0,0006	0,0012
experiment-49	0,0007	0,3826	0,0021	0,0043
experiment-50	0,0014	0,3285	0,0027	0,0054
experiment-51	0,0007	0,1710	0,0029	0,0057
experiment-52	0,0128	0,0	0,0	0,0
experiment-53	0,0186	0,6384	0,0008	0,0017
experiment-54	0,0207	0,0	0,0	0,0
experiment-55	0,0007	0,6352	0,0013	0,0026
experiment-56	0,0007	0,2653	0,0005	0,0011
experiment-57	0,0057	0,4907	0,0010	0,0020
experiment-58	0,0071	0,2941	0,0006	0,0012
experiment-59	0,0128	0,1630	0,0013	0,0025
experiment-60	0,0007	0,7424	0,0008	0,0017
experiment-61	0,0042	0,3775	0,0013	0,0027
experiment-62	0,0007	0,3748	0,0020	0,0040
experiment-63	0,0007	0,1717	0,0013	0,0027
experiment-64	0,0064	0,6372	0,0013	0,0027
experiment-65	0,0150	0,0	0,0	0,0
experiment-66	0,0186	0,0	0,0	0,0
experiment-67	0,0007	0,2888	0,0006	0,0012
experiment-68	0,0057	0,7271	0,0015	0,0030

experiment-69	0,0028	0,6666	0,0014	0,0028
experiment-70	0,0064	0,3807	0,0014	0,0029
experiment-71	0,0028	0,4204	0,0008	0,0017
experiment-72	0,0064	0,4658	0,0011	0,0022
experiment-73	0,0164	0,6602	0,0014	0,0029
experiment-74	0,0164	0,6359	0,0013	0,0027
experiment-75	0,0136	0,6115	0,0013	0,0026
experiment-76	0,0200	0,4326	0,0083	0,0163
experiment-77	0,0186	0,3046	0,0063	0,0123
experiment-78	0,0128	0,3371	0,0057	0,0112
experiment-79	0,0751	0,0014	0,0052	0,0021
experiment-80	0,9792	0,1040	0,9601	0,1776
experiment-81	0,4280	0,0750	0,1158	0,0852
experiment-82	0,0207	0,3242	0,0197	0,0372
experiment-83	0,0193	0,4365	0,0115	0,0224
experiment-84	0,0186	0,4423	0,0082	0,0161
experiment-85	0,0186	0,6683	0,0015	0,0030
experiment-86	0,0171	0,6639	0,0015	0,0030
experiment-87	0,0186	0,6712	0,0014	0,0028
experiment-88	0,0314	0,1630	0,0130	0,0240
experiment-89	0,0171	0,4046	0,0070	0,0139
experiment-90	0,0186	0,2743	0,0066	0,0130
experiment-91	0,0007	0,0	0,0	0,0
experiment-92	1,0	0,0738	1,0	0,1367
experiment-93	1,0	0,0345	0,8333	0,0654
experiment-94	0,1947	0,1369	0,1144	0,1233
experiment-95	0,0722	0,2001	0,0404	0,0672
experiment-96	0,0193	0,3092	0,0235	0,0437
experiment-97	0,0214	0,6591	0,0015	0,0030
experiment-98	0,0186	0,6642	0,0015	0,0030
experiment-99	0,0186	0,6827	0,0014	0,0029
experiment-100	0,8503	0,0217	0,8068	0,0420
experiment-101	0,2075	0,0756	0,0704	0,0715
experiment-102	0,0136	0,2397	0,0076	0,0148
experiment-103	0,0007	0,0	0,0	0,0
experiment-104	0,0007	0,0	0,0	0,0
experiment-105	1,0	0,0683	1,0	0,1272
experiment-106	0,4581	0,0720	0,3496	0,1165
experiment-107	0,3894	0,0758	0,1802	0,1060

experiment-108	0,2161	0,0988	0,0663	0,0790

Fonte: O autor.