

UNIVERSIDADE ESTADUAL DE MARINGÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

IGOR ANDRÉ PEGORARO SANTANA

**Exploração de redes neurais recorrentes na recomendação sensível ao
contexto de músicas**

MARINGÁ

2020

IGOR ANDRÉ PEGORARO SANTANA

Exploração de redes neurais recorrentes na recomendação sensível ao contexto de músicas

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação.

Orientador: Marcos Aurélio Domingues

MARINGÁ

2020

Dados Internacionais de Catalogação-na-Publicação (CIP)
(Biblioteca Central - UEM, Maringá - PR, Brasil)

S232e

Santana, Igor André Pegoraro

Exploração de redes neurais recorrentes na recomendação sensível ao contexto de músicas / Igor André Pegoraro Santana. -- Maringá, PR, 2020.
84 f.: il. color., figs., tabs.

Orientador: Prof. Dr. Marcos Aurélio Domingues.

Dissertação (Mestrado) - Universidade Estadual de Maringá, Centro de Tecnologia, Departamento de Informática, Programa de Pós-Graduação em Ciência da Computação, 2020.

1. Redes neurais recorrentes. 2. Sistemas de recomendação sensíveis ao contexto. 3. Sistemas de recomendação de músicas. 4. *Embeddings*. I. Domingues, Marcos Aurélio , orient. II. Universidade Estadual de Maringá. Centro de Tecnologia. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. III. Título.

CDD 23.ed. 004

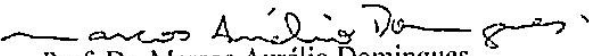
FOLHA DE APROVAÇÃO

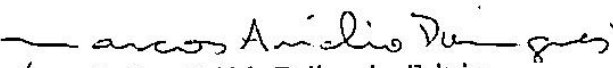
IGOR ANDRÉ PEGORARO SANTANA

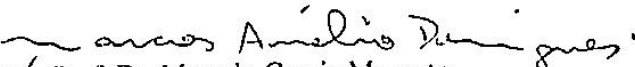
Exploração de redes neurais recorrentes na recomendação sensível ao contexto de músicas

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Departamento de Informática, Centro de Tecnologia da Universidade Estadual de Maringá, como requisito parcial para obtenção do título de Mestre em Ciência da Computação pela Banca Examinadora composta pelos membros:

BANCA EXAMINADORA


Prof. Dr. Marcos Aurélio Domingues
Universidade Estadual de Maringá – DIN/UEM


p/ Prof. Dra. Valéria Delisandra Feltrim
Universidade Estadual de Maringá – DIN/UEM


p/ Prof. Dr. Marcelo Garcia Manzato
Universidade de São Paulo – ICMC/USP

Aprovada em: 13 de agosto de 2020.
Local da defesa: Sala virtual
<https://mceet.google.com/bqu-vmce-oeK>.

DEDICATÓRIA

À memória de Eunice Trentin Sant'Anna e Neuli Carneiro Bonfim, que se foram há pouco tempo, mas fizeram parte integral da minha vida. Obrigado por tudo.

AGRADECIMENTOS

Aos meus pais e ao meu irmão, agradeço por todo o amor e ensinamentos durante toda a minha vida. Vocês foram - e sempre serão - essenciais em minha vida, e espero algum dia retribuir tudo que fizeram por mim.

À Universidade Estadual de Maringá, por ter sido minha segunda casa durante os últimos anos de graduação e mestrado.

Aos meus amigos da UEM que me fazem companhia há anos, e que se não fosse eles não sei se teria conseguido terminar a graduação e o mestrado.

Aos meus amigos, por sua companhia e amizade, que por tantos anos, estiveram junto comigo nessa caminhada.

Aos meus professores, por terem me proporcionado conhecimento durante todos esses anos.

Ao meu orientador, Prof. Dr. Marcos Aurélio Domingues, pela orientação por todos esses anos de pesquisa.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPQ, pelo apoio financeiro durante o mestrado.

Ao meu primo Rafael, por nossas infinitas conversas que por muitas vezes, são sobre assunto nenhum. Que elas durem muitos ciclos.

Ao meu amigo Renan, que tanto me ajuda no meu caminho de ser uma pessoa melhor durante os vários anos de amizade.

À todas as pessoas que fizeram parte desse caminho.

Exploração de redes neurais recorrentes na recomendação sensível ao contexto de músicas

RESUMO

Serviços de distribuição de conteúdo *online* contribuem com um volume cada vez maior de dados na *internet* e, dentre esses serviços, plataformas de *streaming* de músicas crescem cada dia mais em número de usuários e no tamanho de seu catálogo. Para auxiliar o usuário a encontrar músicas de acordo com seu interesse, sistemas de recomendação de músicas podem ser utilizados para filtrar uma grande quantidade de músicas de acordo com o perfil do usuário. No entanto, o contexto no qual o usuário ouvirá a música deve ser levado em consideração, o que justifica a utilização de sistemas de recomendação sensíveis ao contexto. Embora haja trabalhos sobre sistemas de recomendação sensíveis ao contexto de músicas, não há muitas técnicas automáticas para obter a informação contextual para esses sistemas. Este trabalho teve como objetivo utilizar Redes Neurais Recorrentes para obter informação contextual (vetor de *embeddings*) para cada música, por meio de uma análise da sequência de músicas que os usuários ouviram. Os vetores de *embeddings* foram utilizados por quatro sistemas de recomendação sensíveis ao contexto de músicas e a avaliação desses sistemas foi feita em duas bases de dados. Os resultados obtidos mostraram que os *embeddings* (informação contextual) obtidos pela Rede Neural Recorrente proposta obtivera melhores resultados nessas duas bases de dados que o modelo de *baseline* para todas as métricas utilizadas na avaliação.

Palavras-chave: Redes Neurais Recorrentes, Sistemas de Recomendação Sensíveis ao Contexto, Sistemas de Recomendação de Músicas, *Embeddings*, Aquisição de Contexto.

Exploiting recurrent neural networks in context-aware music recommendations

ABSTRACT

Day by day, online content delivery services suppliers grow the volume of data on the internet. Music streaming services are one of those services that increase the number of users every day, as well as the number of songs in their catalog. To help their users to find songs that fit their interests, music recommender systems can be used to filter a large number of songs according to the preference of the user. However, the context in which the user listens to songs must be taken into account, which justifies the usage of context-aware recommender systems. Although there are some works about context-aware music recommender systems, there is a lack of automatic techniques for extracting contextual information for these systems. The goal of this work is to propose a Recurrent Neural Network to acquire contextual information (embeddings) for each song, given the sequence of songs that each user has listened to. These embeddings were used by four context-aware music recommender systems and evaluated in two datasets. The results showed that the embeddings (contextual information) obtained by the Recurrent Neural Network, proposed in this work, present better results than the baseline model in both datasets for all metrics used in the evaluation.

Keywords: Recurrent Neural Networks, Context-Aware Recommender Systems, Music Recommendation, Embeddings, Context Acquisition.

LISTA DE QUADROS

QUADRO 1	–	Sumário dos trabalhos obtidos pela revisão sistemática.	48
QUADRO 2	–	Subconjunto da base de dados <i>Music4All</i>	50
QUADRO 3	–	Subconjunto da base de dados <i>Music4All</i> com sessões.	58
QUADRO 4	–	Atributos presentes para cada música na base de dados <i>Music4All</i>	62
QUADRO 5	–	Classificação de uma possível recomendação de uma música para um usuário.	67

LISTA DE FIGURAS

FIGURA 1	– Sistemas de recomendação de filtragem colaborativa.	18
FIGURA 2	– Sistemas de recomendação baseados em conteúdo.	21
FIGURA 3	– Sistemas de recomendação baseados em conhecimento.	23
FIGURA 4	– Classificação de sistemas de recomendação sensíveis ao contexto.	26
FIGURA 5	– Exemplo de representação de palavras como <i>embeddings</i>	28
FIGURA 6	– Modelos <i>Skip-Gram</i> e CBOV propostos por Mikolov et al. (2013a).	29
FIGURA 7	– Exemplo de janela contextual de tamanho 2.	30
FIGURA 8	– Neurônio genérico de uma rede neural.	32
FIGURA 9	– Modelo genérico de uma MLP.	33
FIGURA 10	– Modelo genérico de uma RNN.	34
FIGURA 11	– RNN que foi desdobrada de acordo com os índices de tempo.	35
FIGURA 12	– Exemplo da atuação de uma RNN com uma frase.	35
FIGURA 13	– Interior de um bloco de memória de uma LSTM.	39
FIGURA 14	– Número de trabalhos publicados por ano.	42
FIGURA 15	– Domínios explorados pelos trabalhos.	43
FIGURA 16	– Domínios explorados na categoria de Redes Neurais <i>feedforward</i>	45
FIGURA 17	– Métodos propostos por Wang et al. (2018).	51
FIGURA 18	– Modelo conjunto de aprendizado de <i>embeddings</i> de músicas.	57
FIGURA 19	– Processo de criação da base de dados <i>Music4All</i>	60
FIGURA 20	– Estatísticas de <i>tags</i> e gêneros na base de dados <i>Music4All</i>	62
FIGURA 21	– Execuções por música em ambas as bases de dados.	65
FIGURA 22	– Processo de <i>5-fold cross-validation</i> utilizado no trabalho.	66
FIGURA 23	– Resultados dos sistemas de recomendação para a base de dados <i>Music4All</i> com 5 recomendações.	71
FIGURA 24	– Resultados dos sistemas de recomendação para a base de dados <i>Xiami</i> com 5 recomendações.	72
FIGURA 25	– Resultados dos sistemas de recomendação para a base de dados <i>Xiami</i> com 10 recomendações.	72
FIGURA 26	– Resultados dos sistemas de recomendação para a base de dados <i>Music4All</i> com 10 recomendações.	73
FIGURA 27	– Processo da revisão sistemática.	86

LISTA DE TABELAS

TABELA 1	– Exemplo de matriz de avaliações.	19
TABELA 2	– Parâmetros utilizados para a obtenção dos resultados.	69
TABELA 3	– Parâmetros utilizados para a obtenção de <i>embeddings</i> propostos por Wang et al. (2018).	69
TABELA 4	– Resultados dos sistemas de recomendação para a base de dados <i>Music4All</i> com 5 recomendações.	70
TABELA 5	– Resultados dos sistemas de recomendação para a base de dados <i>Xiami</i> com 5 recomendações.	70
TABELA 6	– Resultados dos sistemas de recomendação para a base de dados <i>Music4All</i> com 10 recomendações.	73
TABELA 7	– Resultados dos sistemas de recomendação para a base de dados <i>Xiami</i> com 10 recomendações.	74

SUMÁRIO

1	INTRODUÇÃO	12
1.1	HIPÓTESE E OBJETIVO	14
1.2	PROPOSTA E RESULTADOS	14
1.3	ORGANIZAÇÃO DO TEXTO	15
2	FUNDAMENTAÇÃO TEÓRICA	16
2.1	SISTEMAS DE RECOMENDAÇÃO	16
2.1.1	Sistemas de Recomendação Sensíveis ao Contexto	24
2.2	EMBEDDINGS	27
2.2.1	Word2vec	28
2.3	REDES NEURAIIS RECORRENTES	31
2.3.1	Redes Long Short-Term Memory	38
3	TRABALHOS RELACIONADOS	42
4	OBTENÇÃO DE EMBEDDINGS COMO INFORMAÇÃO CONTEXTUAL	49
4.1	OBTENÇÃO DE EMBEDDINGS	49
4.2	MODELO DO <i>BASELINE</i>	49
4.2.1	Music2vec e Session-Music2vec	50
4.3	SISTEMAS DE RECOMENDAÇÃO CONTEXTUAIS	53
5	DESENVOLVIMENTOS	56
5.1	MODELO CONJUNTO DE APRENDIZADO DE EMBEDDINGS BASEADO EM REDES NEURAIIS RECORRENTE	56
5.2	MUSIC4ALL	59
6	AVALIAÇÃO EXPERIMENTAL	64
6.1	BASES DE DADOS	64
6.2	METODOLOGIA DE AVALIAÇÃO	65
6.3	RESULTADOS	68
7	CONCLUSÃO E TRABALHOS FUTUROS	75
	REFERÊNCIAS	78
	Apêndice A – PROTOCOLO DA REVISÃO SISTEMÁTICA	84

1 INTRODUÇÃO

Com a popularização de serviços *online* especializados em distribuição de conteúdo, o volume de informações disponibilizados por esses sistemas cresce cada vez mais. Esses serviços estão nos mais diversos domínios, porém o seu maior foco está na área de *streaming* de mídias digitais, no qual os usuários possuem acesso a um grande catálogo de filmes, músicas, livros, entre outras mídias.

Dentre esses serviços, uma área que migrou de maneira definitiva para serviços digitais é a área fonográfica. De acordo com o IFPI (2019), no ano de 2018 houve um crescimento de 32,9% em serviços de *streaming* pagos de músicas em relação ao ano de 2017, que já possuía um crescimento de 45,5% em relação ao ano de 2016, com 255 milhões de usuários tendo contas pagas em serviços de *streaming* de músicas.

Com o crescimento do *streaming* e da distribuição de músicas, no entanto, o volume de informação disponível para os usuários desses serviços tende a crescer. O Spotify¹, por exemplo, possui cerca de 50 milhões de músicas disponíveis em seu catálogo. Para lidar com essa quantidade de informação, vê-se necessária a implementação de ferramentas que auxiliem os usuários desses serviços a lidarem com o conhecido problema de sobrecarga de informação, que ocorre quando a quantidade de informação apresentada para o usuário ultrapassa a sua capacidade de processá-la (EPPLER; MENGIS, 2004).

Junto com os serviços de *streaming*, uma mudança radical no modo como as pessoas ouvem música ocorreu com a introdução de *smartphones* e demais dispositivos móveis. Enquanto está ouvindo música, uma pessoa pode estar acessando suas redes sociais, conversando com outras pessoas em aplicativos de mensagens e respondendo *e-mails*, utilizando a música apenas como plano de fundo. Com isso, pode-se dizer que uma pessoa está inserida em um determinado contexto enquanto está ouvindo as músicas. Além disso, Kim et al. (2002) demonstraram que as pessoas, de maneira geral, procuram músicas de acordo com alguma ocasião, evento ou estado emocional.

¹<https://newsroom.spotify.com/company-info>

Uma ferramenta eficaz para lidar com a sobrecarga de informação são sistemas de recomendação, que são capazes de recomendar músicas de acordo com a preferência do usuário. Diversos trabalhos foram publicados na área de sistemas de recomendação com enfoque em recomendação de músicas (CHEN; CHEN, 2005; BU et al., 2010; KOENIGSTEIN et al., 2011), porém o trabalho de Celma (2010) ganhou destaque, pois revisou sistemas que eram considerados estado da arte e identificou suas características em comuns, métricas que podem ser utilizadas para avaliar esses sistemas e características que bases de dados de sistemas de recomendação de músicas possuem.

Porém, sabendo que as pessoas ouvem músicas de acordo com o contexto no qual estão inseridas, vê-se necessária a utilização de sistemas de recomendação que possam utilizar essas informações contextuais, visto que abordagens tradicionais não são capazes de incorporá-las em seus modelos. Assim, para utilizar essas informações, é necessário utilizar um Sistema de Recomendação Sensível ao Contexto. O trabalho de Kaminskis e Ricci (2012) revisou diversos sistemas de recomendação de músicas sensíveis ao contexto, evidenciando que esse tipo de sistema funciona de maneira efetiva para esse domínio. Já adquirir a informação contextual para esses sistemas de recomendação, no entanto, não é uma tarefa trivial. Assim, o objetivo principal deste trabalho consiste em obter informações contextuais, que podem ser utilizadas no problema de recomendação contextual de músicas, utilizando Redes Neurais Recorrentes (RNNs).

Nas últimas décadas, a utilização de arquiteturas de Redes Neurais Recorrentes obteve grande sucesso em diferentes domínios, como nas áreas de reconhecimento de fala e escrita, obtendo resultados que são considerados estado da arte (COVINGTON et al., 2016). De maneira geral, essas arquiteturas são capazes de capturar relacionamentos entre usuários e itens de maneira efetiva, além de sua capacidade de encontrar relacionamentos intrínsecos nos dados, contanto que exista uma grande quantidade de dados disponíveis para serem analisados (ZHANG et al., 2019).

Apesar de existirem diversas arquiteturas de redes neurais que foram exploradas junto com sistemas de recomendação, como visto em Zhang et al. (2019), existe uma lacuna com relação a trabalhos que explorem arquiteturas de Redes Neurais Recorrentes para obtenção de informação contextual para sistemas de recomendação sensíveis ao contexto.

1.1 HIPÓTESE E OBJETIVO

Diante do aumento da quantidade de músicas e da eficiência das Redes Neurais Recorrentes, a hipótese levantada neste trabalho é que utilizar informações intrínsecas de contexto obtidas por Redes Neurais Recorrentes melhora as recomendações feitas por sistemas de recomendação sensíveis ao contexto de músicas.

Sendo assim, o objetivo do trabalho é propor um modelo que utiliza Redes Neurais Recorrentes capaz de obter informações contextuais a partir da sequência de músicas ouvidas pelo usuário, utilizando seu histórico de execução.

1.2 PROPOSTA E RESULTADOS

Nesta dissertação de mestrado foi proposta uma arquitetura de RNN que utiliza redes *Long Short-Term Memory* (LSTM) para analisar uma sequência de músicas que os usuários ouviram e obter um vetor de *embeddings* (um vetor de números reais para cada música). Esse vetor de *embeddings* pode ser considerado contextual, pois a RNN analisa quais músicas apareceram na sequência de músicas ouvidas antes e depois de cada música.

Como *baseline* de comparação para a arquitetura proposta, foi utilizado o modelo proposto por Wang et al. (2018), que tem como objetivo extrair *embeddings* de músicas, considerado estado da arte na área de recomendação sensível ao contexto de músicas. Para executar a avaliação experimental com a arquitetura proposta e a de *baseline*, foram utilizados quatro sistemas de recomendação de músicas sensíveis ao contexto desenvolvidos por Wang et al. (2018).

Os sistemas de recomendação foram executados em duas bases de dados que possuem o histórico de execução de músicas de milhares de usuários. Uma das bases utilizou um *crawler* do serviço de música chinês *Xiami*² e a segunda base de dados, chamada *Music4All*, foi criada a partir do histórico de execução dos usuários do last.fm³. Os resultados obtidos mostram que a arquitetura de Rede Neural Recorrente proposta apresentou melhor resultado que o *baseline*, obtendo vetores de *embeddings* capazes de capturar informações contextuais superiores em ambas as bases de dados.

²<https://www.xiami.com>

³<https://www.last.fm>

1.3 ORGANIZAÇÃO DO TEXTO

O restante deste trabalho está organizado da seguinte forma. A Seção 2 apresenta conceitos sobre sistemas de recomendação tradicionais e sensíveis ao contexto, *embeddings*, word2vec e redes neurais, com enfoque em Redes Neurais Recorrentes. A Seção 3 apresenta os resultados de uma revisão sistemática que foi realizada para obter trabalhos que utilizaram técnicas de obtenção de *embeddings* para sistemas de recomendação sensíveis ao contexto. Na Seção 4, são apresentados os modelos utilizados como *baseline* para extração de *embeddings* e os sistemas de recomendação utilizados para avaliar tais modelos. Na Seção 5, a arquitetura de extração de *embeddings* proposta é apresentada, junto com a base de dados proposta no trabalho. As bases de dados, a metodologia de avaliação, e os resultados obtidos com este trabalho de mestrado são apresentados na Seção 6. Finalmente, na Seção 7, são apresentadas as conclusões e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção são apresentados conceitos sobre sistemas de recomendação tradicionais e sensíveis ao conceito, *embeddings*, *word2vec* e Redes Neurais Recorrentes que serão utilizados ao longo desta dissertação.

2.1 SISTEMAS DE RECOMENDAÇÃO

Os sistemas de recomendação foram desenvolvidos com o objetivo de auxiliar usuários que não possuem a experiência ou competência para avaliar grandes quantidades de itens disponibilizados por um *website* (RESNICK; VARIAN, 1997). De acordo com Ricci et al. (2015), são técnicas e ferramentas capazes de fornecer sugestões de itens que podem ser do interesse do usuário. Item é um termo utilizado para denotar o que um sistema de recomendação recomendaria para um usuário.

No caso de um *website* de uma livraria, por exemplo, um usuário que lê com frequência pode encontrar com facilidade os livros que são de seu interesse dentre muitas opções disponíveis. No entanto, um usuário com pouca experiência pode utilizar-se de um sistema de recomendação para lhe recomendar livros de que goste em vez de procurar no catálogo do *website*.

Em geral, os benefícios da utilização de sistemas de recomendação vão além de fornecer sugestões de itens para os usuários. Em Ricci et al. (2015) são destacados alguns desses benefícios:

- **Aumento do número de itens vendidos:** Como um sistema de recomendação é capaz de recomendar itens personalizados ao usuário, esses itens possuem uma probabilidade maior de atender às vontades do usuário. Do ponto de vista de um prestador de serviços, a meta principal é aumentar o número de usuários que aceitam recomendações e compram os itens recomendados;
- **Aumento da diversidade de itens vendidos:** Visto que o sistema pode propor novos

itens para o usuário que não sejam itens populares, é possível aumentar a diversidade dos itens vendidos pelo sistema ao recomendar itens menos populares;

- **Melhor entendimento do perfil do usuário:** Para poder fornecer recomendações para os usuários, um sistema de recomendação precisa conhecer o perfil de seus usuários. Esse conhecimento pode então ser utilizado para outros fins, como para campanhas de *marketing* ou administração de estoque.

Para que o sistema de recomendação possa fornecer recomendações para os usuários, são coletadas informações de diversas fontes. De maneira geral, como visto em (RICCI et al., 2015), os dados utilizados por um sistema de recomendação dizem respeito as transações de uma aplicação *web*, mais conhecidas como transações. As transações são os relacionamentos entre usuários e itens.

As transações de uma aplicação *web* contém informações sobre as interações que os usuários tiveram com os itens da aplicação. Para o sistema de recomendação, essas informações são úteis pois demonstram as preferências dos usuários para os itens da aplicação. Em alguns casos, as transações podem conter uma avaliação do usuário sobre um item. Essas avaliações, que são a maneira mais popular de um sistema de recomendação utilizar transações, podem ser obtidas de maneira implícita ou explícita (RICCI et al., 2015). Em transações que obtém avaliações de maneira implícita, o sistema tenta inferir a opinião dos usuários por meio de suas ações no *website*. Quando essas avaliações são obtidas de maneira explícita, o usuário é solicitado a dar sua opinião sobre o item em que está visualizando. De acordo com Schafer et al. (2007), as avaliações explícitas podem tomar as seguintes formas:

- Avaliações numéricas que variam em uma escala. Como um exemplo, as avaliações do *website* da Amazon¹ variam de 1 – 5 estrelas;
- Avaliações ordinais, nas quais o usuário seleciona a opção que melhor descreve sua opinião com relação ao item que está avaliando. Exemplo de opções que o usuário pode selecionar são “concordo fortemente, concordo, neutro, discordo, discordo fortemente”;
- Avaliações binárias, nas quais o usuário apenas escolhe se determinado item é bom ou não, por exemplo, se algum usuário curtiu uma publicação no Facebook²;
- Avaliações unárias, que indicam se o usuário comprou ou observou um item, ou se o item foi avaliado positivamente. Um exemplo de avaliações unárias ocorre em históricos de

¹<https://www.amazon.com>

²<https://www.facebook.com>

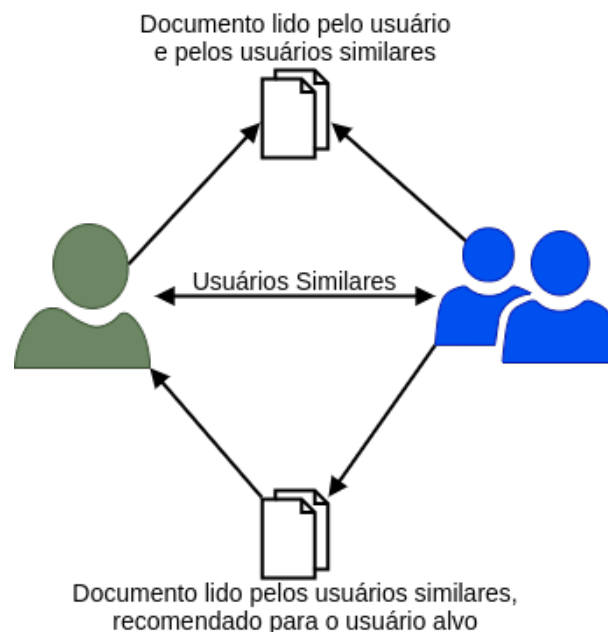
execução de músicas de usuários, caso a música esteja presente, quer dizer que o usuário ouviu aquela música.

Os sistemas de recomendação, de maneira geral, divergem na maneira em que são construídos com relação à como utilizam essas avaliações para recomendar itens. Existem quatro paradigmas, de acordo com Jannach et al. (2010), nos quais podem ser categorizados sistemas de recomendação: Sistemas de Recomendação de Filtragem Colaborativa, Sistemas de Recomendação Baseados em Conteúdo, Sistemas de Recomendação Baseados em Conhecimento e Sistemas de Recomendação Híbridos.

Sistemas de recomendação de filtragem colaborativa têm como objetivo em comum usar informações de comunidade para realizar as recomendações, sem utilizar informações adicionais sobre os itens e os usuários. Seu objetivo, como definido em Jannach et al. (2010), é usar informações sobre os hábitos anteriores de uma comunidade de usuários para predizer para o usuário atual do sistema quais itens ele provavelmente gostará.

A Figura 1 apresenta uma representação dos sistemas de recomendação de filtragem colaborativa, na qual os itens (representados na figura por documentos) que foram lidos tanto pelo usuário alvo quanto pelos seus usuários similares, são utilizados para recomendar itens novos para o usuário alvo.

Figura 1: Sistemas de recomendação de filtragem colaborativa.



Fonte: Autoria própria.

A ideia principal utilizada por sistemas de recomendação de filtragem colaborativa é

que as avaliações feitas por um usuário alvo para um novo item provavelmente serão similares a de outro usuário, caso ambos usuários tenham avaliado itens de maneira similar no passado (NING et al., 2015). Dentro dessa categoria, os sistemas de recomendação podem ser divididos em dois tipos: baseados em memória e baseados em modelo.

Sistemas baseados em memória (também conhecidos como baseados em vizinhança), de acordo com Jannach et al. (2010), aceitam como entrada uma matriz que contém as avaliações que os usuários fizeram para todos os itens, de acordo com o método de avaliação usado pelo sistema. Um exemplo de matriz de avaliações está disponível na Tabela 1, com avaliações de usuários para itens em uma escala de 1 à 5.

Tabela 1: Exemplo de matriz de avaliações.

	Item 1	Item 2	Item 3	Item 4
Usuário 1	3	1	4	-
Usuário 2	2	4	3	2
Usuário 3	4	1	-	1

Fonte: Autoria Própria.

Os sistemas baseados em memória predizem os valores faltantes na matriz de avaliações e, após isso, geram uma lista de N itens recomendados. Tais itens, no entanto, não deverão incluir itens que já foram consumidos pelo usuário.

Em contrapartida, os sistemas baseados em modelo constroem um modelo de aprendizado a partir da matriz de avaliações e, com isso, obtêm um conjunto de características relevantes dos usuários e itens que serão utilizadas posteriormente para prever novas avaliações (NING et al., 2015).

Os dois sistemas de recomendação mais conhecidos dentre os sistemas de filtragem colaborativa são os sistemas *User-based K-Nearest Neighbors* (UserKNN) e *Item-Based K-Nearest Neighbors* (ItemKNN) (JANNACH et al., 2010).

O UserKNN, mais conhecido sistema de recomendação do paradigma de filtragem colaborativa, possui como ideia principal identificar os usuários que possuem preferências similares (vizinhos) ao usuário alvo e então, para cada item p que o usuário alvo não interagiu, calcular uma predição para p com base nas predições feitas pelos seus vizinhos (JANNACH et al., 2010).

O sistema, no entanto, precisa identificar quais são os k usuários que possuem a maior similaridade com o usuário alvo. De acordo com Jannach et al. (2010), uma das medidas de similaridades mais utilizadas para se definir se usuários são similares é o coeficiente de similaridade de Pearson, definido na Equação 1, na qual assume-se P como o conjunto de itens

$P = \{p_1, \dots, p_m\}$; R como uma matriz $n \times m$ de avaliações $r_{i,j}$, com $i \in 1 \dots n$ e $j \in 1 \dots m$; e o símbolo \bar{r}_a como a média das avaliações do usuário a .

$$\text{sim}(a, b) = \frac{\sum_{p \in P} (r_{a,p} - \bar{r}_a) (r_{b,p} - \bar{r}_b)}{\sqrt{\sum_{p \in P} (r_{a,p} - \bar{r}_a)^2} \sqrt{\sum_{p \in P} (r_{b,p} - \bar{r}_b)^2}} \quad (1)$$

Após encontrar os k vizinhos mais similares, a predição é calculada para os itens não vistos pelo usuário alvo. Essa predição em geral é calculada por meio de uma agregação das avaliações dos usuários vizinhos para os demais itens não avaliados pelo usuário alvo (ADOMAVICIUS et al., 2005).

Apesar de ser um dos primeiros sistemas de recomendação de filtragem colaborativa, o UserKNN sofre com problemas de escalabilidade (NING et al., 2015; JANNACH et al., 2010; SARWAR et al., 2001) quando o número de usuários ultrapassa o número de itens disponíveis no sistema, o que se torna comum em aplicações *web* de grande porte. Isso ocorre porque a busca de vizinhos para o usuário alvo pode se tornar muito custosa.

O sistema de recomendação ItemKNN, proposto por Sarwar et al. (2001), cria uma vizinhança de k itens por meio de um cálculo de similaridade entre os itens. A equação que é usada como padrão para o cálculo da similaridade entre itens é a similaridade cosseno, definida na Equação 2, na qual cada item é representado por um vetor contendo todas as avaliações que recebeu.

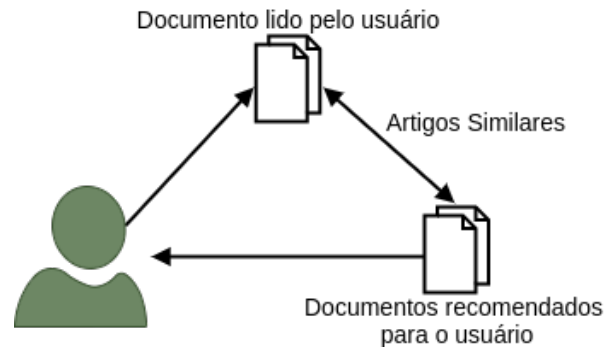
$$\text{sim}(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| * |\vec{b}|} \quad (2)$$

De acordo com Sarwar et al. (2001), a predição do sistema de recomendação ItemKNN para um item i é calculada por meio de uma soma ponderada entre a similaridade entre os itens similares k e as avaliações que o usuário alvo deu para esses itens, dividido pelo soma das similaridades.

Diferentemente dos sistemas de filtragem colaborativa, os sistemas de recomendação baseados em conteúdo realizam a recomendação de itens por meio da similaridade de itens com o perfil do usuário, e não por meio de uma comunidade (RICCI et al., 2015). Esses sistemas analisam as representações dos itens para identificar itens que serão do interesse do usuário, de acordo com o seu perfil.

A Figura 2 apresenta uma representação dos sistemas de recomendação baseados em conteúdo, no qual os documentos lidos pelo usuário são utilizados para recomendar novos

Figura 2: Sistemas de recomendação baseados em conteúdo.



Fonte: A autoria própria.

documentos que o usuário pode gostar.

No caso de um sistema de recomendação baseado em conteúdo de notícias, as palavras-chaves de uma notícia podem ser utilizadas como representação de uma notícia (PAZZANI; BILLSUS, 2007). A partir das representações, Pazzani e Billsus (2007) definem dois tipos de informações que podem ser utilizadas para criar o perfil de um usuário em um sistema de recomendação baseado em conteúdo:

1. Um modelo das preferências do usuário como uma descrição dos tipos de itens que interessam ao usuário;
2. Um histórico das interações do usuário com o sistema de recomendação. Esse tipo de informação inclui tanto os itens que o usuário visualizou, quanto informações de compra de produtos e ou a avaliação que ele deu para tais itens.

De acordo com Bobadilla et al. (2013), são definidos três passos para a construção de um sistema de recomendação baseado em conteúdo:

1. Extrair atributos relevantes dos itens;
2. Comparar tais atributos com as preferências do usuário alvo;
3. Recomendar os itens que sejam mais similares.

A utilização de sistemas de recomendação baseados em conteúdo possui diversas vantagens, como visto em Lops et al. (2011):

- **Independência do Usuário:** Como os sistemas de recomendação baseados em conteúdo fazem suas predições considerando apenas o perfil do usuário, uma aplicação com poucos usuários consegue fazer boas recomendações;

- **Transparência:** Como as recomendações são feitas levando em conta as características do item e do usuário, é possível descrever quais características foram levadas em consideração para que o item tenha sido recomendado ao usuário alvo;
- **Item Recente:** Como o algoritmo não depende da avaliação de outros usuários parecidos em algum item específico, itens que foram adicionados recentemente podem ser recomendados ao usuário devido à similaridade de características.

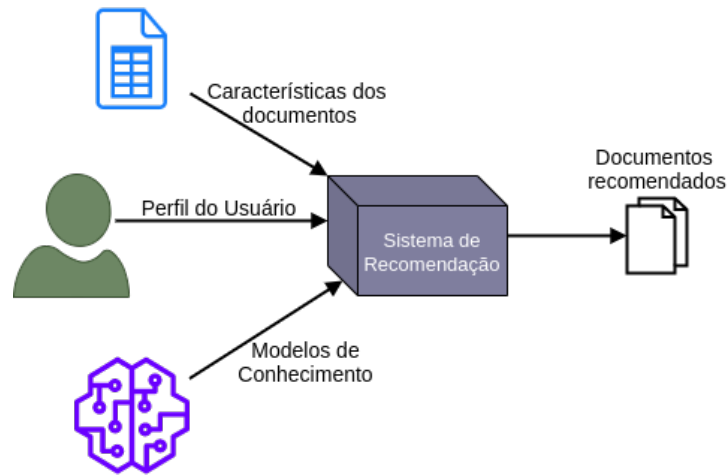
Ainda que um sistema de recomendação baseado em conteúdo possua diversas vantagens, alguns problemas inerentes a sua natureza de recomendação podem ser observados. Ainda em Lops et al. (2011), as seguintes desvantagens são descritas:

- **Análise limitada de conteúdo:** É necessário existir conhecimento prévio do conteúdo no qual o sistema de recomendação será aplicado, visto que um sistema de recomendação baseado em conteúdo não consegue fornecer sugestões boas se o conteúdo analisado não contém informação suficiente;
- **Super especialização:** Esse efeito ocorre porque um sistema de recomendação baseado em conteúdo tende, com o tempo, a recomendar itens sem nenhum grau de novidade para o usuário, limitando o número de itens que podem ser recomendados;
- **Usuário recente.** Um usuário, para obter recomendações boas, precisa ter avaliado outros itens previamente para que um modelo sobre suas preferências (perfil) seja criado.

Já os sistemas de recomendação baseados em conhecimento, descritos em Burke (2000), diferentemente dos sistemas baseados em conteúdo e de filtragem colaborativa, não possuem como entrada o histórico do usuário. Seu papel é recomendar itens com o usuário tendo um papel ativo durante o processo de recomendação, fornecendo de maneira explícita suas preferências para que o sistema de recomendação possa filtrar de maneira adequada de acordo com seus gostos.

Como o usuário deve fornecer de maneira explícita o que deseja do sistema, o banco de dados necessário para a implementação desse tipo de sistema não precisa ser grande. De maneira geral, esse tipo de abordagem é aplicada em domínios nos quais os dados são escassos ou especializados, como na compra de casas, carros ou outros itens de custo elevado. Além disso, itens que não são comprados com frequência também podem necessitar de um sistema baseado em conhecimento, dada à dificuldade para criar um perfil devido à falta de interações do usuário (FELFERNIG et al., 2015).

Figura 3: Sistemas de recomendação baseados em conhecimento.



Fonte: Autoria própria.

Apesar de não necessitarem de uma base de dados muito grande, é necessário o conhecimento prévio do domínio dos itens que serão recomendados para que as interações entre os usuários e o sistema de recomendação sejam efetivas (JANNACH et al., 2010). A Figura 3 apresenta uma representação dos dados de entrada de um sistema de recomendação baseado em conhecimento, que são as características dos documentos, o perfil do usuário, que representa suas preferências, e um modelo de conhecimento.

Os sistemas de recomendação híbridos foram propostos com o intuito de combinar os pontos fortes e suprimir os pontos fracos dos demais paradigmas. Com isso, Jannach et al. (2010) definiu arquiteturas genéricas que podem ser utilizadas para criar esses sistemas híbridos:

- **Arquitetura Monolítica:** A arquitetura monolítica consiste em usar características de diversos sistemas de recomendações em um único sistema de recomendação híbrido. Como um exemplo, um sistema de recomendação baseado em conteúdo pode utilizar, além das informações dos itens, informações de sistemas de filtragem colaborativa, como os usuários parecidos com o usuário atual, para recomendar novos itens;
- **Arquitetura Paralela:** Nessa arquitetura, dois ou mais sistemas de recomendação são utilizados para gerar as recomendações para o usuário e um pós-processamento é utilizado para escolher quais recomendações são mais relevantes. Essa escolha pode ser feita por votação ou utilizando pesos para os resultados dos sistemas de recomendação;
- **Arquitetura em Sequência:** Na arquitetura em sequência, dois ou mais sistemas de recomendação são utilizados em sequência e, a saída de um sistema de recomendação é

utilizada como entrada para o sistema de recomendação subsequente. Como um exemplo, um sistema de recomendação baseado em conteúdo pode escolher os itens mais relevantes para o usuário de acordo com o seu perfil, e suas recomendações podem ser utilizadas em um sistema de recomendação baseado em conhecimento.

Como um exemplo de combinação de sistemas de recomendação para superar os pontos negativos, sistemas de recomendação colaborativos, sofrem com o problema de novos itens e novos usuários, visto que itens que ainda não foram avaliados por usuários tendem a não ser recomendados. Usuários que não fizeram nenhuma avaliação também não conseguem encontrar usuários parecidos. Já os sistemas baseados em conteúdo, similar aos sistemas colaborativos, sofrem com o problema de novos usuários. Porém, diferentemente dos sistemas colaborativos, o problema do novo usuário se dá pois como o usuário ainda não avaliou nenhum item no sistema, é impossível criar um perfil para o mesmo.

Em Adomavicius et al. (2005) são descritas algumas estratégias para combinar sistemas de recomendação baseados em conteúdo e de filtragem colaborativa:

1. Combinar as predições de sistemas de recomendações de filtragem colaborativa e baseados em conteúdo. Isso pode ser feito comparando as predições para descobrir qual tem o maior grau de confiança e, então, usá-la;
2. Incorporar características de sistemas baseados em conteúdo em um sistema de filtragem colaborativa por meio das representações dos usuários obtidas pelo sistema baseado em conteúdo. Para fazer a predição, em vez de utilizar apenas os itens avaliados, utiliza-se as representações dos itens junto com a vizinhança dos itens ou usuários mais similares. Isso pode fazer com que um item seja recomendado não só quando possui uma avaliação alta de outros usuários, mas também quando possuir características semelhantes à do usuário.

2.1.1 SISTEMAS DE RECOMENDAÇÃO SENSÍVEIS AO CONTEXTO

Um sistema de recomendação sensível ao contexto, diferentemente dos sistemas de recomendação tradicionais, utilizam informações contextuais no processo da predição das avaliações dos itens. Conforme Adomavicius et al. (2005), “a predição exata das preferências do consumidor depende, sem dúvida, do grau em que incorporamos a informação contextual relevante no método de recomendação”.

O contexto, como visto em Adomavicius e Tuzhilin (2015), pode ser definido por meio de um conceito diferente em cada área de aplicação. Portanto, muitas definições podem

ser encontradas dentro de cada área, ou até dentro de uma subárea. A seguir, são apresentados exemplos dessas definições:

- **Mineração de Dados:** O contexto, como encontrado na literatura de mineração de dados, foi definido por Palmisano et al. (2008) como eventos que caracterizam a vida de um usuário e que podem determinar uma mudança nas suas preferências, *status* e valor para uma empresa. Exemplo dessa definição são um trabalho novo, casamento, nascimento de um filho, entre outros;
- **Customização de *E-commerce*:** De acordo com Palmisano et al. (2008), a intenção da compra por parte de um consumidor em um *e-commerce* determina uma informação contextual. Como exemplo, um usuário pode comprar um livro para aprender sobre determinado conteúdo ou ferramentas para utilizar em um *hobby*;
- **Recuperação de Conhecimento:** O contexto, quando utilizado em recuperação de informação, tem como foco a sua utilização em problemas de curto-prazo e interesses do usuário no momento específico, como: “encontrar todos os restaurantes em Maringá que servem lanches vegetarianos” (PALMISANO et al., 2008).

A informação contextual que é utilizada pelo sistema de recomendação pode estar presente de diversas formas em uma base de dados. Em Adomavicius et al. (2005), são classificadas em três categorias de acordo com o quanto o sistema de recomendação sabe sobre essas informações:

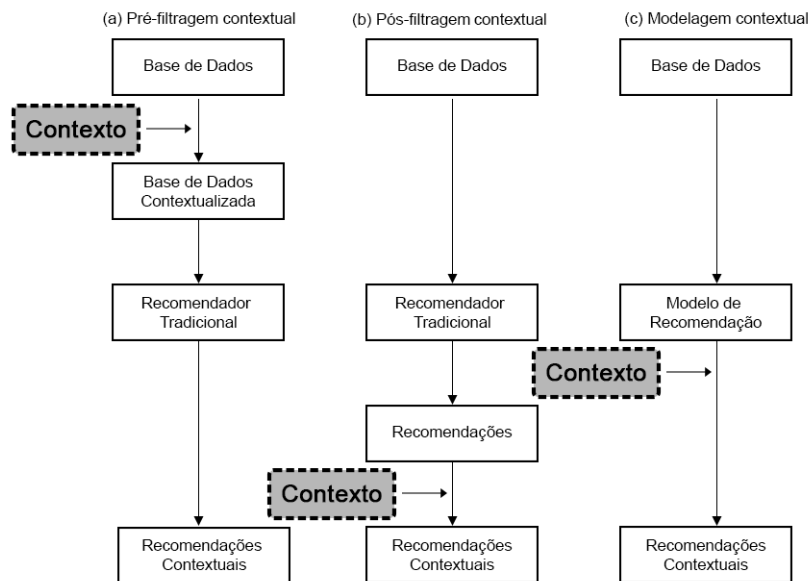
- **Totalmente observável:** Tudo sobre as informações contextuais relevantes para o sistema de recomendação, além de sua estrutura e seus valores, é conhecido quando a recomendação é feita;
- **Parcialmente observável:** Apenas alguns dados sobre as informações contextuais são conhecidos pelo sistema de recomendação, independentemente de qual parte é desconhecida. Se um sistema de recomendação conhece as informações contextuais junto com sua estrutura, mas não seus valores, o mesmo pode ser considerado parcialmente observável;
- **Não observável:** Nenhuma informação contextual é conhecida pelo sistema de recomendação. A recomendação é feita por meio de conhecimento latente adquirido de maneira implícita.

Além do quanto o sistema de recomendação sensível ao contexto sabe sobre a informação, Adomavicius et al. (2005) definiu que essas informações podem variar em uma base de dados, dividindo então em duas categorias: estáticas e dinâmicas.

Sistemas de recomendação nos quais as informações contextuais permanecem estáveis ao longo do tempo, são estáticos. Quando essas informações mudam de acordo com o tempo, eles são chamados de dinâmicos. Como um exemplo, um sistema de recomendação sensível ao contexto de músicas pode perceber ao longo do tempo que a companhia do usuário não influencia o modo no qual ele ouve músicas, e desconsiderar essa informação.

Em Adomavicius e Tuzhilin (2015), os sistemas de recomendação sensíveis ao contexto são subdivididos em três grupos, **pré-filtragem contextual**, **pós-filtragem contextual** e **modelagem contextual**, de acordo com a utilização da informação contextual, como mostrado na Figura 4.

Figura 4: Classificação de sistemas de recomendação sensíveis ao contexto.



Fonte: Imagem adaptada de Adomavicius e Tuzhilin (2015).

O sistema de recomendação sensível ao contexto que se utiliza de uma abordagem de **pré-filtragem contextual** faz o uso da informação contextual já em uma primeira fase, na base de dados, para então utilizar um sistema de recomendação tradicional nesses dados com as informações de contexto.

Na **pós-filtragem contextual**, como visto na Figura 4, uma filtragem é aplicada após a recomendação ter sido realizada por um sistema de recomendação tradicional. Em Adomavicius

e Tuzhilin (2015), esse método pode ser classificado de duas maneiras: baseado em heurísticas e baseado em modelo. Os métodos baseados em heurísticas pretendem encontrar atributos de um usuário em algum contexto, e então utilizar esses atributos para ajustar as recomendações, seja filtrando itens que foram recomendados que não possuem essas características ou ordenando esses itens com base no número de características que possuem. Para os métodos baseados em modelo, modelos preditivos são construídos para calcular a probabilidade de que o usuário escolheria aquele item em um determinado contexto, usando então a probabilidade de relevância para ajustar as recomendações. Isso pode ser feito filtrando itens que tem uma probabilidade de relevância menor que um limite pré-definido ou ordenando esses itens com base na probabilidade de relevância que eles possuem.

Os sistemas de recomendação de **modelagem contextual** usam a informação contextual diretamente no método de recomendação para recomendar os itens (ADOMAVICIUS; TUZHILIN, 2015). Diferentemente das outras abordagens, a abordagem de modelagem contextual dá origem a diversos modelos preditivos multidimensionais (construídos utilizando árvores de decisão, modelos regressores, modelos probabilísticos, etc) ou cálculos heurísticos que incorporam essa informação contextual aos dados de usuário e ao item.

Neste trabalho, foram utilizados quatro sistemas de recomendação sensíveis ao contexto: Music2vec-TopN (M-TN), Session-Music2vec-TopN (SM-TN), Context-Session-Music2vec-TopN (CSM-TN) e Context-Session-Music2vec-UserKNN (CSM-UK). Esses sistemas de recomendação podem ser classificados como sistemas de recomendação sensíveis ao contexto no paradigma de pré-filtragem contextual, pois os dados de entrada já possuem informação contextual. Esses sistemas de recomendação serão apresentados na Subseção ??.

2.2 EMBEDDINGS

Diversos tipos de dados, como imagens e músicas, podem ser modelados por meio de sinais analógicos para serem utilizados por algoritmos e redes neurais. Uma música, por exemplo, pode ser modelada por meio de espectogramas. Porém, dados textuais não podem ser processados em sua forma original, necessitando de uma representação que seja adequada para o computador.

Representar palavras como um vetor de números reais tem sua origem nos estudos das representações distribuídas, como pode ser visto no trabalho de Rogers e McClelland (2014). No caso das representações distribuídas, um item é representado por um padrão de atividades em um conjunto de elementos computacionais (neurônios, no caso de redes neurais), sendo que

cada elemento está envolvido na representação de diferentes itens (ROGERS; MCCLELLAND, 2014).

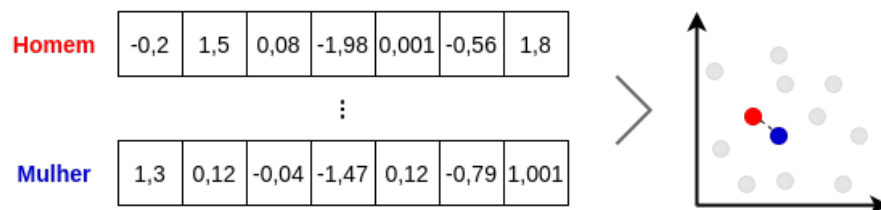
Para se obter esse vetores, conhecidos também como *word embeddings*, foram utilizadas diferentes abordagens. As primeiras abordagens que foram exploradas tiveram como base modelos latentes para obtenção de embeddings, entre elas, o modelo de Deerwester et al. (1990), chamado de *Latent Semantic Analysis* (LSA), utiliza fatoração de matrizes para a obtenção de *word embeddings*. Um outro modelo, proposto posteriormente por Blei et al. (2003), utiliza um modelo probabilístico para a obtenção dos *embeddings*.

Os primeiros modelos para obtenção de *word embeddings* que utilizam redes neurais, de acordo com Li et al. (2018), foram os modelos de Xu e Rudnicky (2000) e Bengio et al. (2003), conhecidos como Modelos de Redes Neurais Linguísticas. A motivação por trás desses modelos é de que palavras são mais suscetíveis a compartilharem sentidos se aparecem perto umas das outras (LI et al., 2018).

Apesar dos modelos de Xu e Rudnicky (2000) e Bengio et al. (2003), o trabalho que popularizou o uso de *word embeddings* com redes neurais foi o trabalho proposto por Mikolov et al. (2013a), chamado word2vec, que é descrito na subseção a seguir.

A Figura 5 apresenta um exemplo de representação de palavras como *embeddings*. Neste exemplo, duas palavras alvo (homem e mulher), são transformadas em *embeddings* e, a partir de sua representação vetorial, é possível visualizá-las em um gráfico. Quando palavras são transformadas em vetores, é possível utilizar operações de vetores nas mesmas, como o cálculo de similaridade entre vetores. Isso permite que elas sejam usadas em diversas áreas de pesquisa.

Figura 5: Exemplo de representação de palavras como *embeddings*.



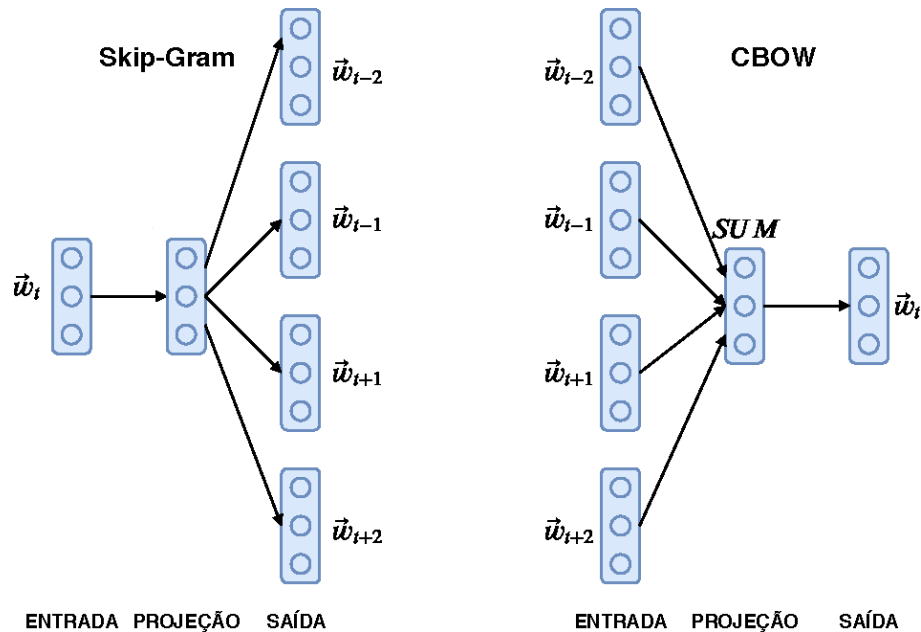
Fonte: Autoria própria.

2.2.1 WORD2VEC

O word2vec é composto por dois modelos de redes neurais linguísticas, *Continuous Skip-Gram* e *Continuous Bag-of-Words* (CBOW), ilustrados na Figura 6, sendo que ambos são

utilizados para a obtenção de *word embeddings*, porém cada um com um propósito diferente. O modelo de *Skip-Gram* possui como entrada uma palavra alvo, e produz como saída as palavras que estão ao redor dessa palavra alvo. Já no modelo de CBOW, as palavras ao redor dessa palavra alvo são utilizadas como entrada com o objetivo de prever qual a palavra alvo.

Figura 6: Modelos *Skip-Gram* e CBOW propostos por Mikolov et al. (2013a).



Fonte: Autoria própria.

O objetivo do modelo *Skip-Gram*, como definido por Mikolov et al. (2013b), é encontrar representações de palavras que são eficazes para prever as palavras adjacentes em uma frase ou em um documento. De uma maneira formal, dada uma sequência de palavras de treino $w_1, w_2, w_3, \dots, w_T$, o objetivo do modelo *Skip-Gram* é maximizar a probabilidade logarítmica média,

$$\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t), \quad (3)$$

na qual c corresponde ao tamanho da janela contextual. A janela contextual é uma janela que desliza por todo o texto, obtendo as palavras adjacentes da palavra alvo. A Figura 7 exemplifica o funcionamento da janela contextual, utilizando como exemplo uma janela contextual de tamanho 2.

Em ambos os modelos, CBOW e *Skip-Gram*, quando o vetor atribuído para uma palavra de entrada (ou vetores, para o CBOW) é utilizado no processo de aprendizado para prever uma saída, os elementos desse vetor são ajustados pela rede neural dado o quão similar

a saída esperada foi a saída predita. Por meio desse ajuste dos elementos do vetor, os vetores de palavras que são consideradas similares, como visto na Figura 5, se tornam mais próximas no espaço vetorial, enquanto as mais diferentes acabam por ficar mais longe.

No exemplo da Figura 7, a janela desliza por toda a frase obtendo as palavras adjacentes que estão antes e depois da palavra alvo, quando possível. Para a palavra “*roupa*”, destacada dentro do retângulo, a janela contextual possuiria as palavras “roeu, a, do, rei”.

Figura 7: Exemplo de janela contextual de tamanho 2.

O rato roeu a roupa do rei de Roma.

W_{t-2}	W_{t-1}	Alvo	W_{t+1}	W_{t+2}
-	-	O	rato	roeu
-	O	rato	roeu	a
O	rato	roeu	a	roupa
rato	roeu	a	roupa	do
⋮				
rei	de	Roma	-	-

Fonte: Autoria própria.

A formulação básica do *Skip-Gram*, definida por Mikolov et al. (2013b), define $p(w_{t+j}|w_t)$ usando a função *softmax*,

$$p(w_o|w_I) = \frac{\exp(v'_{w_o} v_{w_I})}{\sum_{w=1}^W \exp(v'_w v_{w_I})}, \quad (4)$$

no qual v_w e v'_w são as representações vetoriais da palavra de entrada e de saída, e W é o número de palavras no vocabulário. Devido ao custo computacional de calcular $\log p(w_o|w_I)$ para o vocabulário W , outras estratégias são consideradas. Uma maneira computacionalmente eficiente é usar a técnica de *Negative Sampling*, uma adaptação da técnica de *Noise Contrastive Estimation* (NCE) proposta por Gutmann e Hyvärinen (2010).

A técnica NCE, apesar de ser capaz de obter uma aproximação de uma maximização da probabilidade logarítmica da função *softmax*, possui características que não interessam ao *Skip-Gram* (MIKOLOV et al., 2013b). Por isso, a técnica de *Negative Sampling* pode ser determinada como uma simplificação da técnica NCE, e é definida da seguinte forma:

$$\log \sigma(v'_{w_o} v_{w_I}) + \sum_{i=1}^k \mathbb{E}_{w_i \sim P_n(w)} \left[\log \sigma(-v'_{w_i} v_{w_I}) \right]. \quad (5)$$

A Equação 5 então substituirá todos os termos $\log p(w_o|w_I)$ na função objetivo

(Equação 3).

O segundo modelo proposto por Mikolov et al. (2013a), *Continuous Bag-of-Words* (CBOW), é similar a uma arquitetura de rede neural linguística simples. Sua principal diferença para o modelo de *Skip-Gram*, como pode ser visto na Figura 6, está presente em como o modelo foi definido para a obtenção dos *word embeddings*.

No *Skip-Gram*, o modelo tem como entrada uma única palavra e tem como objetivo prever as palavras que estão em sua janela contextual, enquanto que no modelo CBOW a entrada do modelo são as palavras que estão na janela contextual, com o objetivo de prever a palavra que está no centro da janela. O modelo CBOW possui esse nome visto que, assim como a técnica *Bag-of-Words*, a ordem das palavras da janela contextual não influencia na predição da palavra central.

O sucesso dos modelos propostos por Mikolov et al. (2013a) se dá não só por sua simplicidade e capacidade de serem executados em conjuntos de dados muito grandes, mas porque os *word embeddings* gerados pelos modelos são capazes de encontrar estruturas e relacionamentos semânticos das palavras.

Os vetores aprendidos com os modelos do word2vec em conjuntos com uma grande quantidade de dados, por exemplo, são capazes de aprender relacionamentos semânticos sutis entre palavras, como o de uma cidade e qual país a cidade pertence, como, por exemplo, França está para Paris assim como Alemanha está para Berlim.

2.3 REDES NEURAIIS RECORRENTES

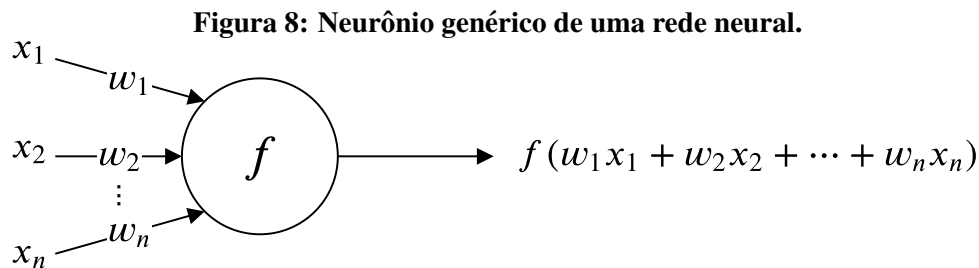
O desenvolvimento de trabalhos e pesquisas com Redes Neurais Artificiais foram motivados pelo reconhecimento de que o cérebro humano funciona de forma diferente dos computadores. O cérebro pode ser visto como um computador paralelo, não-linear e altamente complexo, capaz de utilizar seus componentes estruturais, ou “neurônios”, para realizar algumas tarefas (reconhecimento de padrões, controle de motores, etc) de maneira mais rápida que computadores comuns (HAYKIN, 2008).

De acordo com Haykin (2008), uma rede neural é utilizada para modelar o modo com o qual o cérebro realiza uma tarefa ou função de interesse. Para que as redes neurais consigam bons desempenhos, elas utilizam de uma grande quantidade de interconexões entre neurônios para produzir resultados. A definição de uma rede neural artificial, sugerida por Haykin (2008), é a seguinte:

Uma rede neural é um imenso processador paralelo distribuído construído de pequenos neurônios que possuem uma propensão natural a guardar conhecimento experimental e torná-lo disponível para uso. Ela assemelha-se ao cérebro de duas maneiras:

1. O conhecimento é adquirido pela rede por meio do seu ambiente em um processo de aprendizado;
2. O peso da conectividade entre os neurônios, chamado de pesos sinápticos, são usados para guardar o conhecimento adquirido.

A Figura 8 apresenta uma representação de um neurônio genérico de uma rede neural, com suas entradas e sua saída. Um neurônio pode ter várias entradas i , sendo que seus valores de entrada são representados por um valor x_i . A função f , que compreende o “corpo” do neurônio, é selecionada arbitrariamente de acordo com o problema. Os valores de entrada do neurônio possuem um peso associados a si, significando que o valor de entrada x_i é multiplicado por um peso w_i associado ao valor. Esses valores são então integrados ao neurônio por meio de uma função de agregação e a função do neurônio é então avaliada. O resultado da função de ativação do neurônio é passado para frente na rede neural e é chamado de valor de ativação.



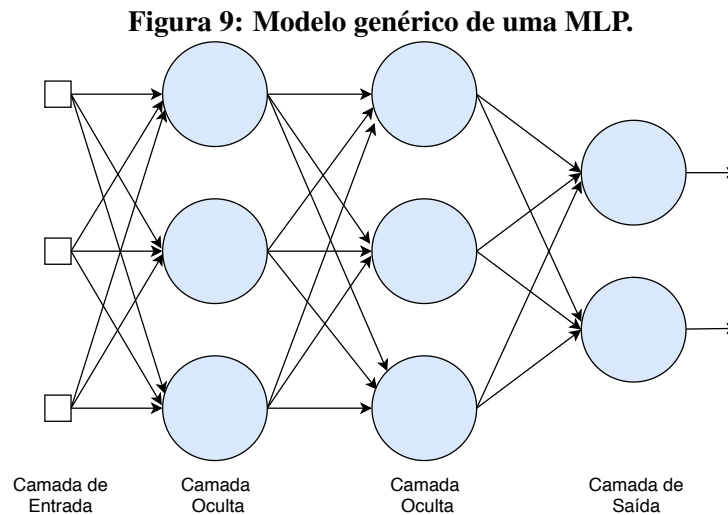
Fonte: Adaptado de Rojas (1996).

Como visto em Rojas (1996), se pensarmos em cada nó de uma rede neural como sendo uma função capaz de transformar suas entradas em uma saída, então redes neurais nada mais são que redes de funções. Os diferentes modelos de redes neurais disponíveis na literatura diferem principalmente nas escolhas feitas sobre as funções utilizadas, o padrão de interconexão entre os neurônios e como a transmissão da informação é realizada.

Um dos modelos mais proeminentes da literatura quando se trata de redes neurais, é o modelo *MultiLayer Perceptron* (MLP), proposto por Rumelhart e MacClelland (1986). Também chamado de *feedforward neural networks*, o objetivo do modelo MLP é aproximar uma função f^* tal que para um classificador $y = f^*(x)$, ele mapeie uma entrada x para uma categoria y . O objetivo de uma MLP é definir um mapeamento $y = f(x; \theta)$ e aprender o valor dos parâmetros θ que resulte na melhor aproximação da função (GOODFELLOW et al., 2016).

O modelo MLP, no entanto, pode variar com relação ao número de camadas que possui. Sua primeira camada, a **camada de entrada**, receberá os dados de entrada e os direcionará

para as **camadas ocultas** da rede neural. O número de camadas ocultas de uma rede MLP pode variar, sendo que o número de camadas determina a profundidade do modelo. O termo *deep learning* (aprendizado profundo) surge dessa terminologia. A camada final do modelo é a **camada de saída**, no qual os resultados serão fornecidos pela rede após todo o processo de aprendizado. A Figura 9 ilustra um modelo de MLP que possui duas camadas ocultas com três neurônios em cada camada.



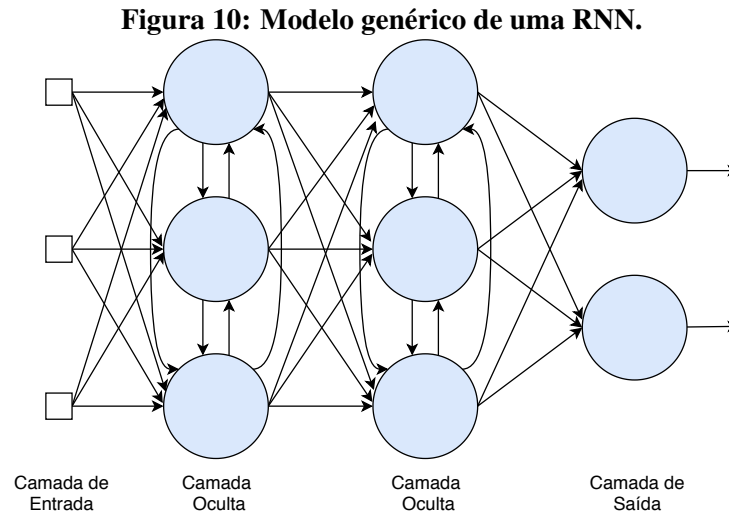
Fonte: Autoria própria.

Assim como posto na definição de Haykin (2008), uma rede neural aprende por meio de um processo de aprendizado. O método de aprendizado utilizado para treinar MLPs, difundido por Rumelhart e MacClelland (1986), é chamado de *back-propagation*, e é composto de duas fases:

1. Na primeira fase, conhecida como fase *forward*, os pesos das conexões com os neurônios são atualizados, fixados e o sinal de entrada é propagado pela rede, avançando da camada de entrada até a camada de saída.
2. Na segunda fase, chamada de fase *backward*, é produzido um sinal de erro por meio de uma comparação da saída esperada da rede com o resultado obtido. Esse sinal é propagado em sentido contrário, ou seja, da camada de saída para a camada de entrada, camada por camada, com o objetivo de atualizar os pesos das conexões dos neurônios.

O modelo *Multilayer Perceptron*, assim como outros modelos de redes *feedforwards*, são modelos que tentam definir uma aproximação de uma função $y = f^*(x)$. Isso impõe uma forte restrição nos modelos, visto que esses modelos são capazes de mapear um dado x de entrada para um resultado de saída y (GRAVES, 2012).

Para que seja possível mapear uma sequência de dados, no entanto, é necessário alterar o modelo das redes *feedforwards* para permitir que exista conexões cíclicas entre neurônios da mesma camada. Redes que possuem essas conexões cíclicas deixam de se chamar redes *feedforward* e passam a se chamar Redes Neurais Recorrentes (RNNs). Na Figura 10, é possível notar as conexões entre os neurônios da mesma camada.



Fonte: Autoria própria.

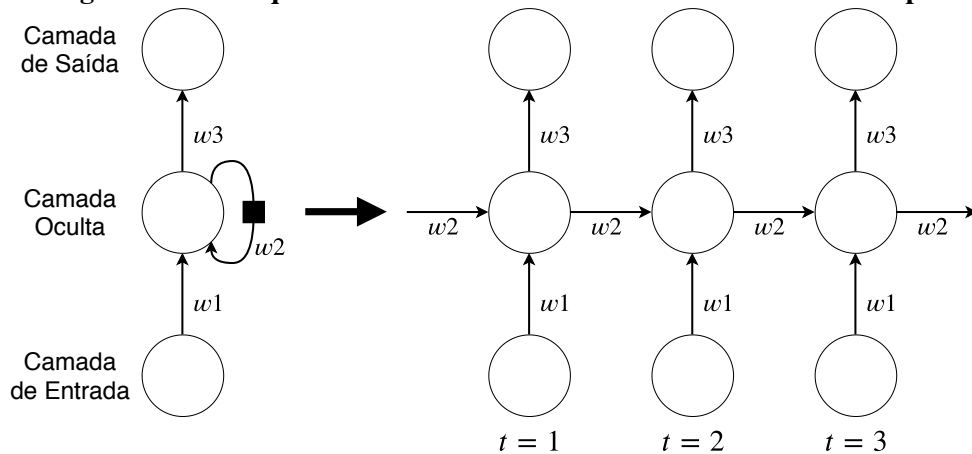
As RNNs tiram proveito do compartilhamento de parâmetros em diferentes partes de seu modelo, o que torna possível estender e aplicar seu modelo para sequências de diferentes tamanhos (GOODFELLOW et al., 2016). Seu principal benefício vem de suas conexões recorrentes, que criam uma “memória” das entradas anteriores que são persistidas em seu estado interno, e influenciam a saída da rede (GRAVES, 2012).

Uma maneira mais intuitiva de visualizar RNNs é considerar as redes estando em um formato “desdobrado”, ou seja, sem ciclos. Na Figura 10, é possível ver ciclos entre conexões de uma mesma camada. Considerando que uma RNN opera em uma sequência no qual o índice de tempo t varia de 1 à T , desdobrar um modelo significa representá-lo de uma maneira em que não estejam presentes ciclos. A Figura 11 mostra uma RNN que foi desdobrada.

Na Figura 11, pode ser visto o mesmo modelo de RNN porém em duas representações: a representação a esquerda possui a camada oculta com a conexão cíclica (o quadrado preto simboliza um intervalo de tempo de valor 1 na sequência). Na representação a direita, para cada índice de tempo na sequência, o modelo é representado por completo sem ciclos, indicando que o valor é passado de um índice de tempo para o outro por meio de uma conexão entre os dois.

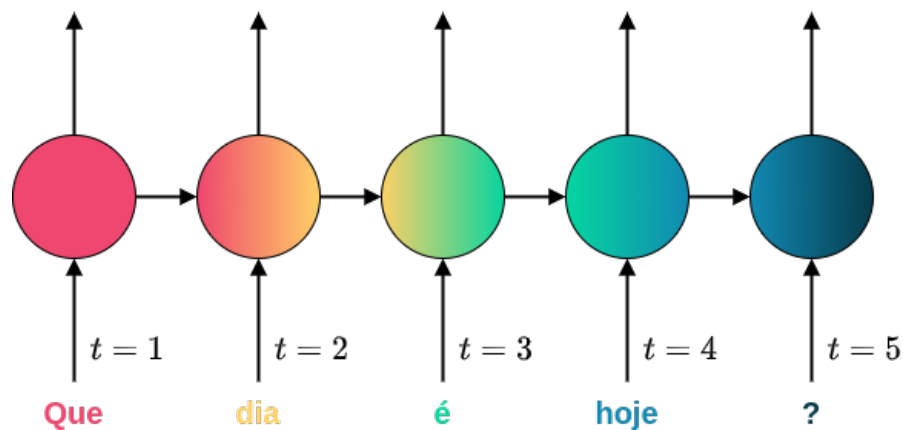
Ainda na Figura 11, o peso das conexões entre a camada de entrada e a camada oculta estão denotadas por “ w_1 ”, o peso das conexões entre as camadas ocultas (pesos recorrentes)

Figura 11: RNN que foi desdobrada de acordo com os índices de tempo.



Fonte: Adaptado de Graves (2012) e Goodfellow et al. (2016).

Figura 12: Exemplo da atuação de uma RNN com uma frase.



Fonte: Autoria própria.

estão denotados por " w_2 " e os pesos das conexões entre a camada oculta e camada de saída estão denotados por " w_3 ". Esse modelo simplificado de representação assume que cada nó de uma camada por compreender n neurônios no modelo real, por exemplo, se a rede da Figura 11 possuísse 256 neurônios na camada oculta, seria usado um único neurônio para representar todo o conjunto.

Como pode ser visto no exemplo da Figura 12, em cada um dos índices de tempo a RNN processa não só a entrada do índice de tempo atual, mas também o resultado do processamento do índice de tempo anterior. Esse compartilhamento de parâmetros permite que a RNN consiga processar sequências de tamanhos variados. No exemplo, ao chegar no índice de tempo $t=3$, a rede possui informação dos índices de tempo 1 e 2, o que permite acumular conhecimento.

Dessa maneira, uma rede neural treinada para conversar com pessoas, como um *Chatbot*, consegue entender no índice de tempo $t = 5$ que o usuário esteja fazendo uma pergunta sobre qual o dia atual.

Assim como no modelo MLP, as RNNs aprendem por meio de uma fase *forward* e uma fase *backward*. De acordo com Graves (2012), a fase *forward* de uma RNN é similar a fase *forward* de um modelo MLP com uma única camada oculta, exceto pelo fato de que os valores de ativação da camada oculta são provenientes tanto da camada de entrada quanto da camada oculta do último índice de tempo.

Considerando uma RNN com I neurônios de entrada, H neurônios ocultos e K neurônios de saída que processa uma sequência de entrada x de tamanho T . Seja x_i^t o valor da entrada i no tempo t , e que a_j^t e b_j^t sejam respectivamente a entrada do neurônio j no tempo t e a ativação do neurônio j no tempo t . Como definido em (GRAVES, 2012), para os neurônios ocultos temos a seguinte entrada:

$$a_h^t = \sum_{i=1}^I w_{ih} x_i^t + \sum_{h'=1}^H w_{h'h} b_{h'}^{t-1}, \quad (6)$$

que corresponde ao somatório ponderado dos valores de cada neurônio de entrada somado ao somatório dos valores ponderados de cada neurônio da camada oculta do índice de tempo anterior. O valor de entrada a_j^t é passado para uma função de ativação θ não linear e diferenciável. Exemplos de funções de ativação para RNNs são as funções sigmóide e tangente hiperbólica. O valor de ativação é definido por Graves (2012) da seguinte forma:

$$b_h^t = \theta_h(a_h^t). \quad (7)$$

A sequência completa de ativações dos neurônios ocultos pode ser calculada começando com o valor de $t = 1$ e, então, aplicar as Equações 6 e 7 recursivamente, incrementando o valor de t (GRAVES, 2012). Essa aplicação recursiva começando em $t = 1$ requer que o valor de b_i^0 seja escolhido previamente, o que corresponde ao estado da rede antes de receber qualquer informação. Os valores de entrada dos neurônios da camada de saída podem ser calculados então utilizando os valores de ativação das camadas ocultas, como definido em (GRAVES, 2012):

$$a_k^t = \sum_{h=1}^H w_{hk} b_h^t. \quad (8)$$

Na camada de saída, podem ser utilizadas funções de ativação que são comuns ao modelo de MLP como a função logística e a função *softmax* (GRAVES, 2012). Para determinar o quão correta a saída da rede está, é necessário utilizar uma função de perda \mathcal{L} (GOODFELLOW et al., 2016). Assim, como o algoritmo da fase *forward*, o algoritmo responsável pela fase *backward* de uma RNN é similar ao de uma MLP. Para realizar a fase de *backward*, o algoritmo *Back-Propagation Through Time* (BPTT), proposto por Werbos (1990), consiste em atualizar os pesos das conexões entre neurônios de acordo com a função de perda \mathcal{L} .

O algoritmo BPTT, similar ao algoritmo *back-propagation*, usa o método de gradiente descendente para encontrar a derivada da função de perda em razão ao peso das conexões dos neurônios, para então ajustar os pesos na direção de uma inclinação negativa (GOODFELLOW et al., 2016).

Assim como o algoritmo *back-propagation* comum, o algoritmo BPTT consiste em aplicar a regra da cadeia repetidamente, começando da camada de saída da RNN e indo em direção a camada de entrada. Por se tratar de uma RNN, a função de perda \mathcal{L} depende do valor de ativação da camada oculta não só pela sua influência na camada de saída, mas também por sua influência na camada oculta no próximo índice de tempo. Para se obter os termos de um neurônio da camada oculta de uma RNN, utiliza-se a seguinte equação definida por Graves (2012):

$$\delta_h^t = \theta'(a_h^t) \left(\sum_{k=1}^K \delta_k^t w_{hk} + \sum_{h'=1}^H \delta_{h'}^{t+1} w_{hh'} \right), \quad (9)$$

dado que δ pode ser definido pela seguinte equação:

$$\delta_j^t \stackrel{\text{def}}{=} \frac{\partial \mathcal{L}}{\partial a_j^t}. \quad (10)$$

A sequência completa de termos pode ser calculada aplicando a Equação 9 de maneira recursiva, com o valor inicial de $t = T$, decrementando o valor de t a cada índice de tempo. Como no exemplo apresentado na Figura 12, se a rede não conseguisse a saída desejada ao final da frase, os pesos seriam atualizados para que o treinamento continuasse. Sabendo então que os termos são reutilizados a cada índice de tempo, toda a sequência é somada com o objetivo de se obter as derivadas em relação aos pesos da rede, para que os pesos possam ser atualizados:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \sum_{t=1}^T \frac{\partial \mathcal{L}}{\partial a_j^t} \frac{\partial a_j^t}{\partial w_{ij}} = \sum_{t=1}^T \delta_j^t b_i^t. \quad (11)$$

A partir dos processos *forward* e BPTT, uma RNN é capaz de realizar o processo de aprendizado. De acordo com Goodfellow et al. (2016), existem arquiteturas distintas para se construir uma RNN, dentre as quais:

- Redes que produzem uma saída a cada índice de tempo;
- Redes que produzem uma saída apenas no último índice de tempo;
- Redes que possuem apenas um valor de entrada, e que utilizam as saídas do índice de tempo atual como a entrada do próximo índice de tempo.

Durante o treinamento de RNNs, no entanto, um problema observado por Bengio et al. (1994) mostrou que tais redes sofrem com dependências de longo prazo ao passar pelo processo de aprendizado. Como um exemplo de dependência de longo prazo, considere a seguinte frase: “Cresci na França e só me mudei após meus 25 anos, por isso falo francês fluentemente”. Em uma RNN que tenta prever as próximas palavras de uma frase, para prever a palavra “francês”, no final da frase, a RNN precisaria de informações da palavra “França”, que aparece no início da frase. Essa dependência da palavra “francês” para a palavra “França” pode ser vista como uma dependência de longo prazo.

Quando essa dependência de longo prazo ocorre em uma RNN, caso os termos (definidos na Equação 10) assumam um valor muito pequeno, eles encolherão exponencialmente até desaparecerem praticamente, impossibilitando o aprendizado da rede. Por outro lado, caso eles assumam um valor muito grande, esses valores assumirão um valor muito grande, fazendo com que a rede neural não consiga aprender de maneira correta (PASCANU et al., 2013).

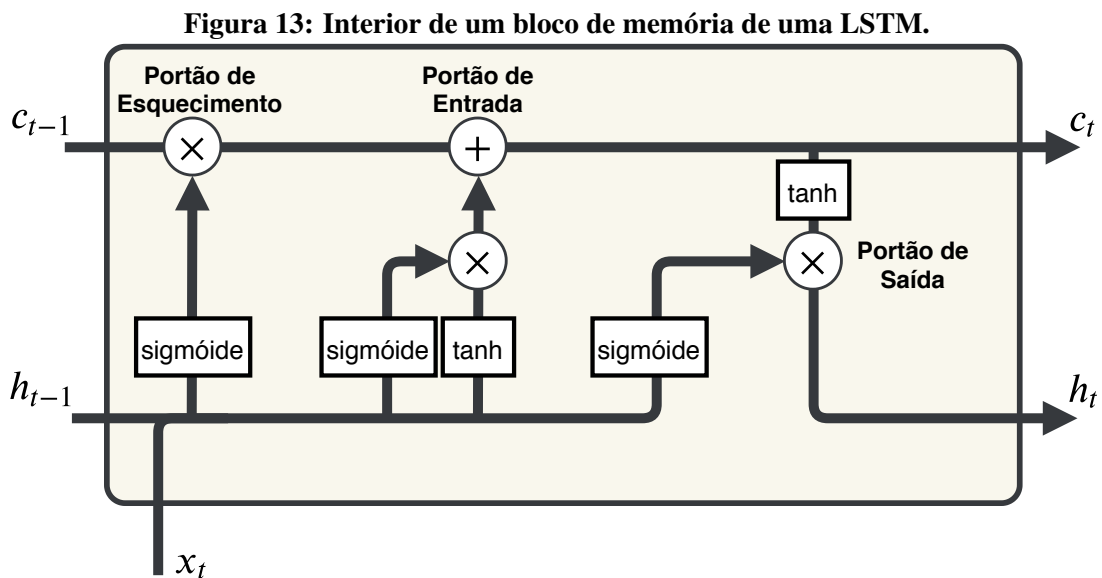
2.3.1 REDES LONG SHORT-TERM MEMORY

Com o objetivo de resolver o problema de dependência de longo prazo em RNNs, Hochreiter e Schmidhuber (1997) propuseram um modelo de rede chamado de redes *Long Short-Term Memory* (LSTM), capaz de resolver esse problema por meio de portões de informação, que são capazes de guardar e esquecer informações que são relevantes para a rede em determinados períodos de tempo.

Na frase de exemplo da Figura 12, “Que dia é hoje?”, uma RNN simples poderia perder o contexto do que está sendo perguntado, pois ao chegar no último índice de tempo, as informações acumuladas pela rede neural poderiam ter se perdido.

A arquitetura de uma rede LSTM, como definida por Graves (2012), consiste de um conjunto de sub-redes recorrentes conectadas umas as outras, conhecidas como blocos de memória. Cada bloco de uma rede LSTM possui três “portões”, que são utilizados para controlar o fluxo de informações que será lido pelo bloco de memória. O modelo de um bloco de memória de uma LSTM pode ser visto na Figura 13.

O processo de treinamento de uma rede LSTM é similar ao processo de treinamento de uma rede RNN apresentado anteriormente, seguindo o modelo de *forward* e *backward* e utilizando o algoritmo de *Back-Propagation Through Time*, no qual os gradientes são calculados para atualizar os pesos das conexões entre os neurônios (no caso da LSTM, dos blocos de memória) (GOODFELLOW et al., 2016).



Fonte: Autoria própria.

Dentro do bloco de memória LSTM, existem dois fluxos de dados, como visto na Figura 13. O primeiro fluxo de dados, representado pela flecha mais acima da imagem, é chamado de estado do neurônio (denotado por c_t). Esse estado do neurônio representa a memória da LSTM, e carrega as informações relevantes durante o processamento da sequência. No bloco de memória, os diferentes portões decidem que tipo de informações são adicionadas ao estado do neurônio.

O segundo fluxo de dados (denotado por h_t), chamado de estado oculto da LSTM, é responsável por produzir as saídas de cada bloco de memória. Assim como o estado do

neurônio, o estado oculto também é passado para o bloco de memória do próximo índice de tempo (GRAVES, 2012).

São utilizadas duas funções principais dentro do bloco de memória que, em geral, são utilizadas em outras redes neurais como função de ativação: função sigmóide e tangente hiperbólica. Como visto em Sutskever (2013), a função sigmóide é utilizada quando deseja-se que os valores de entrada possuam uma variação entre 0 e 1. Esse comportamento pode ser útil quando se deseja **esquecer** ou **manter** alguma informação relevante para a rede. Já a função tangente hiperbólica é utilizada para regular esses valores, mantendo esses valores sempre -1 e 1 , impedindo que os valores cresçam ou diminuam demais.

O Portão de Esquecimento é responsável por decidir quais informações serão esquecidas no índice de tempo atual, dado o estado oculto do índice de tempo anterior e a entrada do índice de tempo atual. Esses vetores são concatenados e passados para uma função sigmóide. O vetor resultante da função sigmóide atualizará o estado do neurônio por meio de um produto vetorial (SUTSKEVER, 2013). Na frase de exemplo, “Que dia é hoje?”, talvez a rede possa encarar que a palavra “é” não seja relevante para o processamento da frase. Com isso, informações sobre essa palavra serão passadas para o Portão de Esquecimento.

O Portão de Entrada, como o próprio nome diz, é responsável pela adição de informação no estado do neurônio. Assim como o Portão de Esquecimento, o estado oculto do índice de tempo anterior e a entrada do índice de tempo atual são passados por uma função sigmóide para decidir converter os valores dos vetores entre 0 e 1. Esse vetor também será passado pela função tangente hiperbólica para que sejam normalizados e, após a aplicação dessas duas funções, uma operação de produto vetorial será aplicada, para que seja decidido que informação será importante manter no estado do neurônio. Esse vetor resultante terá seus elementos somados com o vetor do estado do neurônio.

O último portão, Portão de Saída, é responsável por definir qual será a saída do bloco de memória. O primeiro passo é aplicar a função sigmóide no vetor concatenado do estado oculto do índice de tempo anterior e da entrada do índice de tempo atual. O próximo passo é aplicar então a função tangente hiperbólica para regularizar o resultado do estado do neurônio já atualizado. Por fim, será aplicado o produto vetorial nesses dois vetores para produzir a saída do bloco de memória, que pode ser usado também como a predição do bloco.

Por meio desse mecanismo de portões e das operações multiplicativas, é possível para as LSTMs aprenderem e armazenarem informação por longos períodos de tempo, mitigando o problema do *vanishing gradient* (GRAVES, 2012). O problema de *vanishing gradient* ocorre quando os pesos das conexões da rede neural se tornam muito pequenos e, com isso, se torna

cada vez mais difícil para a rede neural atualizar os erros nos próximos períodos de tempo.

Na frase de exemplo, a LSTM pode decidir por meio do seu treino que as palavras “Que” e “é” não são relevantes para o resultado final, mas que as palavras “dia”, “hoje” e “?” são importantes para definir que uma pergunta sobre o dia atual está sendo feita.

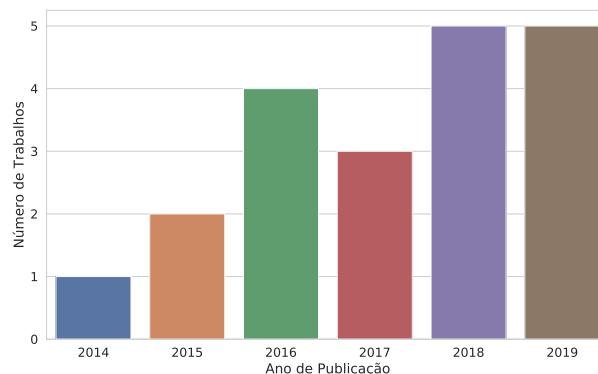
O presente trabalho de mestrado utiliza Redes Neurais Recorrentes para extrair *embeddings* da sequência de músicas que o usuário ouviu e, após essa extração de *embeddings*, as utiliza em sistemas de recomendação sensíveis ao contexto. Os conceitos apresentados foram utilizados tanto como base de pesquisa para os trabalhos relacionados, apresentados na Seção 3, quanto para o desenvolvimento deste trabalho.

3 TRABALHOS RELACIONADOS

Nesta seção são apresentados os trabalhos encontrados na literatura que estão relacionados com o escopo desta dissertação de mestrado. Com o objetivo de encontrar trabalhos de diversos domínios de aplicação que se relacionem com esse escopo, que consiste em adquirir *embeddings* que serão utilizados em sistemas de recomendação sensíveis ao contexto, uma revisão sistemática foi realizada. O protocolo da revisão sistemática que resultou na seleção de 20 trabalhos pode ser encontrado no Apêndice A.

Os trabalhos obtidos na revisão sistemática por meio do protocolo de pesquisa foram todos publicados após o trabalho de Mikolov et al. (2013a), mesmo sem nenhuma restrição de pesquisa no protocolo, o que ressalta a importância do word2vec na área de *word embeddings*. Na Figura 14 é possível ver a evolução no número de trabalhos publicados por ano.

Figura 14: Número de trabalhos publicados por ano.

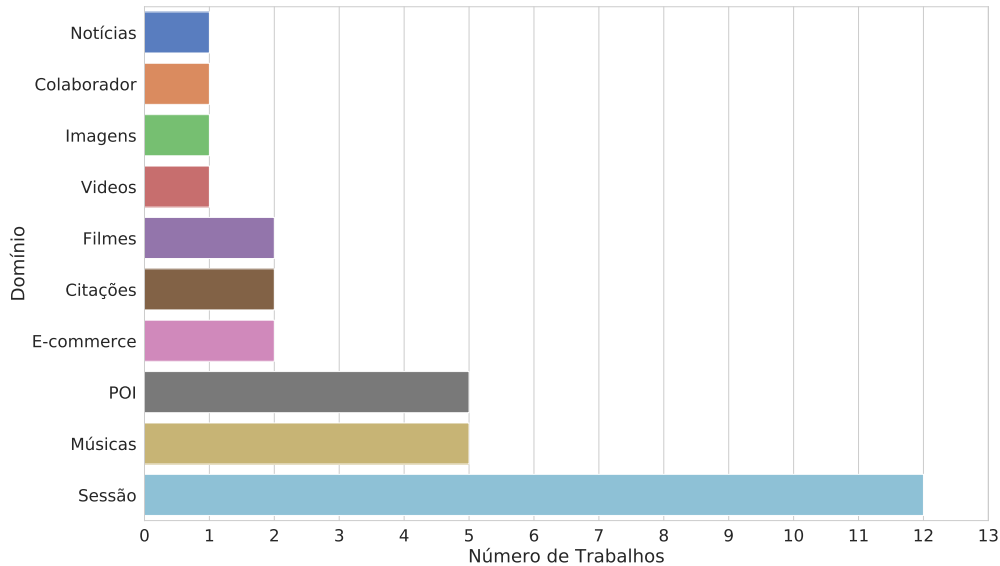


Fonte: Autoria própria.

No total, foram 10 domínios explorados pelos trabalhos, como pode ser visto na Figura 15. O domínio que mais foi explorado foi o domínio de Sessão, no qual 12 trabalhos o exploraram. De maneira geral, ele foi explorado em conjunto com outros domínios. O domínio de Músicas, por exemplo, apareceu em conjunto 4 vezes com o domínio de Sessão. A quantidade de trabalhos que explora o domínio de Sessão em conjunto com outro domínio

denota que o estudo de sessões tem se mostrado predominante em relação aos demais domínios.

Figura 15: Domínios explorados pelos trabalhos.



Fonte: Autoria própria.

Para classificar um trabalho no domínio de Sessão, foi utilizada a definição de sessão proposta por Wang et al. (2019):

Uma sessão é uma coleção de itens que são coletados ou consumidos em um evento ou em determinado período de tempo ou uma coleção de ações ou eventos que aconteceram em um período de tempo.

Trabalhos que utilizavam sessões pressupõem que a sequência na qual o usuário interagiu com os itens pode fornecer bons *embeddings* contextuais para sistemas de recomendação. Alguns domínios, como o domínio da Música, por exemplo, são bons candidatos para sistemas de recomendação baseados em sessão, visto que usuários tendem a ouvir músicas em sequência.

Alguns trabalhos (TAN et al., 2016a; SMIRNOVA; VASILE, 2017) que estavam no domínio de Sessão, no entanto, não tinham o intuito de utilizar sessões em conjunto com algum outro domínio. Seu objetivo era construir uma arquitetura genérica que fosse possível obter *embeddings* para serem usados em sistemas de recomendação sensíveis ao contexto independente do domínio.

Os 20 trabalhos selecionados foram agrupados em três categorias de acordo com o tipo de modelo que foi utilizado para adquirir os *embeddings*:

1. Modelos adaptados da área de Processamento de Linguagem Natural;
2. Modelos baseados em Redes Neurais *feedforward*;
3. Modelos baseados em Redes Neurais Recorrentes.

O único trabalho que não se enquadrava em nenhuma das três categorias foi o de Xie et al. (2016), que usou diversos grafos bipartidos que mapearam Pontos de Interesse (POIs) e informações contextuais. Os *embeddings* foram obtidos pelo relacionamento entre os POIs e as informações contextuais.

Na categoria dos trabalhos que utilizam modelos adaptados da área de processamento de linguagem natural, os primeiros trabalhos (TANG et al., 2014; DENG et al., 2015) que foram publicados adaptaram a técnica de *bag-of-words* para se obter *embeddings*. Apesar do modelo *bag-of-words* ser utilizado para se obter uma representação contextual dos itens, o vetor obtido pelo modelo *bag-of-words* possui diversas limitações, sendo a principal, o tamanho do vetor.

Ambos os trabalhos não usam sessões, que é uma característica dos demais trabalhos dessa categoria. O trabalho de Tang et al. (2014) aplicou uma adaptação do modelo de *bag-of-words* para o domínio de Citações e usou as palavras ao redor da citação como informação contextual, enquanto o trabalho de Deng et al. (2015) explorou o domínio de Músicas, utilizando as emoções como informação contextual. Por trabalhar com um conjunto finito de emoções, a limitação do *bag-of-words* do tamanho do vetor não se aplica ao caso do trabalho de Deng et al. (2015).

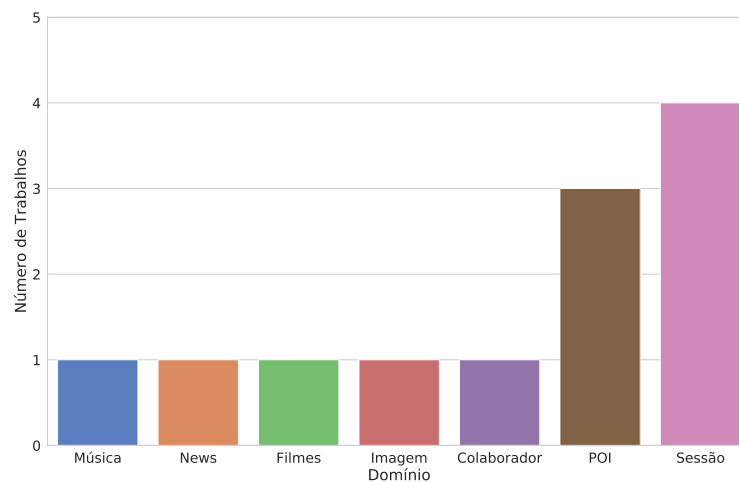
Os demais trabalhos (WANG et al., 2016; ZHAO et al., 2017; LI et al., 2018; WANG et al., 2018) da categoria utilizaram o *word2vec* (apresentado na Seção 2.2), adaptando o modelo *Skip-Gram* para obter informação contextual de sessões para o seu domínio. Os trabalhos de Wang et al. (2016) e Wang et al. (2018) utilizam o modelo de *Skip-Gram* no domínio de Músicas, com o objetivo de se obter *embeddings* considerando as músicas ao redor de uma música alvo. A principal diferença do trabalho de Wang et al. (2016) foi a inclusão de informações adicionais (metadados sobre as músicas) no processo de obtenção de *embeddings*, enquanto o trabalho de Wang et al. (2018) utilizou informação apenas da sequência de músicas ouvidas pelo usuário.

Apesar de estarem em domínios diferentes (POI e E-commerce), os trabalhos de Zhao et al. (2017) e Li et al. (2018) utilizaram o modelo de *Skip-Gram* para a obtenção de *embeddings*

utilizando apenas a informação da sequência de itens como contexto. O modelo de Li et al. (2018), no entanto, usou uma técnica que dava prioridade para os itens que apareciam mais vezes na sequência, o que trouxe resultados superiores para o domínio de E-commerce.

A categoria de modelos baseados em Redes Neurais *feedforward* pode ser vista como a categoria mais heterogênea de todas, não só pela variedade de modelos, mas pela variedade de domínios explorados. No total, são sete domínios explorados de um total de 10 domínios, como pode ser visto na Figura 16.

Figura 16: Domínios explorados na categoria de Redes Neurais *feedforward*.



Fonte: Autoria própria.

Duas arquiteturas de redes neurais foram usadas mais de uma vez dentro dessa categoria. A primeira arquitetura, *AutoEncoder*, é uma arquitetura de rede neural que tem como objetivo copiar os dados de entrada para a saída por meio de um aprendizado não supervisionado, aprendendo uma representação dos dados nas camadas internas da rede (BALDI, 2011). A segunda arquitetura, Redes Neurais Convolucionais (CNNs), proposta por LeCun et al. (1999), tem como foco principal o processamento e reconhecimento de imagens, porém como visto em Goodfellow et al. (2016), pode ser utilizada para processar sequências de dados. Os demais trabalhos que não utilizaram essas arquiteturas propuseram um modelo de rede neural *feedforward* próprio de acordo com seus dados.

Os trabalhos que usaram o modelo *AutoEncoder* (UNGER, 2015; VÖTTER et al., 2019), o utilizaram em domínios diferentes (POI e Músicas com Sessões, respectivamente), e fins diferentes. O trabalho de Unger (2015) utilizou o modelo para obter informação contextual sobre o usuário em forma de vetores a partir de dados obtidos por meio de sensores de

smartphones, que então seriam usados para recomendar localizações em tempo real para o usuário. O trabalho de Vötter et al. (2019) adaptou diversos modelos de *AutoEncoder* para a tarefa de recomendação de próxima música. Em vez de usar o *AutoEncoder* para transformar as músicas em *embeddings*, as *playlists* foram transformadas em *embeddings* e as próximas músicas foram computadas a partir da *playlist*. Por se tratar de recomendação de próxima música e utilizar informações de *playlists*, o trabalho de Vötter et al. (2019) está classificado no domínio de Sessão em conjunto com o domínio de Músicas.

Dois trabalhos (ZHANG et al., 2018; Da Costa; DOLOG, 2019) dentro dessa categoria utilizaram CNNs para obter *embeddings* - apesar do trabalho de Zhang et al. (2018) ter utilizado uma RNN para recomendação, a obtenção de *embeddings* foi feita com uma CNN. Como visto anteriormente, CNNs podem ser utilizadas para processar sequências assim como RNNs, e ambos os trabalhos exploraram o domínio de Sessão em conjunto com os demais domínios. Dentre todos os trabalhos obtidos pela revisão sistemática, o trabalho de Da Costa e Dolog (2019) foi o que explorou o maior número de domínios - Filmes, POI e Imagens; utilizando a sequência desses itens para obter os *embeddings* utilizados.

O trabalho de Zhang et al. (2018) propõe um modelo com uma CNN que é responsável por aprender informações sobre as características das notícias que foram lidas pelo usuário (como a categoria e palavras-chave), mas também sobre a sequência de notícias que o usuário interagiu no *website*. Todas essas informações, após o treinamento, resulta em um vetor de *embeddings* que é utilizado por uma RNN que recomenda notícias para o usuário.

Os demais trabalhos (YANG et al., 2017; LIU et al., 2018) desenvolveram cada um modelo próprio de rede *feedforward* de acordo com as informações contextuais que utilizaram. O trabalho de Yang et al. (2017) utilizou dois tipos de *embeddings* em sua rede *feedforward*, um para o usuário e um para o POI, e os aprendeu utilizando a mesma rede neural. As informações contextuais utilizadas pelo trabalho foram a geolocalização dos POIs e informação sobre grupos sociais dos usuários. Em seu modelo, os *embeddings* do usuário e do POI são combinados para que se possa obter a preferência contextual do usuário com relação ao POI. Já o trabalho de Liu et al. (2018) tenta recomendar pessoas para cooperar em trabalhos acadêmicos, e foi categorizado no domínio Colaborador. Sua arquitetura de rede neural *feedforward* tenta mapear os tópicos nos quais o colaborador estaria disposto a trabalhar e o perfil do colaborador (seus trabalhos passados) em um mesmo vetor de *embeddings*, que é então usado para encontrar quais são os outros colaboradores que possuem o vetor de *embeddings* mais similar.

A última categoria de trabalhos é a dos que utilizaram RNNs para se obter *embeddings* para os itens. Dentre os trabalhos nesta categoria, apenas dois de sete não utilizaram alguma

arquitetura de RNN para explorar o domínio de Sessão. Esses trabalhos (TAN et al., 2016a; XIA et al., 2019) utilizaram diferentes arquiteturas de RNNs em diferentes domínios, mas ambos utilizaram as arquiteturas para obter *embeddings* de palavras. As palavras, no entanto, não eram o item que seria recomendado, mas sim uma das informações utilizadas para construir o *embedding* do item.

O único trabalho (BEUTEL et al., 2018) que utilizou o modelo mais simplificado de RNN (descrita na Sessão 2.3), explorou o domínio de Vídeos em conjunto com o de Sessão. Beutel et al. (2018) desenvolveu uma técnica chamada *Latent Cross*, que consiste em incorporar informação contextual em uma RNN por meio de uma operação de produto vetorial entre o vetor que contém as características da informação contextual com os estados ocultos da RNN.

Uma arquitetura de RNN que foi explorada nessa categoria pelos trabalhos foi a arquitetura LSTM (descrita na Subseção 2.3.1), e foi utilizada nos trabalhos de Tan et al. (2016a) e Zhao et al. (2019). O trabalho de Tan et al. (2016a), como dito anteriormente, não explorou o domínio de Sessão apesar de utilizar uma arquitetura de LSTM. Seu trabalho explorou o domínio de Citações e utilizou a média dos *embeddings* das palavras em uma citação para representar uma citação. Com isso, foi possível recomendar citações de acordo com o contexto (as palavras) que ficariam ao redor dela em um texto. O trabalho de Zhao et al. (2019) explorou o domínio de Filmes em conjunto com o domínio de Sessão, e utilizou uma arquitetura de LSTM para obter *embeddings* por meio da sequência dos filmes que um usuário assistiu.

Os demais trabalhos utilizaram uma arquitetura proposta por Cho et al. (2015), chamada *Gated Recurrent Unit* (GRU), similar à arquitetura LSTM, porém sem o portão de saída e que demanda menos tempo para treinar. O trabalho de Xia et al. (2019), como mencionado anteriormente, não faz parte do domínio de Sessão, apenas do domínio de E-commerce. O trabalho teve como objetivo aprender o perfil do usuário e dos itens por meio das avaliações, e a representação vetorial das avaliações foi aprendida por meio de uma média dos *embeddings* das palavras contidas na avaliação deixada pelo usuário. Para aprender os *embeddings* das palavras, foi utilizada uma GRU bi-direcional, que consegue aprender utilizando informações tanto do passado quanto do futuro.

Assim como o trabalho de Tan et al. (2016a), o trabalho de Smirnova e Vasile (2017) não explorou nenhum outro domínio que não fosse o domínio de Sessão. Ambos tentaram otimizar uma arquitetura GRU por meio de uma incorporação da informação contextual junto com os dados da GRU. A diferença do trabalho de Smirnova e Vasile (2017), no entanto, é que em vez de incluir essas informações junto com os dados ocultos, como foi feito em Tan et al. (2016a), essas informações contextuais foram incluídas tanto nas camadas de entrada quanto

nas camadas de saída.

O trabalho de Mayerl et al. (2019) explorou o domínio de Músicas em conjunto com o domínio de Sessão, e utilizou um modelo simples de GRU para obter os *embeddings* das músicas. Similar ao trabalho de Wang et al. (2018), foi utilizado o modelo proposto para obter os *embeddings* a partir da sequência de músicas que o usuário ouviu na aplicação. No entanto, Mayerl et al. (2019) utilizou a GRU tanto para obter os *embeddings* quanto para posteriormente recomendar as músicas, para a tarefa de recomendação de próxima música.

O Quadro 1 sintetiza os trabalhos que foram obtidos pela revisão sistemática. A categoria NLP representa os trabalhos que utilizaram modelos adaptados da área de Processamento de Linguagem Natural, a categoria FFNN representa os trabalhos que utilizaram redes *feedforward* e a categoria RNN representa os trabalhos que utilizaram Redes Neurais Recorrentes. A categoria GRAFOS simboliza os trabalhos que usaram grafos para obter os *embeddings*.

Quadro 1: Sumário dos trabalhos obtidos pela revisão sistemática.

Trabalho	Ano de Publicação	Domínio(s)	Categoria	Modelo Utilizado
Unger (2015)	2015	POI	FFNN	AutoEncoder
Yang et al. (2017)	2017	POI	FFNN	Rede feedforward
Zhang et al. (2018)	2018	Notícias; Sessão	FFNN	CNN
Liu et al. (2018)	2018	Colaborador	FFNN	Rede feedforward
Da Costa e Dolog (2019)	2019	Filmes; POI; Imagem; Sessão	FFNN	CNN
Vötter et al. (2019)	2019	Música; Sessão	FFNN	AutoEncoder
Xie et al. (2016)	2016	POI	GRAFOS	Grafos Bipartidos
Tang et al. (2014)	2014	Citação	NLP	Bag-of-words
Deng et al. (2015)	2015	Música	NLP	Bag-of-words
Wang et al. (2016)	2016	Música; Sessão	NLP	Word2vec
Zhao et al. (2017)	2017	POI; Sessão	NLP	Word2vec
Li et al. (2018)	2018	E-commerce; Sessão	NLP	Word2vec
Wang et al. (2018)	2018	Música; Sessão	NLP	Word2vec
Tan et al. (2016a)	2016	Citação	RNN	RNN - LSTM
Tan et al. (2016b)	2016	Sessão	RNN	RNN - GRU
Smirnova e Vasile (2017)	2017	Sessão	RNN	RNN - GRU
Beutel et al. (2018)	2018	Vídeos; Sessão	RNN	RNN
Xia et al. (2019)	2019	E-commerce	RNN	RNN - GRU
Zhao et al. (2019)	2019	Filmes; Sessão	RNN	RNN - LSTM
Mayerl et al. (2019)	2019	Música; Sessão	RNN	RNN - GRU
Trabalho proposto	2020	Música; Sessão	RNN	RNN - LSTM

Fonte: Autoria própria.

A próxima seção apresenta tanto o modelo proposto neste trabalho para obtenção de *embeddings*, quanto o modelo de Wang et al. (2018), que foi utilizado como base de comparação para validar o modelo proposto. Além disso, descreve também a base de dados *Music4All* criada durante a realização deste trabalho.

4 OBTENÇÃO DE EMBEDDINGS COMO INFORMAÇÃO CONTEXTUAL

Nesta seção é apresentado o modelo proposto por Wang et al. (2018), que utiliza redes neurais tradicionais para a obtenção de *embeddings*, isto é, informação contextual que pode ser utilizada por sistemas de recomendação sensíveis ao contexto. Esse modelo é utilizado como *baseline* de comparação para a proposta apresentada nesta dissertação de mestrado. Além disso, são apresentados os sistemas de recomendação sensíveis ao contexto que foram utilizados para avaliar os *embeddings*, isto é, a informação contextual obtida pelo modelo.

4.1 OBTENÇÃO DE EMBEDDINGS

Como visto na Seção 3, diversos trabalhos utilizam métodos de obtenção de *embeddings* para representar itens em sistemas de recomendação. Com relação as recomendações sensíveis ao contexto de músicas, pode-se obter além de uma representação de tais itens, uma representação que seja capaz de incluir informações contextuais. Pode-se obter essas informações contextuais em formato de *embeddings* levando em consideração diferentes tipos de informação, como: informações sobre o gênero da música, *tags* que usuários atribuíram a música, entre outras.

Neste trabalho, as informações contextuais foram obtidas por meio de *embeddings*, que são resultantes de uma análise da sequência de músicas que um usuário ouviu. Como modelo de comparação para o trabalho proposto, foi escolhido o modelo descrito em (WANG et al., 2018), que é apresentado a seguir. Este trabalho foi escolhido pois utiliza o modelo de obtenção de *embeddings* apresentado na Subseção 2.2, um modelo que é utilizado nos mais diversos domínios de aplicação para se obter *embeddings* de uma determinada sequência.

4.2 MODELO DO BASELINE

De maneira formal, como definido por Wang et al. (2018), seja $U = \{u_1, u_2, \dots, u_{|U|}\}$ o conjunto de usuários e $M = \{m_1, m_2, \dots, m_{|M|}\}$ o conjunto de músicas, nos quais $|U|$ e $|M|$

remetem ao número total de usuários e músicas únicos na base de dados, respectivamente. Para cada usuário u , seu histórico de execução de músicas corresponde às músicas que foram ouvidas pelo usuário com a data e a hora nos quais elas foram ouvidas, definido como $H^u = \{m_1^u, m_2^u, \dots, m_{|H^u|}^u\}$, no qual $m_i^u \in M$. O histórico de execução do usuário pode ser dividido em sessões $S^u = \{S_1^u, S_2^u, \dots, S_{|S^u|}^u\}$ de acordo com o intervalo de tempo entre uma música e outra. Assim, cada sessão n do usuário u pode ser definida como $S_n^u = \{m_{n,1}^u, m_{n,2}^u, \dots, m_{n,|s_n^u|}^u\}$, no qual $m_{n,j}^u \in M$.

Como um exemplo para a separação de sessões para um subconjunto da base de dados apresentada no Quadro 2, utilizaremos um intervalo entre músicas de 30 minutos. Neste exemplo, para o usuário $u = \text{user_i2GzdCDG}$, ele possuiria as seguintes sessões $S^u = \{S_1^u, S_2^u, S_3^u\}$, sendo $S_1^u = \{6163, 6164\}$, $S_2^u = \{6165, 6166, 6167, 6168\}$ e $S_3^u = \{6169, 6170, 6171, 6172\}$.

Quadro 2: Subconjunto da base de dados *Music4All*.

Usuário	ID da Música	Música	Timestamp
user_i2GzdCDG	6163	F.U.U. - Dream Wife	2019-02-07 11:32
user_i2GzdCDG	6164	Hey Cool Kid - Cloud Nothings	2019-02-07 11:40
user_i2GzdCDG	6165	Someone Great - LCD Soundsystem	2019-02-07 12:13
user_i2GzdCDG	6166	Orfeo et Eurydice: Mélodie for Piano Solo - Ch...	2019-02-07 12:37
user_i2GzdCDG	6167	I Know It's Over - 2011 Remastered Version - T...	2019-02-07 13:00
user_i2GzdCDG	6168	I'm Waiting Here - Bonus Track - David Lynch	2019-02-07 13:09
user_i2GzdCDG	6169	We Exist - Arcade Fire	2019-02-07 14:40
user_i2GzdCDG	6170	H In New England - Max Richter	2019-02-07 14:52
user_i2GzdCDG	6171	From Dog to God - Prayers	2019-02-07 15:11
user_i2GzdCDG	6172	1990 - Soviet Soviet	2019-02-07 15:18

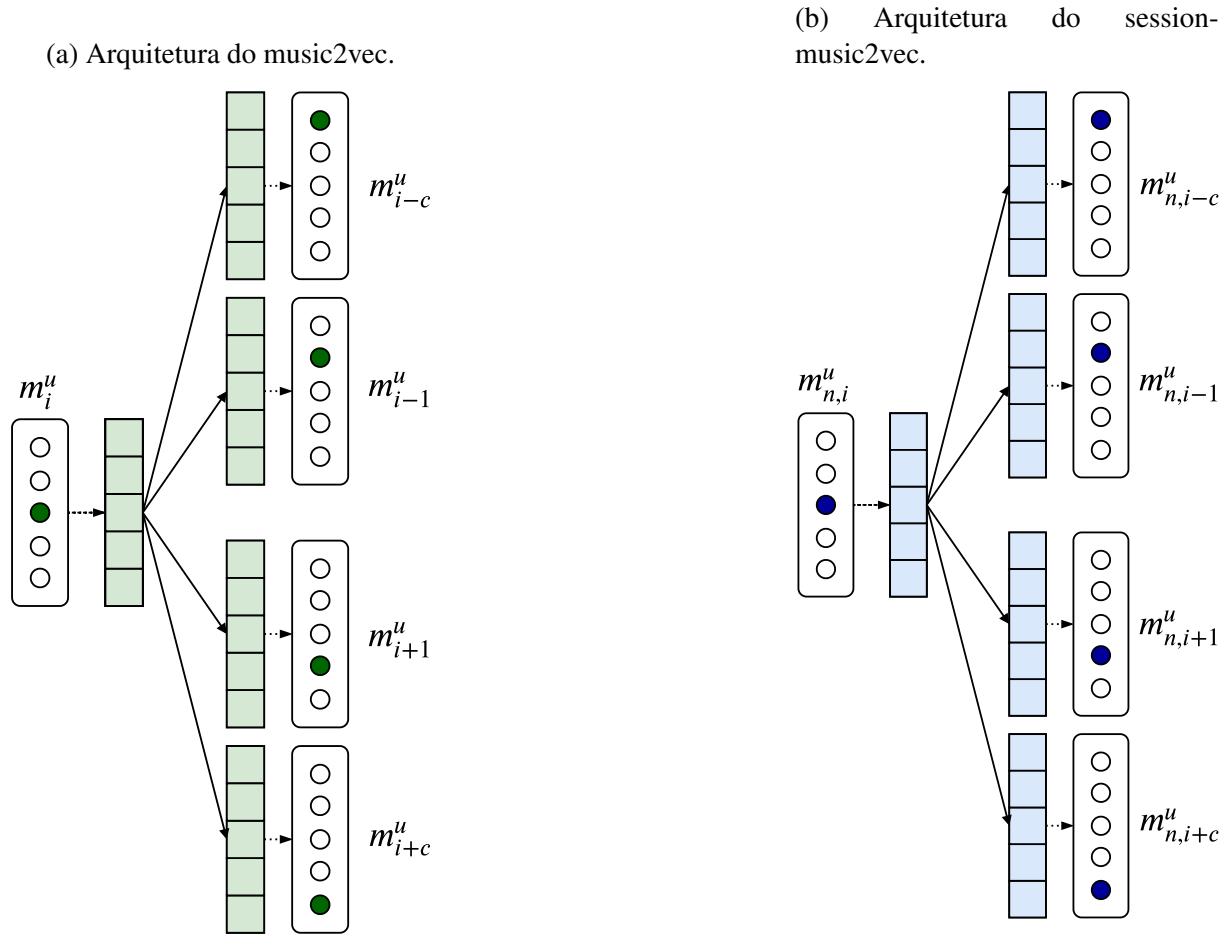
Fonte: Autoria própria.

4.2.1 MUSIC2VEC E SESSION-MUSIC2VEC

Os métodos para obtenção de *embeddings* *music2vec* e *session-music2vec* compõem o modelo proposto por Wang et al. (2018), que é utilizado como *baseline* nesta dissertação. A Figura 17 ilustra os dois métodos. É possível ver que ambos os métodos, tanto o *music2vec* (Figura 17a) quanto o *session-music2vec* (Figura 17b), possuem a mesma arquitetura do modelo *Skip-Gram* apresentado na Seção 2.2. A principal diferença para o modelo *Skip-Gram* se dá pelo fato de que em vez de utilizar palavras para se obter os *embeddings*, tanto o *music2vec* quanto o *session-music2vec* utilizam músicas.

O método *music2vec* tem como objetivo aprender os *embeddings* das músicas dado o histórico de execução completo do usuário. A ideia principal por trás do modelo *music2vec* é que a sequência de músicas ouvidas pelos usuários refletem suas preferências para música durante aquele período de tempo, e que a coocorrência dessas músicas em uma sequência indica

Figura 17: Métodos propostos por Wang et al. (2018).



Fonte: Adaptado de Wang et al. (2018).

a similaridade entre essas músicas (WANG et al., 2018). Com isso, os *embeddings* de músicas que aparecem perto umas das outras devem aparecer perto uns dos outros em um espaço de baixa dimensão.

De maneira geral, o music2vec aprende o *embedding* de uma música m_i^u das músicas vizinhas $\{m_{i-c}^u : m_{i+c}^u\} \setminus m_i^u$ no histórico de execução completo H^u . De maneira formal, como descrito por Wang et al. (2018), a função objetivo do modelo music2vec é definida como:

$$L = \sum_{u \in U} \sum_{m_i^u \in H^u} \sum_{-c \leq j \leq c} \log p(m_{i+j}^u | m_i^u), \quad (12)$$

no qual c corresponde ao tamanho da janela contextual. Assim como no word2vec, a janela contextual desliza por todo o histórico de execução de músicas do usuário. Na Equação 12, $p(m_{i+j}^u | m_i^u)$ representa a probabilidade condicional de uma música m_{i+j}^u estar na janela contextual da música m_i^u em H^u , que formalmente é definido usando a função *softmax* definida

como:

$$p(m_{i+j}^u | m_i^u) = \exp(\mathbf{v}_{m_i^*}^T \cdot \mathbf{v}_{m_{i+j}^*}) / \sum_{m \in M} \exp(\mathbf{v}_{m_i^*}^T \cdot \mathbf{v}_m'), \quad (13)$$

no qual \mathbf{v}_m e \mathbf{v}_m' são os vetores de *embeddings* de entrada e saída da música m , respectivamente. Como visto nas Equações 12 e 13, a janela contextual do modelo *music2vec* percorre todo o histórico de execução do usuário. No entanto, os gostos musicais do usuário podem variar muito durante todo o histórico de execução e, por isso, a abordagem baseada em sessões é mais viável para capturar essas variações, ou seja, a abordagem baseada em sessões pode capturar o contexto no qual o usuário está ouvindo a música naquele momento.

O método *session-music2vec* tenta capturar a informação contextual no histórico de execução do usuário. em vez de aprender os *embeddings* com o histórico inteiro do usuário, o histórico é dividido em sessões menores que são utilizadas para aprender os *embeddings*.

Assim como o *music2vec*, o *session-music2vec* se baseia também no modelo de *Skip-Gram*, porém em vez da janela contextual deslizar por todo o histórico de execução do usuário para obter os *embeddings*, ela desliza por cada uma das sessões do usuário. O modelo, em síntese, procura aprender o vetor de *embeddings* de uma música $m_{n,i}^u$, de suas músicas vizinhas $\{m_{n,i-c}^u : m_{n,i+c}^u\} \setminus m_{n,i}^u$, na sessão n do usuário u . Como descrito por Wang et al. (2018), a função objetivo do *session-music2vec* pode ser definida como

$$L = \sum_{u \in U} \sum_{S_n^* \in S^*} \sum_{m_{n,i}^* \in S_n^*} \sum_{-c \leq j \leq c} \log p(m_{n,i+j}^u | m_{n,i}^u), \quad (14)$$

que é similar a função objetivo definida na Equação 12, porém itera também sobre todas as sessões do usuário. De maneira similar, $p(m_{n,i+j}^u | m_{n,i}^u)$ representa a probabilidade condicional de uma música vizinha $m_{n,i+j}^u$ dado a música $m_{n,i}^u$ em uma determinada sessão S_n^u , que é definida usando a função *softmax*:

$$p(m_{n,i+j}^u | m_{n,i}^u) = \exp(\mathbf{v}_{m_{n,i}^u}^T \cdot \mathbf{v}_{m_{n,i+j}^u}') / \sum_{m \in M} \exp(\mathbf{v}_{m_{n,i}^u}^T \cdot \mathbf{v}_m'). \quad (15)$$

Assim, os métodos que compõem o modelo proposto por Wang et al. (2018) são capazes de capturar *embeddings* que refletem dois tipos de informação do usuário para as músicas:

1. Preferência geral, que diz respeito ao gosto mais amplo do usuário com relação à música,

que não muda com o tempo;

2. Preferência contextual, que pode ser vista como uma preferência local, e que diz respeito à preferência do usuário na sessão atual.

4.3 SISTEMAS DE RECOMENDAÇÃO CONTEXTUAIS

Com o objetivo de avaliar os *embeddings* extraídos pelo modelo apresentado na Seção 4.2, Wang et al. (2018) propôs 4 sistemas de recomendação sensíveis ao contexto que utilizam esses *embeddings* para calcular as preferências do usuário e recomendar músicas com base em tais preferências.

Tais sistemas de recomendação fazem uso de duas preferências, que são construídas a partir dos *embeddings* obtidos com o modelo. A **preferência geral** de um usuário u para músicas, aprendida por meio de seu histórico de execução completo $H^u = \{m_1^u, m_2^u, \dots, m_{|H^u|}^u\}$, pode ser definida como:

$$\mathbf{p}_g^u = \frac{1}{|H^u|} \sum_{m_i^u \in H^u} \mathbf{v}_{m_i^u}^{m2v}, \quad (16)$$

no qual $\mathbf{v}_{m_i^u}^{m2v}$ corresponde ao vetor de *embeddings* – que pode ser obtido pelo *baseline* (por meio do método *music2vec* do modelo proposto por Wang et al. (2018)) ou pela camada de *Embeddings Gerais* do modelo proposto neste trabalho – da música m . A Equação 16 realiza uma operação de média entre os vetores de *embeddings* das músicas do histórico de execução completo para construir a preferência geral.

Para a **preferência contextual** de um determinado usuário u , dada sua atual sessão $S_n^u = \{m_{n,1}^u, m_{n,2}^u, \dots, m_{n,|S_n^u|}^u\}$, sua preferência pode ser definida como:

$$\mathbf{p}_c^u = \frac{1}{|S_n^u|} \sum_{m_{n,i}^u \in S_n^u} \mathbf{v}_{m_{n,i}^u}^{s2v}, \quad (17)$$

no qual $\mathbf{v}_{m_{n,i}^u}^{s2v}$ corresponde ao vetor de *embeddings* – que pode ser obtido pelo *baseline* (por meio do método *session-music2vec* do modelo proposto por Wang et al. (2018)) ou pela camada de *Embeddings Contextuais* do modelo proposto neste trabalho – da música m . A Equação 17 realiza uma operação de média entre os vetores de *embeddings* das músicas da sessão atual para se obter a preferência contextual.

Dados os *embeddings* gerais e contextuais das músicas, junto com as preferências

gerais e contextuais dos usuários, Wang et al. (2018) definiu quatro sistemas de recomendações sensíveis ao contexto de músicas: Music2vec-TopN (M-TN), Session-Music2vec-TopN (SM-TN), Context-Session-Music2vec-TopN (CSM-TN) e Context-Session-Music2vec-UserKNN (CSM-UK).

O sistema de recomendação M-TN é o único dos quatro sistemas que utiliza apenas as informações gerais das músicas e dos usuários para predizer a preferência do usuário, isto é, não usa informações contextuais. Assim, como definido por Wang et al. (2018), dado um usuário u e sua preferência geral \mathbf{p}_g^u para as músicas, o sistema calcula a similaridade cosseno entre \mathbf{p}_g^u e o vetor de *embeddings* gerais $\mathbf{v}_{m_i}^{m2v}$ para cada música no conjunto de músicas M , recomendando as top- N músicas mais similares ao usuário. Formalmente, a preferência prevista (pp) do usuário u para a música m pode ser definida como:

$$pp_{M-TN}(u, m) = \cos(\mathbf{p}_g^u, \mathbf{v}_m^{m2v}). \quad (18)$$

Similar ao sistema de recomendação M-TN, o SM-TN substitui informações gerais pelas informações contextuais com o objetivo de recomendar músicas com base nas preferências contextuais do usuário. Dado o usuário u e sua preferência contextual \mathbf{p}_c^u , o SM-TN então calcula a similaridade cosseno entre o vetor de *embeddings* contextuais $\mathbf{v}_{m_{n,i}}^{s2v}$ das músicas e a preferência contextual do usuário. As top- N músicas com o maior valor de similaridade são recomendadas ao usuário. A preferência do usuário u para a música m pode ser definida como:

$$pp_{SM-TN}(u, m) = \cos(\mathbf{p}_c^u, \mathbf{v}_m^{s2v}). \quad (19)$$

O sistema de recomendação CSM-TN tem como objetivo utilizar ambas as preferências do usuário u na hora de fazer as recomendações para o usuário. Dadas as preferências geral \mathbf{p}_g^u e contextual \mathbf{p}_c^u do usuário u para o conjunto de músicas M , o CSM-TN calcula a similaridade cosseno entre a preferência \mathbf{p}_g^u e os *embeddings* gerais $\mathbf{v}_{m_i}^{m2v}$ das músicas, e a similaridade cosseno entre a preferência \mathbf{p}_c^u e os *embeddings* contextuais $\mathbf{v}_{m_{n,i}}^{s2v}$ também das músicas do conjunto de músicas M . Após os cálculos, essas similaridades serão somadas para que sejam obtidas as músicas mais similares de acordo com as preferências gerais e contextuais. Formalmente, a preferência do usuário é definida como:

$$PP_{CSM-TN}(u, m) = \cos(\mathbf{p}_g^u, \mathbf{v}_m^{m2v}) + \cos(\mathbf{p}_c^u, \mathbf{v}_m^{s2v}). \quad (20)$$

O quarto sistema de recomendação, CSM-UK, combina um sistema de recomendação

tradicional, chamado de UserKNN (RESNICK et al., 1994), com os vetores de *embeddings* aprendidos.

Como apresentado na Seção 2.1, um sistema de recomendação UserKNN precisa de uma função de similaridade entre dois usuários. O sistema CSM-UK utiliza a seguinte função de similaridade entre dois usuários u e v , proposta por Wang et al. (2018):

$$\text{sim}(u, v) = \sum_{m \in M_u \cap M_v} \frac{1}{\sqrt{|M_u| \times |M_v|} + \cos(\mathbf{p}_g^u, \mathbf{p}_g^v)}, \quad (21)$$

no qual u e v são dois usuários, e M_u e M_v correspondem ao conjunto de músicas ouvidas pelos usuários u e v . Com a definição da similaridade entre os usuários, a tarefa do sistema de recomendação CSM-UK consiste em recomendar as top- N músicas mais similares, dado a preferência calculada por meio dos vetores de *embeddings* e dos usuários similares ao usuário alvo. Com isso, a preferência do usuário alvo u para uma música m pode ser definida como:

$$PP_{CSM-UK}(u, m) = \left(\sum_{v \in U_{u,K} \cap U_m} \frac{\text{sim}(u, v)}{|U_{u,k} \cap U_m|} \right) + \cos(\mathbf{p}_c^u, \mathbf{v}_m^{s2v}), \quad (22)$$

onde $U_{u,K}$ é o conjunto com os K usuários mais similares a u , e U_m é o conjunto de usuários que ouviram a música m .

Os sistemas de recomendação apresentados nesta subseção serão utilizados para avaliar os *embeddings* obtidos não só pelo modelo apresentado na Subseção 4.2.1, mas também pelo modelo de Rede Neural Recorrente proposto na Seção 5.

5 DESENVOLVIMENTOS

Nesta seção é apresentado o modelo de obtenção de *embeddings* proposto neste trabalho, que busca extrair os *embeddings* por meio de redes neurais recorrentes. Também é apresentada a base de dados *Music4All*, construída com o objetivo de fornecer mais uma base de dados para avaliar o modelo proposto.

5.1 MODELO CONJUNTO DE APRENDIZADO DE EMBEDDINGS BASEADO EM REDES NEURAS RECORRENTE

No modelo de Wang et al. (2018) foram propostos dois métodos para a obtenção de *embeddings* que podem ser utilizados em duas situações diferentes: um quando se pretende obter a preferência geral do usuário, e outro quando se pretende obter a preferência contextual do usuário. Com o objetivo de explorar a versatilidade das redes neurais artificiais, nesta dissertação de mestrado foi proposto um modelo¹, baseado em Redes Neurais Recorrentes, capaz de aprender ao mesmo tempo ambas as preferências dos usuários para as músicas, isto é, produzir *embeddings* que podem ser utilizados em sistemas de recomendações sensíveis ao contexto.

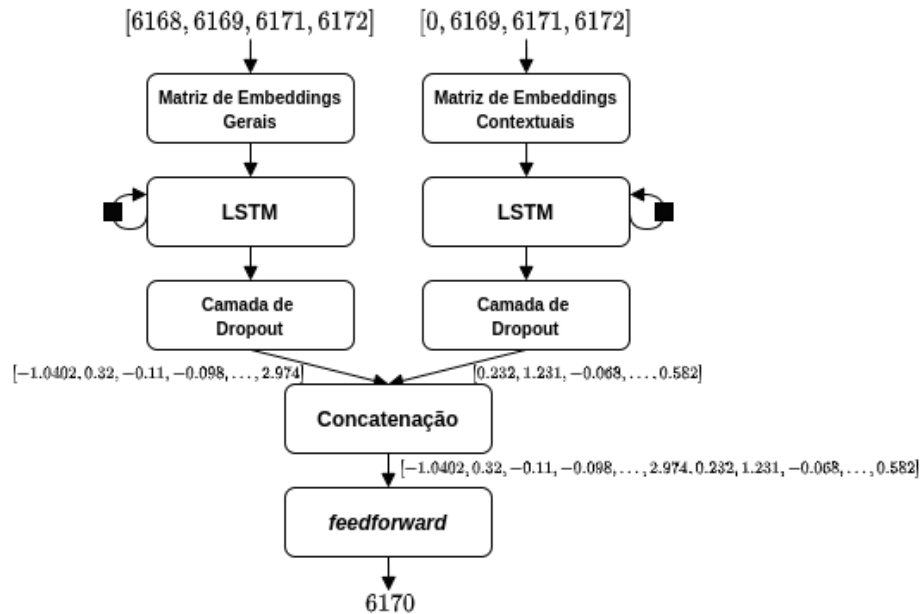
O objetivo principal do modelo proposto nesta dissertação é, assim como o modelo CBOW proposto por Mikolov et al. (2013a), prever a música central de uma janela contextual dadas suas músicas vizinhas. No entanto, ao utilizar uma arquitetura de Redes Neurais Recorrentes, em específico, redes LSTM, apresentadas na Subseção 2.3.1, a rede se torna capaz de levar em consideração as músicas vizinhas na ordem em que aparecem para classificar a música central, ao contrário do modelo CBOW, no qual a ordem das músicas vizinhas não são relevantes. Assim, é possível levar em consideração a sequência em que as músicas vizinhas estão para criar os *embeddings* das músicas.

A Figura 18 apresenta uma representação da rede neural proposta e de suas camadas. A entrada de dados do lado esquerdo da figura corresponde a janela contextual de uma música m_i^u ,

¹<https://github.com/igorsantana/rnn-embeddings>

levando em consideração o histórico completo de execução do usuário H^u , enquanto a entrada de dados do lado direito da figura corresponde a janela contextual da mesma música $m_{n,i}^u$, porém levando em consideração a sessão na qual a música está inserida.

Figura 18: Modelo conjunto de aprendizado de *embeddings* de músicas.



Fonte: Autoria própria.

Para exemplificar as entradas da rede proposta, utilizaremos o exemplo do Quadro 2 com o intervalo de sessão de 30 minutos, atualizado no Quadro 3. Utilizando a música com o ID 6170 como a música candidata para a predição e uma janela contextual $c = 2$, a janela contextual da preferência geral tomaria a seguinte forma $\{m_{i-c}^u : m_{i+c}^u\} = \{6168, 6169, 6171, 6172\}$, enquanto a janela contextual da preferência contextual tomaria a seguinte forma $\{m_{n,i-c}^u : m_{n,i+c}^u\} = \{0, 6169, 6171, 6172\}$. O 0 é utilizado como um algoritmo especial que indica a falta de informação para a rede, pois como a sessão começa na música de ID 6168, a janela contextual não consegue obter uma música anterior.

Como pode ser visto na Figura 18, as janelas contextuais da música 6170 serão utilizadas como entrada do modelo, tanto para definição das preferências gerais quanto contextuais. Esse processo ocorrerá para todo o histórico de execução do usuário, passando por todas as músicas e suas janelas contextuais.

Na camada de *embeddings*, existe uma matriz para cada uma das preferências, onde cada linha representará uma música da base de dados e o número de colunas representa o tamanho de vetor de *embeddings*, que é escolhido durante a execução. Essa matriz é inicializada

Quadro 3: Subconjunto da base de dados *Music4All* com sessões.

Usuário	ID da Música	Música	Timestamp	Sessão
user_i2GzdCDG	6163	F.U.U. - Dream Wife	2019-02-07 11:32	S_1^u
user_i2GzdCDG	6164	Hey Cool Kid - Cloud Nothings	2019-02-07 11:40	S_1^u
user_i2GzdCDG	6165	Someone Great - LCD Soundsystem	2019-02-07 12:13	S_2^u
user_i2GzdCDG	6166	Orfeo et Eurydice: Mélodie for Piano Solo - Ch...	2019-02-07 12:37	S_2^u
user_i2GzdCDG	6167	I Know It's Over - 2011 Remastered Version - T...	2019-02-07 13:00	S_2^u
user_i2GzdCDG	6168	I'm Waiting Here - Bonus Track - David Lynch	2019-02-07 13:09	S_2^u
user_i2GzdCDG	6169	We Exist - Arcade Fire	2019-02-07 14:40	S_3^u
user_i2GzdCDG	6170	H In New England - Max Richter	2019-02-07 14:52	S_3^u
user_i2GzdCDG	6171	From Dog to God - Prayers	2019-02-07 15:11	S_3^u
user_i2GzdCDG	6172	1990 - Soviet Soviet	2019-02-07 15:18	S_3^u

Fonte: Autoria própria.

de maneira aleatória, onde os elementos dos vetores de *embeddings* das músicas receberão um valor entre $-0,05$ e $0,05$. A medida que o modelo passa pelo processo de aprendizado de acordo com as fases *forward* e com o BPTT, apresentados na Subseção 2.3, esses valores serão atualizados de acordo com o erro resultante da saída esperada e da saída obtida da rede.

Então, cada uma dessas músicas da janela contextual será utilizada como entrada em uma etapa de tempo nas redes LSTMs, que são treinadas com o objetivo de aprender o relacionamento intrínsecos entre tais músicas. Cada etapa de tempo consiste de um bloco de memórias LSTM, que processará a informação daquela etapa de tempo e resultará em um vetor de tamanho correspondente ao número de memórias LSTM dentro do bloco.

Após as últimas etapas de tempo das redes LSTM, os vetores resultantes de tal etapa são concatenados em um único vetor. Esse vetor será utilizado como entrada para uma rede densa *feedforward*, similar a rede MLP apresentada na Subseção 2.3, com um número de neurônios igual ao tamanho do vetor concatenado resultante. O objetivo dessa rede *feedforward* é utilizar esse vetor para tentar prever qual a música central da janela contextual, já com as informações sequenciais processadas pela rede LSTM.

Nos neurônios da camada *feedforward* do modelo, foi utilizada a função de ativação *softmax*, que normaliza o vetor resultante em uma distribuição de probabilidade para a predição de cada música da base de dados. Os elementos do vetor resultante terão um valor entre 0 e 1, e a soma de todos os elementos do vetor resultará em 1. O elemento que possuir o maior valor entre os outros representa a música predita pela rede.

O processo de treino e teste foi feito de maneira que 80% das músicas fossem separadas para o conjunto de treino, e 20% para teste. Após separar a base de dados, foram organizados conjuntos de 64 sequências a serem treinadas pelo modelo, para que não houvesse problema de falta de memória. Para treinar a rede, foi utilizado o otimizador *Adam*, uma versão adaptada do processo de treinamento apresentado na Subseção 2.3.

Para prevenir que o modelo proposto sofresse de um problema comum em redes neurais chamado de *overfitting*, ou sobreajuste, foi utilizada a técnica de Dropout, que desativa alguns neurônios aleatoriamente durante o processo de treinamento, fazendo com que a rede seja mais efetiva para fazer previsões. O *overfitting*, ou sobreajuste, ocorre quando o modelo se ajusta demais aos dados de treinamento, porém não consegue generalizar o aprendizado para novos itens. Neste modelo, em cada camada LSTM, foi utilizado um valor de Dropout de 20%.

Após a rede estar treinada, os vetores de *embeddings* podem ser extraídos da camada de *embeddings*, por meio de uma consulta simples à matriz de *embeddings*. Esses *embeddings* serão utilizados em duas finalidades:

- Calcular a preferência geral e contextual do usuário;
- Representar uma música de acordo com a janela contextual para serem recomendadas por sistemas de recomendação sensíveis ao contexto.

5.2 MUSIC4ALL

Neste trabalho, além do modelo de obtenção de *embeddings* proposto, também foi criada uma base de dados de recomendação de músicas com o histórico de execução de cada usuário (SANTANA et al., 2020). A base, chamada de *Music4All*², foi criada com o intuito de ser uma segunda base de dados para avaliar o modelo proposto, além da base de dados proposta por Wang et al. (2018).

Apesar do objetivo inicial ter sido a criação de uma base de dados para a recomendação de músicas, a base de dados *Music4All* foi estendida com diferentes tipos de informações para que ela pudesse ser utilizada em diferentes áreas de pesquisa. Para poder obter esses diferentes tipos de informações, a construção da base foi dividida em duas fases, como pode ser visto na Figura 19: 1) fase dos usuários e 2) fase das músicas. A fase dos usuários foi responsável por obter o histórico de execução dos usuários, enquanto a fase das músicas foi responsável por obter dados sobre as músicas, como o áudio e as letras.

A fase dos usuários teve seu início por meio da obtenção de um usuário aleatório da aplicação last.fm³ como um ponto de partida e, a partir deste usuário, os seus usuários amigos foram obtidos por meio da API do last.fm⁴, de maneira recursiva, até que uma grande

²<https://sites.google.com/view/contact4music4all>

³<https://www.last.fm>

⁴<https://www.last.fm/api>

Figura 19: Processo de criação da base de dados *Music4All*.



Fonte: Autoria própria.

quantidade de usuários tivesse sido obtida. Com a conclusão dessa etapa recursiva, foram obtidos 40,940 usuários, excluindo os usuários duplicados.

Após a obtenção dos usuários, o histórico de execução de cada usuário foi obtido por meio da API do last.fm. O histórico de execução dos usuários compreende músicas que foram ouvidas entre os dias 1 de Janeiro de 2019 até o dia 20 de Março de 2019. Com o histórico de execução de cada usuário obtido, foram removidas todas as informações pessoais e metadados dos usuários por meio de um processo de anonimização, incluindo também o relacionamento entre os usuários. Além disso, foi criado um identificador aleatório para cada usuário.

Para finalizar a fase dos usuários, foram removidos usuários que não possuíam muitas músicas com o objetivo de produzir uma base de dados mais densa. Os usuários que não possuíam pelo menos 1,000 execuções de músicas no seu histórico de execução foram descartados, resultando em um total de 15,602 usuários e 320,073 músicas no final da fase dos usuários.

A fase das músicas consistiu em coletar informações adicionais das músicas, incluindo *tags*, letras, gêneros, áudio e demais metadados. Essas informações foram obtidas de diversas fontes de dados e combinadas posteriormente. Para evitar que músicas ficassem com informações faltando, músicas que não possuíam todas as informações foram descartadas e removidas do histórico de execução dos usuários.

As *tags* de cada música foram obtidas por meio de um *crawler* que, a partir da página de cada música na aplicação last.fm, obteve as *tags* mais populares fornecidas pelos usuários para as músicas. De um total de 320,073 músicas, 306,010 músicas possuíam uma ou mais *tags* associadas.

Para obter os gêneros associados a cada música, foram utilizados os gêneros que estão disponíveis na aplicação *Every Noise At Once*⁵, uma aplicação que tem como objetivo classificar gêneros musicais de acordo com as distinções de gênero feitas pelo Spotify⁶. De 306,010 músicas que possuíam *tags*, foram filtradas músicas que não possuíam nenhum gênero associado, resultando em 266,837 músicas.

Após a filtragem das *tags* e dos gêneros, foram baixados os áudios de cada música em formato mp3 por meio do YouTube⁷. Para obter o *link* do YouTube de cada música, o mesmo *crawler* que foi utilizado para obter *tags* foi utilizado para obter o *link*, visto que grande parte das músicas do last.fm possuem um *link* do YouTube associado. Com o *crawler*, foi possível obter o áudio de 218,429 músicas.

Para facilitar a distribuição e o uso da base de dados, cada áudio foi cortado em um pedaço de 30 segundos utilizando a biblioteca ffmpeg⁸. O corte de áudio foi feito encontrando o ponto central da música, e cortando os 15 segundos anteriores e os 15 segundos posteriores. As músicas que não possuíam um mínimo de 30 segundos foram descartadas. Apenas 8 músicas foram descartadas nessa etapa, resultando em 218,421 músicas.

Para coletar outros metadados e atributos associados a música usando a API do Spotify⁹, foi utilizado o método de busca que associa o nome do artista e o nome da música a um identificador utilizado pelo Spotify. Após a obtenção desse identificador, as informações foram obtidas por meio da API. Músicas que não foram encontradas no Spotify por meio da busca foram descartadas, resultando em um total de 136,390 músicas após essa etapa.

Para obter as letras das músicas, foi utilizada a aplicação Musixmatch¹⁰, por meio de um *crawler*. Das 136,390 músicas que possuíam todos os dados até este ponto, 111,924 músicas possuíam letra associada. Como a língua da letra pode ser uma informação importante na hora do processamento, foi utilizada a ferramenta langdetect¹¹ para inferir o idioma no qual a letra está escrita. Foram mantidas apenas músicas para as quais a ferramenta langdetect conseguiu inferir o idioma com uma probabilidade maior de 90%, resultando em um total de 109,269 músicas.

O passo final da construção da base de dados consistiu em filtrar do histórico de execução dos usuários as músicas que não estavam presentes no conjunto final das 109269

⁵<http://everynoise.com>

⁶<https://www.spotify.com>

⁷<https://www.youtube.com>

⁸<https://ffmpeg.org>

⁹<https://developer.spotify.com/documentation/web-api>

¹⁰<https://www.musixmatch.com>

¹¹<https://pypi.org/project/langdetect>

músicas que possuíam todas as informações. A versão final da base de dados *Music4All* contém 15602 usuários anonimizados, seus históricos de execuções e 109269 músicas, representadas pelos seus pedaços de áudios de 30 segundos, letras e seus demais atributos. A Tabela 4 apresenta um sumário de todos os atributos presentes nas músicas da base de dados *Music4All*.

Quadro 4: Atributos presentes para cada música na base de dados *Music4All*.

Atributo	Descrição
id	Identificador único de 16 caracteres para cada música na base de dados.
artist	Nome do artista que publicou a música no last.fm. Existem 16269 artistas únicos na base de dados.
song	Nome da música.
lang	Língua atribuída à letra da música pela ferramenta langdetect. Existem 46 línguas únicas na base de dados.
spotify_id	Identificador da música no Spotify.
popularity	Valor inteiro em um intervalo de 0 à 100 que representa o quão popular uma música é no Spotify. O valor é baseado em quantas vezes a música foi escutada e o quão recente ela foi ouvida.
album_name	Nome do álbum em que a música está presente. Existem 38363 álbuns diferentes na base de dados.
release	Ano no qual a música foi lançada.
danceability	Valor que está em um intervalo entre 0 e 1, e que representa o quão apropriada a música é para dançar. Esse valor é constituído de uma combinação de elementos musicais, disponibilizado pela API do Spotify.
energy	Valor que está em um intervalo entre 0 e 1, e que representa a medida perceptiva de intensidade e atividade de uma música, disponibilizado pela API do Spotify.
key	Tom geral da música utilizando a notação <i>Pitch Class</i> , disponibilizado pela API do Spotify.
mode	Valor binário que corresponde a escala da música, onde a escala maior possui o valor 1 e a escala menor possui o valor 0.
valence	Valor que está em um intervalo entre 0 e 1, e que representa o quão positiva é uma música.
tempo	Velocidade/Ritmo de uma música, medido em batidas por minuto.
genres	Lista de gêneros associados à música. Existem 853 gêneros únicos presentes na base de dados.
tags	Lista de <i>tags</i> fornecidas pelos usuários, com 19 541 <i>tags</i> únicas.

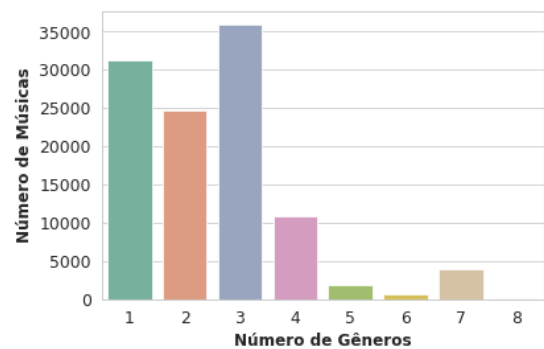
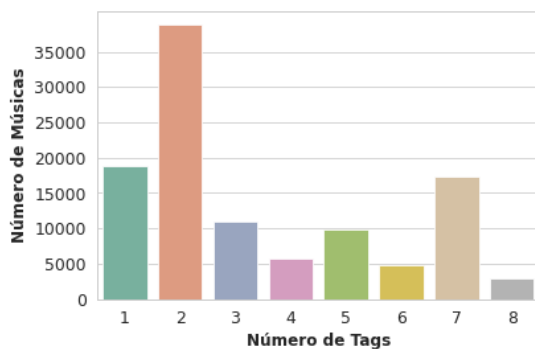
Fonte: Autoria própria.

A Figura 20 apresenta as estatísticas de distribuição de *tags* e gêneros por músicas na base de dados final. Como pode ser visto na Figura 20a, em geral, as músicas possuem uma grande quantidade de *tags* associadas a elas, sendo que a maioria das músicas tem duas ou mais *tags* associadas. O filtro de gênero nas *tags* utilizando o *Every Noise At Once* mostrou uma grande redução no número de *tags*, como pode ser visto na Figura 20b. É possível ver também que a maioria das músicas possui poucos gêneros associados.

Figura 20: Estatísticas de *tags* e gêneros na base de dados *Music4All*.

(a) Distribuição de *tags* por música

(b) Distribuição de gêneros por música



Fonte: Autoria própria.

A próxima seção apresentará a avaliação experimental realizada neste trabalho. Serão

descritas as estatísticas das duas bases de dados, os sistemas de recomendação, a metodologia de avaliação utilizada, e os resultados obtidos.

6 AVALIAÇÃO EXPERIMENTAL

Nesta seção são apresentadas as bases de dados, a metodologia de avaliação e as métricas utilizadas para avaliar o modelo proposto neste trabalho, assim como os resultados obtidos.

6.1 BASES DE DADOS

Neste trabalho foram utilizadas duas bases de dados para obter os *embeddings* (informação contextual) de músicas para sistemas de recomendação sensíveis ao contexto. Ambas as bases de dados contêm o histórico de execução de músicas para cada usuário junto com o horário em que a música foi ouvida, sem informações de sessões, similar ao que foi apresentado no Quadro 2.

A primeira base de dados foi proposta por Wang et al. (2018), por meio de um *crawler* da aplicação *Xiami Music*¹, e é aqui denominada como *Xiami*. A base de dados *Xiami* contém 4,284 usuários, na qual cada usuário possui 1,000 músicas ouvidas em seu histórico de execução. Dessas 1,000 músicas ouvidas, porém, em média, 370 são músicas únicas, o que indica que os usuários dessa base de dados ouvem muitas músicas repetidas.

A segunda base de dados, chamada de *Music4All*, foi criada durante a realização deste trabalho como uma segunda base de dados para avaliação do modelo proposto, como pode ser visto na Subseção 5.2. Ela possui o histórico de execução de músicas de 15,602 usuários da aplicação *last.fm*², obtida por meio da API da própria aplicação.

Em comparação com a base de dados *Xiami*, a *Music4All* possui um número muito maior de usuários, porém possui um número menor de músicas por usuário, com uma média de 361 músicas por usuário, sendo que 184 dessas músicas são músicas únicas. Com isso, podemos perceber que a razão de músicas únicas por usuário da base de dados *Music4All* é maior (0,51) que a da base de dados *Xiami* (0,37).

¹<https://www.xiami.com>

²<https://www.last.fm>

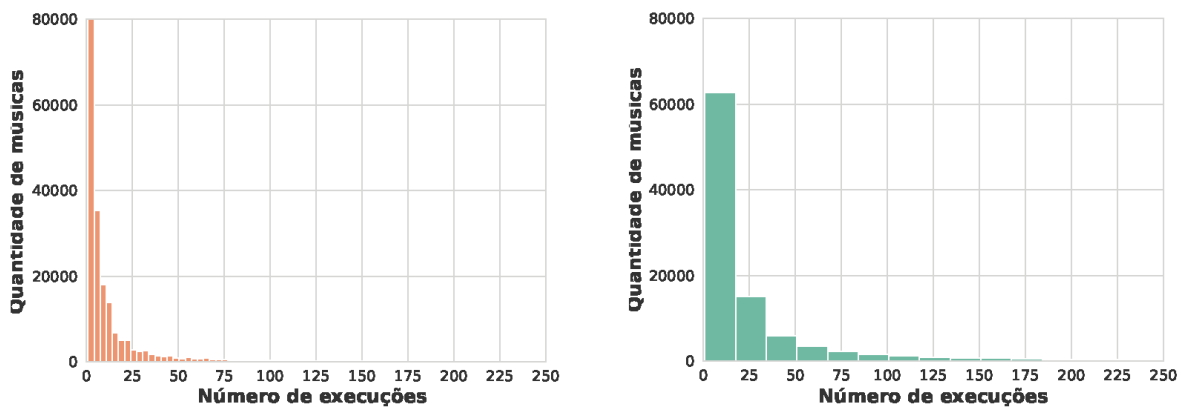
Com relação ao número total de músicas, a base de dados *Xiami* possui um número superior de músicas com relação a *Music4All*, com 361899 e 109269 músicas, respectivamente. Uma comparação entre as bases pode ser vista na Figura 21, com relação ao número de execuções que cada música possui. Tanto na Figura 21a quanto na Figura 21b é possível notar o efeito de cauda longa, efeito comum em bases de dados de sistemas de recomendação e já estudado por diversos autores (HERVAS-DRANE, 2008; FLEDER; HOSANAGAR, 2009).

O efeito de cauda longa corresponde a um comportamento no qual um pequeno grupo de itens, chamados de populares, corresponde a grande maioria do que é consumido, enquanto a grande maioria dos itens não possuem tantos acessos (CELMA, 2010).

Figura 21: Execuções por música em ambas as bases de dados.

(a) Número de execuções por música na base de dados *Xiami*.

(b) Número de execuções por música na base de dados *Music4all*.



Fonte: Autoria própria.

Como é necessário utilizar as sessões dos usuários para criar os vetores de *embeddings*, o histórico de execução dos usuários foi dividido em sessões levando em consideração o tempo passado entre uma música e a próxima música do histórico de execução. Neste trabalho, foi utilizado um valor de 30 minutos para separar as músicas em sessões em ambas as bases de dados, visto que com essa diferença era possível gerar um grande número de sessões que possuíam um número suficiente de músicas.

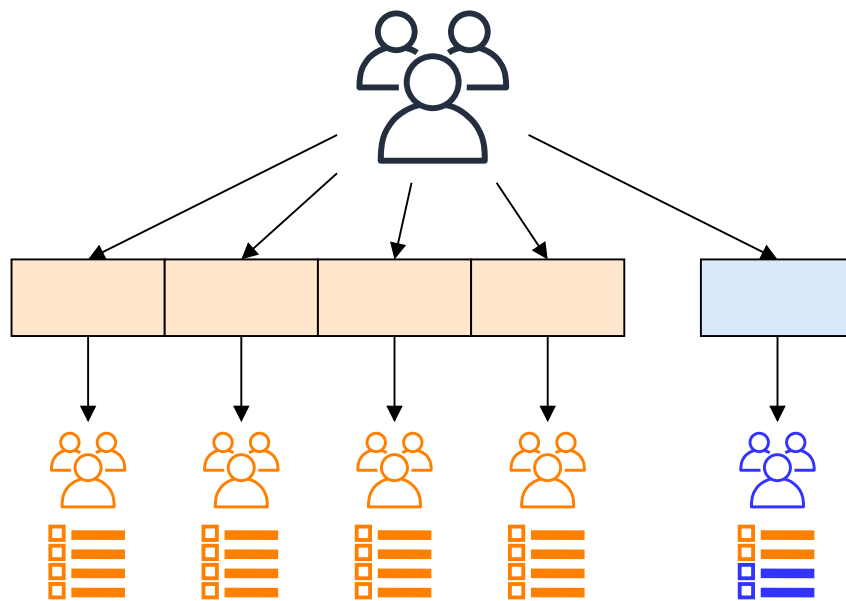
6.2 METODOLOGIA DE AVALIAÇÃO

Para avaliar se o modelo proposto nesta dissertação é capaz de obter resultados superiores ao *baseline*, os sistemas de recomendação apresentados na Subseção 4.3 foram executados nas bases de dados apresentadas na Seção 6.1, usando a abordagem de *k-fold cross-*

validation proposta por Wang et al. (2018).

Com o *k-fold cross-validation*, os usuários das bases de dados são divididos em k partições mutuamente exclusivas de mesmo tamanho, no qual uma dessas partições é escolhida como a partição de teste e as demais $k - 1$ partições são escolhidas como as partições de treino. O processo de escolha da partição de teste é realizado k vezes, sem repetir a partição que já foi escolhida (ALPAYDIN, 2004). A Figura 22 mostra a partição do conjunto de usuários em partições, assumindo o valor de $k = 5$, que foi o valor utilizado neste trabalho.

Figura 22: Processo de 5-fold cross-validation utilizado no trabalho.



Fonte: Autoria própria.

Os usuários que estão nas partições de treino usam todas as músicas de suas sessões para a construção de suas preferências (geral e contextual), como pode ser visto na Figura 22. Já os usuários que estão na partição de teste, utilizam a primeira metade de todas as suas sessões para a construção de suas preferências, e a segunda parte de cada sessão é utilizada como as músicas de teste.

Para avaliar as recomendações feitas pelos sistemas de recomendação, foi medido o desempenho de cada sessão de teste de cada usuário no conjunto de teste por meio de três métricas comumente utilizadas na área de sistemas de recomendação, que são *Precision*, *Recall* e *F-measure*. Além disso, foram calculadas duas métricas utilizadas para avaliar se a ordem em que as recomendações foram feitas pelo sistema de recomendação é satisfatória para o usuário, que são *Mean Average Precision* (MAP) e *Normalized Discounted Cumulative Gain* (NDCG) (JÄRVELIN; KEKÄLÄINEN, 2002).

A métrica *Precision* mede a proporção de recomendações satisfatórias dentre todas as recomendações, e indica a qualidade das recomendações produzidas com ênfase no sucesso das recomendações, e não nas falhas (CHUNG et al., 2018). Tomando como base o Quadro 5, a *Precision* é definida por Shani e Gunawardana (2011) como:

$$Precision = \frac{|tp|}{|tp| + |fp|}. \quad (23)$$

A métrica de *Recall* mede a proporção de recomendações dentre as músicas que os usuários de fato ouviram, e indica a qualidade das recomendações produzidas (CHUNG et al., 2018). A *Recall* é definida por Shani e Gunawardana (2011) como:

$$Recall = \frac{|tp|}{|tp| + |fn|}. \quad (24)$$

A métrica *F-measure* pode ser definida como a média harmônica entre os valores de *Precision* e *Recall*, e assim como as duas métricas que a compõe, possui uma variação de valor entre 0 e 1. Shani e Gunawardana (2011) definem a métrica *F-measure* da seguinte forma:

$$F-measure = 2 * \frac{Precision * Recall}{Precision + Recall}. \quad (25)$$

Quadro 5: Classificação de uma possível recomendação de uma música para um usuário.

	Música Recomendada	Música Não Recomendada
Ouvida	Verdadeiro Positivo (tp)	Falso Negativo (fn)
Não Ouvida	Falso Positivo (fp)	Verdadeiro Negativo (tn)

Fonte: Adaptado de Shani e Gunawardana (2011).

As métricas de avaliação que são responsáveis por avaliar a ordenação, MAP e NDCG, possuem objetivos diferentes. A métrica MAP tem como característica principal garantir que a ordem correta dos itens recomendados que aparecem primeiro na lista de recomendação seja recompensada, penalizando de maneira mais severa as recomendações erradas que aparecem no começo da lista (SHANI; GUNAWARDANA, 2011).

A métrica MAP pode ser calculada como uma média da métrica *Average Precision* (AP) para todas as recomendações feitas para um único usuário. A métrica AP pode ser calculada para uma lista de recomendações com N itens, sendo definida como (SHANI; GUNAWARDANA, 2011):

$$AP@N = \frac{1}{N} \sum_{k=1}^N P(k) \cdot \text{rel}(k), \quad (26)$$

onde $P(k)$ se refere a métrica *Precision* calculada para os primeiros k elementos da lista de recomendação e $\text{rel}(k)$ simboliza a operação de verificar se o item que está na lista de recomendações na posição k é o mesmo que está na posição k na lista de teste, retornando 1 caso seja verdadeiro e 0 caso seja falso.

A métrica NDCG, ao contrário da MAP, não favorece os itens que aparecem primeiro na lista de recomendação, mas tem como objetivo oferecer uma métrica que seja apropriada para longas listas de recomendações, onde a penalidade para as posições mais longes do começo da lista são aplicadas de maneira logarítmica (SHANI; GUNAWARDANA, 2011). Assumindo que um usuário u possui um ganho $g_{u,i}$ por ter-lhe sido recomendado um item i , a média da métrica *Discounted Cumulative Gain* (DCG) para uma lista de J itens pode ser definida como:

$$\text{DCG} = \frac{1}{N} \sum_{u=1}^N \sum_{j=1}^J \frac{g_{u,i_j}}{\log_b(j+1)}, \quad (27)$$

onde i_j representa o item i na posição j da lista. A base do logaritmo pode ser alterada, porém de maneira geral o valor escolhido é 2 ou 10. A NDCG é a versão normalizada da métrica DCG, e pode ser calculada da seguinte forma:

$$\text{NDCG} = \frac{\text{DCG}}{\text{DCG}^*}, \quad (28)$$

onde DCG^* corresponde à DCG calculada utilizando o conjunto de músicas que o usuário ouviu.

Com o intuito de garantir que os resultados sejam consistentes e não tenham ocorrido por acaso, foi utilizado o teste estatístico T de Student, com um nível de confiança de 95%.

6.3 RESULTADOS

Com o intuito de obter os melhores vetores de *embeddings* em ambas as bases de dados, diversos parâmetros foram testados buscando otimizar o modelo proposto. Todos os testes foram executados utilizando a linguagem de programação Python³ e a biblioteca de desenvolvimento de redes neurais Keras⁴. Os parâmetros utilizados pelo modelo proposto e seus valores que produziram o melhor resultado podem ser visto na Tabela 2.

³<https://www.python.org>

⁴<https://keras.io>

Tabela 2: Parâmetros utilizados para a obtenção dos resultados.

Base de Dados	Parâmetro	Valor
Ambas	Valor de Dropout	0.2
Ambas	Tamanho da Janela Contextual	3
<i>Xiami</i>	Quantidade de Unidades LSTM	256
<i>Music4All</i>	Quantidade de Unidades LSTM	512
<i>Xiami</i>	Tamanho do Vetor de Embeddings	256
<i>Music4All</i>	Tamanho do Vetor de Embeddings	1024

Fonte: Autoria Própria.

Para os modelos utilizados como *baseline*, foram utilizados os valores de parâmetro propostos por Wang et al. (2018), visto que foram os parâmetros com o qual foi possível obter os melhores resultados (Tabela 3).

Tabela 3: Parâmetros utilizados para a obtenção de *embeddings* propostos por Wang et al. (2018).

Parâmetro	Valor
Tamanho da Janela Contextual	5
Amostra Negativa	20
Down Sample	$1e^{-3}$
α	0.025
Número de Iterações	5

Fonte: Adaptado de Wang et al. (2018).

A janela contextual (apresentada na Subseção 4.2.1) que obteve melhores resultados foi de tamanho 3, isto é, com 3 músicas antes da música alvo e 3 músicas após. Isso indica que o modelo proposto não necessita de muitas músicas ao redor da música alvo para aprender vetores de *embeddings* capazes de superar o modelo *baseline*, tanto para vetores gerais quanto contextuais.

Vale mencionar, que apesar de possuir um número significativamente menor de músicas que a base *Xiami*, cerca de 30,19% menor, a base de dados *Music4All* necessitou de uma quantidade maior de unidades LSTM para obter bons resultados, assim como de um vetor de *embeddings* muito maior do que a base da *Xiami*.

Neste trabalho, foram obtidos resultados com 5 e 10 músicas recomendadas para cada usuário, com o objetivo de verificar se o comportamento observado seria o mesmo com um número menor e outro maior de recomendações. Os resultados da base de dados *Music4All* com 5 recomendações estão descritos na Tabela 4, enquanto os resultados da base de dados *Xiami* estão descritos na Tabela 5. Todos os resultados são estatisticamente significantes de acordo com o teste estatístico teste T de Student, usando um nível de confiança de 95%.

Como pode ser visto nas Tabelas 4 e 5, o modelo proposto neste trabalho foi capaz de aprender vetores de *embeddings* gerais superiores em ambas as bases de dados, como pode ser visto pelo resultado do sistema de recomendação M-TN, que utiliza apenas vetores de *embeddings* gerais. A maior melhora com relação as métricas pode ser observada nas métricas *Precision* e *F-measure*, com um aumento de cerca de 314% para o *baseline* na base de dados *Music4All*. Na base de dados *Xiami*, a maior melhora foi na métrica NDCG, com um aumento de 227,35%.

Tabela 4: Resultados dos sistemas de recomendação para a base de dados *Music4All* com 5 recomendações.

Modelo	Sistema de Recomendação	Precision	Recall	F-measure	MAP	NDCG
<i>Baseline</i>	M-TN	0,02503	0,02394	0,02083	0,06428	0,02642
Modelo Proposto	M-TN	0,10368	0,09538	0,08639	0,17986	0,10912
<i>Baseline</i>	SM-TN	0,11055	0,10929	0,09717	0,19622	0,13201
Modelo Proposto	SM-TN	0,12007	0,11653	0,10321	0,21566	0,13299
<i>Baseline</i>	CSM-TN	0,06205	0,06708	0,05516	0,13880	0,07907
Modelo Proposto	CSM-TN	0,12615	0,12135	0,10806	0,22307	0,13790
<i>Baseline</i>	CSM-UK	0,11000	0,10887	0,09671	0,19562	0,13151
Modelo Proposto	CSM-UK	0,11902	0,11556	0,10227	0,21386	0,13195

Fonte: Autoria Própria.

Tabela 5: Resultados dos sistemas de recomendação para a base de dados *Xiami* com 5 recomendações.

Modelo	Sistema de Recomendação	Precision	Recall	F-measure	MAP	NDCG
<i>Baseline</i>	M-TN	0,03851	0,02859	0,02573	0,13247	0,02921
Modelo Proposto	M-TN	0,08541	0,08684	0,06599	0,33463	0,09561
<i>Baseline</i>	SM-TN	0,11896	0,13206	0,09489	0,43057	0,15329
Modelo Proposto	SM-TN	0,17280	0,16742	0,12955	0,52695	0,18241
<i>Baseline</i>	CSM-TN	0,07078	0,07682	0,05524	0,27807	0,08694
Modelo Proposto	CSM-TN	0,16754	0,16700	0,12782	0,52862	0,18369
<i>Baseline</i>	CSM-UK	0,11758	0,13108	0,09394	0,42575	0,15220
Modelo Proposto	CSM-UK	0,17225	0,16711	0,12919	0,52615	0,18212

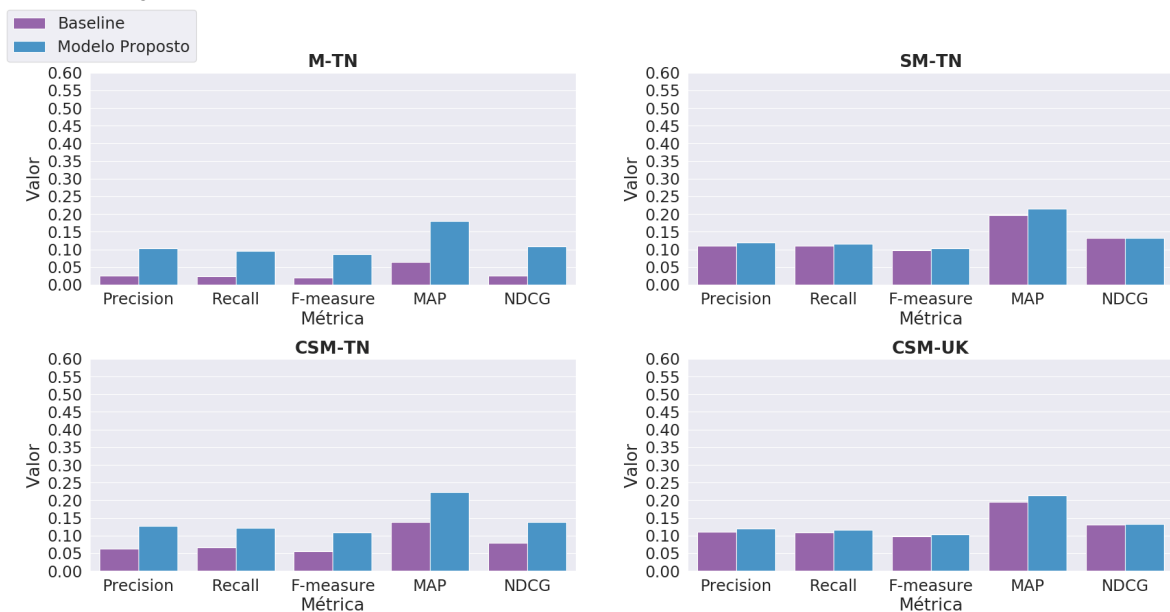
Fonte: Autoria Própria.

Com relação aos vetores de *embeddings* contextuais, foi percebida uma melhora com relação ao *baseline*, mas não tão significativa quanto a melhora dos vetores de *embeddings* gerais, em ambas as bases de dados, como pode ser visto no sistema de recomendação SM-TN. Com a base de dados *Xiami*, pode-se destacar um aumento de 36,53% para a métrica *F-measure*, assim como um aumento de 45,26% para a métrica *Precision*. As demais métricas possuíram um aumento de aproximadamente 20% com relação ao *baseline*. Para a base de dados *Music4All*, nenhuma métrica conseguiu um aumento superior a 10% com relação ao *baseline*, sendo a métrica MAP a que chegou mais perto, com uma melhora de 9,91%.

O sistema de recomendação CSM-TN faz uma combinação dos vetores de *embeddings* gerais e contextuais e, com relação ao *baseline*, foi possível obter uma melhora na métrica de *F-measure* de 96% e 131%, nas bases de dados *Music4All* e *Xiami*, respectivamente. Apesar de ter conseguido uma melhora significativa com relação ao *baseline*, a soma dos vetores de *embeddings* não se traduziu em valores superiores aos sistemas de recomendação que utilizam apenas o vetor de *embeddings* contextual, como pode ser visto na base de dados *Xiami* (Tabela 5). No entanto, houve uma pequena melhora na base de dados *Music4All*, com um aumento de 4,74% na métrica *F-measure*, mas não é possível dizer que a soma dos vetores produziu melhores resultados do que a utilização apenas de vetores de *embeddings* contextuais.

O último sistema de recomendação, CSM-UK, assim como o CSM-TN, também obteve melhorias com relação aos resultados obtidos pelo *baseline*, sendo os resultados mais expressivos na base de dados *Xiami*, como uma melhora de 38% na métrica *F-measure*. Para a base de dados *Music4All*, o aumento com relação ao *baseline* não passou de 10% em nenhuma das métricas. Esse fato indica que a informação dos usuários similares não foi relevante para nenhuma das bases de dados, visto que o CSM-UK é uma adaptação do sistema de recomendação UserKNN, apresentado na Seção 2.1. A melhora dos sistemas de recomendação em ambas as bases de dados com 5 recomendações pode também ser visualizada nas Figuras 23 e 24.

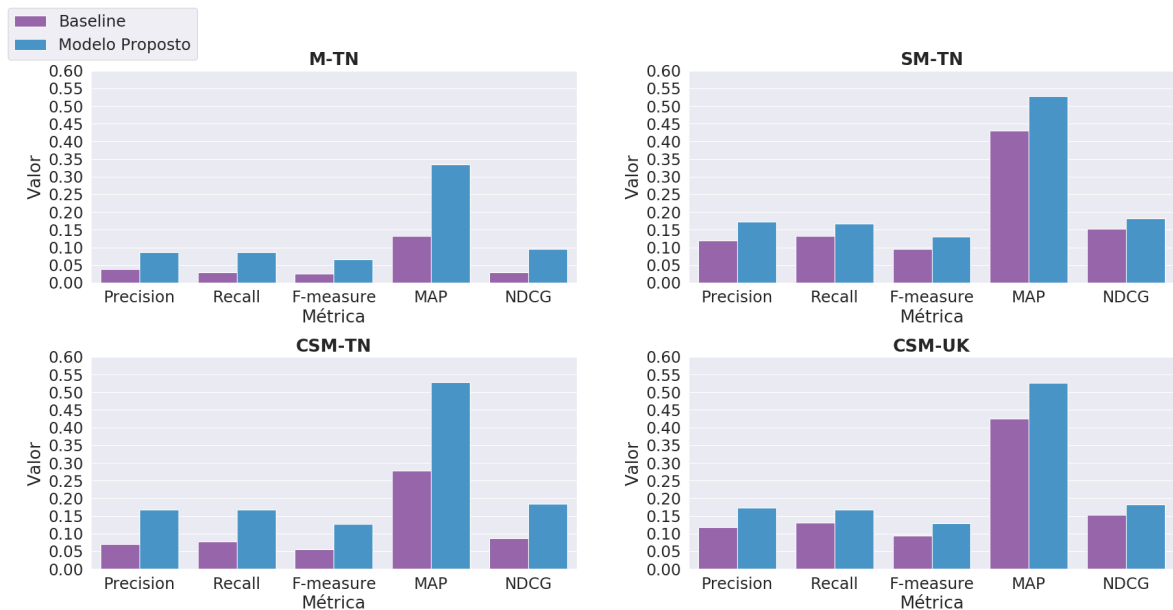
Figura 23: Resultados dos sistemas de recomendação para a base de dados *Music4All* com 5 recomendações.



Fonte: Autoria Própria.

Com 10 recomendações geradas pelos sistemas de recomendação, apenas a base de

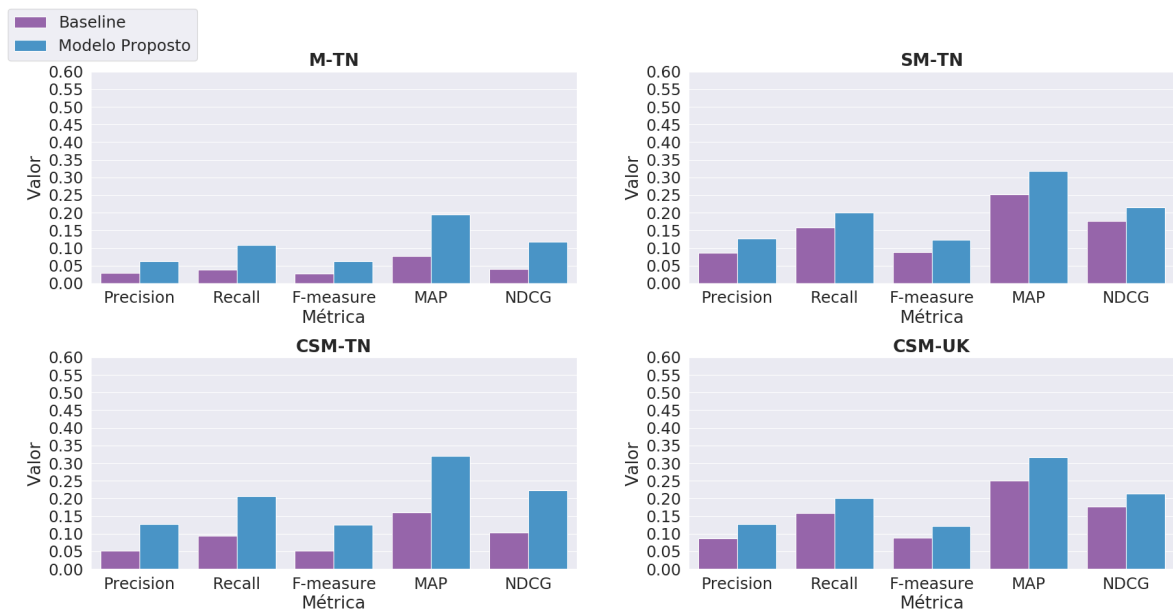
Figura 24: Resultados dos sistemas de recomendação para a base de dados *Xiami* com 5 recomendações.



Fonte: Autoria Própria.

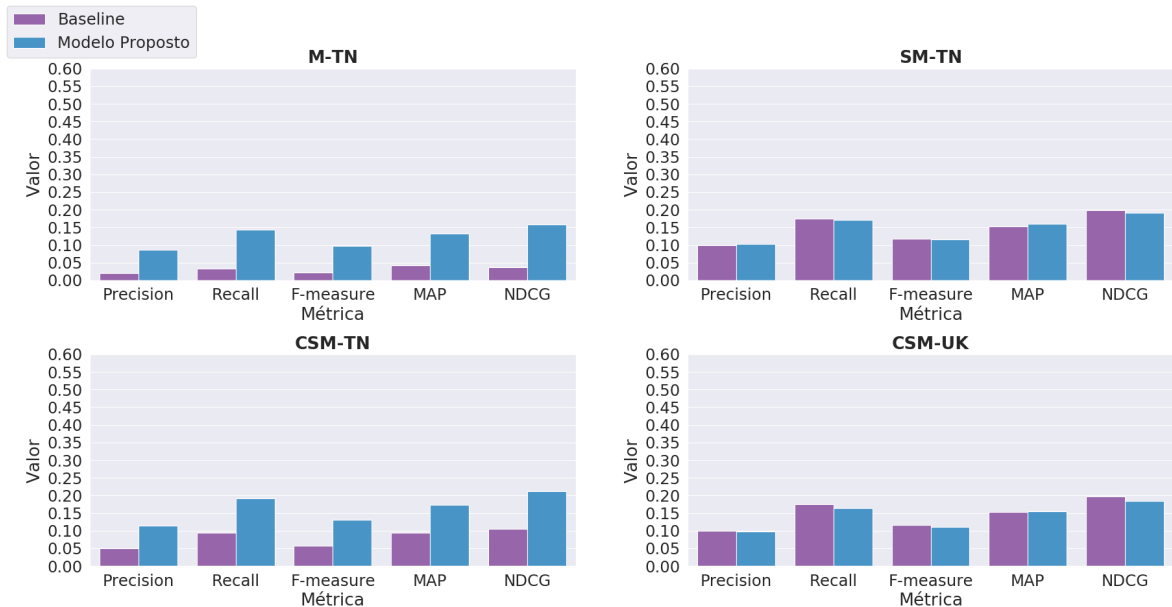
dados *Xiami* conseguiu uma melhora em todas as métricas com relação ao *baseline*, para todos os sistemas de recomendação. As Figuras 25 e 26 apresentam os resultados de todas as métricas para todos os sistemas com 10 recomendações.

Figura 25: Resultados dos sistemas de recomendação para a base de dados *Xiami* com 10 recomendações.



Fonte: Autoria Própria.

Figura 26: Resultados dos sistemas de recomendação para a base de dados *Music4All* com 10 recomendações.



Fonte: Autoria Própria.

Na base de dados *Music4All*, como pode ser visto na Tabela 6, foi possível obter melhoras significativas nos sistemas de recomendação M-TN e CSM-TN, indicando que os vetores de *embeddings* gerais obtidos pelo modelo proposto são capazes de gerar boas recomendações. Para o sistema de recomendação M-TN, foi obtido melhoras de até 344,28% na métrica *F-measure*, e 335,19% na métrica NDCG.

Para o sistema de recomendação SM-TN, as únicas métricas que conseguiram um aumento foram as métricas *Precision* e MAP, com um aumento de 2,64% e 4,85%, respectivamente. No entanto, para o sistema de recomendação CSM-UK, o único aumento foi na métrica MAP, com uma diferença de 1%.

Tabela 6: Resultados dos sistemas de recomendação para a base de dados *Music4All* com 10 recomendações.

Modelo	Sistema de Recomendação	Precision	Recall	F-measure	MAP	NDCG
Baseline	M-TN	1,97%	3,36%	2,19%	4,27%	3,62%
Modelo Proposto	M-TN	8,70%	14,35%	9,75%	13,31%	15,76%
Baseline	SM-TN	9,94%	17,47%	11,68%	15,27%	19,81%
Modelo Proposto	SM-TN	10,20%	17,10%	11,56%	16,01%	19,13%
Baseline	CSM-TN	4,90%	9,30%	5,73%	9,36%	10,42%
Modelo Proposto	CSM-TN	11,45%	19,17%	13,03%	17,33%	21,09%
Baseline	CSM-UK	9,92%	17,45%	11,66%	15,26%	19,79%
Modelo Proposto	CSM-UK	9,78%	16,46%	11,07%	15,46%	18,35%

Fonte: Autoria Própria.

Com 10 recomendações geradas com a base de dados *Xiami*, o comportamento foi similar ao obtido com 5 recomendações. Como pode ser visto na Tabela 7, todas as métricas para os sistemas de recomendação M-TN e CSM-TN apresentaram uma melhora em comparação ao *baseline* de mais de 100%, mostrando que não só os vetores de *embeddings* gerais quanto a combinação dos vetores gerais e contextuais se mostraram superiores.

Tabela 7: Resultados dos sistemas de recomendação para a base de dados *Xiami* com 10 recomendações.

Modelo Utilizado	Recsys	Precision	Recall	F-measure	MAP	NDCG
<i>Baseline</i>	M-TN	2,97%	3,89%	2,74%	7,71%	3,95%
Modelo Proposto	M-TN	6,22%	10,89%	6,30%	19,43%	11,76%
<i>Baseline</i>	SM-TN	8,64%	15,86%	8,83%	25,12%	17,64%
Modelo Proposto	SM-TN	12,69%	20,08%	12,24%	31,74%	21,45%
<i>Baseline</i>	CSM-TN	5,14%	9,44%	5,20%	15,99%	10,27%
Modelo Proposto	CSM-TN	12,69%	20,70%	12,49%	32,09%	22,24%
<i>Baseline</i>	CSM-UK	8,60%	15,81%	8,79%	25,08%	17,59%
Modelo Proposto	CSM-UK	12,66%	20,04%	12,20%	31,69%	21,41%

Fonte: Autoria Própria.

Para o sistema de recomendação SM-TN, pode-se destacar a melhora nas métricas *Precision* e *F-measure*, visto que foram as únicas métricas que possuíram uma melhora superior a 30% com relação ao *baseline*, com 46,85% e 38,61%, respectivamente. Assim como o sistema de recomendação SM-TN, o sistema de recomendação CSM-UK obteve uma melhora similar ao do SM-TN, evidenciando que a informação dos usuários similares não foram úteis para realizar a recomendação.

A partir dos resultados apresentados nesta subseção, é possível concluir que o modelo proposto obteve resultados superiores ao do *baseline* em ambas as bases de dados com 5 recomendações geradas por todos os sistemas de recomendação. Para 10 recomendações, apenas os sistemas de recomendação SM-TN e CSM-UK não conseguiram um desempenho superior em todas as métricas, porém os *embeddings* obtidos pelo modelo se mostraram superiores nos demais sistemas de recomendação.

Na próxima seção, serão apresentadas as conclusões deste trabalho e possíveis direções para trabalhos futuros.

7 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho de mestrado foi proposta uma Rede Neural Recorrente com o objetivo de gerar vetores de *embeddings* (informação contextual) para sistemas de recomendação de músicas sensíveis ao contexto. O modelo proposto possui duas redes *Long Short-Term Memory* em paralelo, mas que fazem parte do mesmo processo de treinamento, para aprender os vetores de *embeddings* gerais e contextuais de cada música, explorando tanto o histórico de execução completo do usuário quanto as suas sessões.

Além da proposta da Rede Neural Recorrente, também foi criada uma base de dados que contém uma grande variedade de informações, além do histórico de execuções de cada usuário. A base de dados, chamada de *Music4All*, possui 15602 usuários, assim como o histórico de execução de cada um, e 109269 músicas, com informações básicas como o nome do artista, álbum e o ano de lançamento, e informações que podem ser utilizadas em outras áreas de pesquisa, como o áudio e a letra da música, além de outros atributos.

A partir dos resultados obtidos com a avaliação experimental, foi possível concluir que o modelo proposto supera o *baseline*, um modelo considerado estado da arte na recomendação contextual de músicas, com os vetores gerais e contextuais, quando são feitas 5 recomendações para cada usuário. Em ambas as bases de dados utilizadas, o modelo proposto foi capaz de gerar recomendações melhores, com melhoras de até 314,74% em métricas como a *F-measure*, para a recomendação de músicas utilizando os vetores gerais na base de dados *Music4All*.

Já com 10 recomendações, a única métrica que conseguiu uma melhora em todos os sistemas de recomendação foi a MAP, na base de dados *Music4All*, que avalia a ordem na qual as músicas foram recomendadas, indicando que independente dos resultados, a ordenação das recomendações ainda foi superior. Para a base de dados *Xiami*, no entanto, os resultados indicaram uma melhora menor nos vetores de *embeddings* gerais do que a base *Music4All*, mas uma melhora maior para os vetores de *embeddings* contextuais.

É importante considerar, no entanto, que o modelo proposto apesar de apresentar melhores resultados, possui um tempo de execução maior e necessita de mais processamento

do que o *baseline*. Para ambas as bases de dados, o *baseline* demorou aproximadamente 20 minutos para construção do modelo, enquanto o modelo proposto demorou aproximadamente 1 dia para a base de dados *Music4All*, e 2 dias para a base de dados *Xiami*, devido a quantidade de músicas. No entanto, por serem modelos *offline*, ou seja, que só precisam ser executados quando houver uma atualização na base de dados, o tempo de execução dos modelos não afeta as recomendações. O tempo de execução foi calculado utilizando um computador que possui um processador Intel(R) Core(TM) i7-7800X, 96 GB de memória RAM, uma placa de vídeo TITAN V e uma GeForce GTX 1080.

Assim, é possível concluir que a utilização de Redes Neurais Recorrentes se mostra eficaz para a aquisição de vetores de *embeddings* de músicas, considerando o histórico de execução completo ou as sessões dos usuários. Em ambas as bases de dados, os resultados se mostraram superiores ao do *baseline* nos sistemas de recomendação que utilizam os vetores de *embeddings* gerais e a combinação dos vetores de *embeddings* gerais e contextuais.

Vale ressaltar que os vários resultados obtidos com presente trabalho de mestrado foram submetidos a duas conferências e uma revista, sendo que, um deles já foi publicado e os outros dois se encontram em processo de avaliação, como apresentado a seguir:

- Igor André Pegoraro Santana, Fabio Pinhelli, Juliano Donini, Leonardo Catharin, Rafael Biazus Mangolin, Yandre Maldonado e Gomes da Costa, Valéria Delisandra Feltrim, Marcos Aurélio Domingues. *Music4All: A New Music Database and Its Applications. In proceedings of the 27th International Conference on Systems, Signals and Image Processing (IWSSIP 2020)*. [Publicado]
- Igor André Pegoraro Santana, Marcos Aurélio Domingues. *Improving Context-Aware Music Recommender Systems with a Dual Recurrent Neural Network. In proceedings of the 7th International Conference on Information Management and Big Data (SIMBig 2020)*. [Em avaliação]
- Igor André Pegoraro Santana, Marcos Aurélio Domingues. *Improving Context-Aware Music Recommendation with Metadata Awareness and Recurrent Neural Networks. Information Retrieval Journal*. [Em avaliação]

Durante o mestrado também foram realizados trabalhos paralelos que possibilitaram a seguinte publicação:

- Igor André Santana, Abner Suniga, Juliano Donini, Camila Vaccari Sundermann, Solange Oliveira Rezende, Marcos Aurélio Domingues. *Transforming Geo-Referenced Data*

in Contextual Information for Context-Aware Recommender Systems. In proceedings of the 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2018).
[Publicado]

Como trabalhos futuros, apesar dos resultados serem promissores, ainda há possibilidades para aprimorar o modelo proposto. Um dos principais pontos de melhora do modelo proposto está em sua arquitetura, onde o objetivo principal seria reduzir a quantidade de memória e processamento utilizados. Para isso, uma possibilidade seria a utilização de um outro modelo de Rede Neural Recorrente, chamado *Gated Recurrent Unit* (GRU). As unidades GRU possuem um desempenho similar as unidades LSTM, porém com um custo de memória e processamento menor. Outra possibilidade seria executar um processo de *grid-search* para otimizar os hiperparâmetros do modelo, para que não seja necessário vetores de *embeddings* tão grandes. Além das otimizações possíveis para o modelo, seria possível também utilizar o modelo já treinado, utilizando como entrada as músicas anteriores de acordo com o histórico de execução e com a sessão, e utilizar um processo recursivo para recomendar a próxima música para o usuário. Por fim, pretende-se avaliar o modelo proposto em outros domínios de aplicação, como, por exemplo, na recomendação de pontos de interesse (POIs).

REFERÊNCIAS

- ADOMAVICIUS, G. et al. Incorporating contextual information in recommender systems using a multidimensional approach. **ACM Transactions on Information Systems**, ACM, v. 23, n. 1, p. 103–145, jan 2005.
- ADOMAVICIUS, G.; TUZHILIN, A. Context-Aware Recommender Systems. In: RICCI, F.; ROKACH, L.; SHAPIRA, B. (Ed.). **Recommender Systems Handbook**. Boston, MA: Springer US, 2015. p. 191–226.
- ALPAYDIN, E. Design and Analysis of Machine Learning Experiments. **Introduction to Machine Learning**, p. 474–515, 2004.
- BALDI, P. Autoencoders, Unsupervised Learning and Deep Architectures. In: **Proceedings of the 2011 International Conference on Unsupervised and Transfer Learning Workshop - Volume 27**. [S.l.]: JMLR.org, 2011. (UTLW'11), p. 37–50.
- BENGIO, Y. et al. A Neural Probabilistic Language Model. In: **Journal of Machine Learning Research**. [S.l.: s.n.], 2003.
- BENGIO, Y.; SIMARD, P.; FRASCONI, P. Learning Long-Term Dependencies with Gradient Descent is Difficult. **IEEE Transactions on Neural Networks**, v. 5, n. 2, p. 157–166, mar 1994.
- BEUTEL, A. et al. Latent Cross: Making Use of Context in Recurrent Recommender Systems. In: **Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining**. New York, New York, USA: ACM Press, 2018. p. 46–54.
- BLEI, D. M.; NG, A. Y.; JORDAN, M. I. Latent Dirichlet allocation. **Journal of Machine Learning Research**, v. 3, p. 993–1022, 2003.
- BOBADILLA, J. et al. Recommender systems survey. **Knowledge-Based Systems**, v. 46, p. 109–132, 2013.
- BU, J. et al. Music recommendation by unified hypergraph: Combining social media information and music content. In: **Proceedings of the ACM Multimedia 2010 International Conference**. New York, New York, USA: ACM Press, 2010. p. 391–400.
- BURKE, R. Knowledge-Based Recommender Systems. **Encyclopedia of library and information systems**, v. 69, 2000.
- CELMA, Ò. **Music Recommendation and Discovery**. 1. ed. [S.l.]: Springer-Verlag Berlin Heidelberg, 2010. 194 p.
- CHEN, H. C.; CHEN, A. L. A music recommendation system based on music and user grouping. In: **Journal of Intelligent Information Systems**. New York, NY, USA: ACM, 2005. v. 24, n. 2-3, p. 113–132.

- CHO, K. et al. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In: **Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation**. [S.l.: s.n.], 2015. p. 103–111.
- CHUNG, Y. et al. Improved neighborhood search for collaborative filtering. **International Journal of Fuzzy Logic and Intelligent Systems**, the Korean Institute of Intelligent Systems, v. 18, n. 1, p. 29–40, mar 2018.
- COVINGTON, P.; ADAMS, J.; SARGIN, E. Deep neural networks for youtube recommendations. In: **Proceedings of the 10th ACM Conference on Recommender Systems**. New York, New York, USA: ACM Press, 2016. p. 191–198.
- Da Costa, F. S.; DOLOG, P. Collective embedding for neural context-aware recommender systems. In: **Proceedings of the 13th ACM Conference on Recommender Systems**. [S.l.: s.n.], 2019. p. 201–209.
- DEERWESTER, S. et al. Indexing by latent semantic analysis. **Journal of the American Society for Information Science**, v. 41, n. 6, p. 391–407, 1990.
- DENG, S. et al. Exploring user emotion in microblogs for music recommendation. **Expert Systems with Applications**, v. 42, n. 23, p. 9284–9293, 2015.
- EPPLER, M. J.; MENGIS, J. The concept of information overload: A review of literature from organization science, accounting, marketing, MIS, and related disciplines. **Information Society**, Routledge, v. 20, n. 5, p. 325–344, 2004.
- FELFERNIG, A. et al. Constraint-based recommender systems. In: **Recommender Systems Handbook, Second Edition**. [S.l.: s.n.], 2015. p. 161–190.
- FLEDER, D.; HOSANAGAR, K. Blockbuster culture’s next rise or fall: The impact of recommender systems on sales diversity. **Management Science**, v. 55, n. 5, p. 697–712, 2009.
- GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning Title**. [S.l.]: The MIT Press, 2016. 777 p.
- GRAVES, A. Neural Networks. In: **Supervised Sequence Labelling with Recurrent Neural Networks**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 15–35.
- GUTMANN, M.; HYVÄRINEN, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In: **Journal of Machine Learning Research**. [S.l.: s.n.], 2010. v. 9, p. 297–304.
- HAYKIN, S. **Neural Networks and Learning Machines**. 3. ed. Ontario, Canada: Pearson Education, 2008. 906 p.
- HERVAS-DRANE, A. Word of Mouth and Recommender Systems : A Theory of the Long Tail. **Business**, p. 1–48, 2008.
- HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, 1997.
- IFPI. **Global Music Report 2019 - Annual State of the Industry**. [S.l.], 2019.

- JANNACH, D. et al. **Recommender systems: An introduction**. [S.l.: s.n.], 2010. 1–335 p.
- JÄRVELIN, K.; KEKÄLÄINEN, J. Cumulated gain-based evaluation of IR techniques. **ACM Transactions on Information Systems**, Association for Computing Machinery, New York, NY, USA, v. 20, n. 4, p. 422–446, oct 2002.
- KAMINSKAS, M.; RICCI, F. Contextual music information retrieval and recommendation: State of the art and challenges. **Computer Science Review**, v. 6, n. 2-3, p. 89–119, 2012.
- KIM, J.-Y.; BELKIN, N. J.; BELKIN, J. Categories of music description and search terms and phrases used by non-music experts. In: **Proceedings of the Third International Conference on Music Information Retrieval**. Paris: [s.n.], 2002. p. 209–214.
- KITCHENHAM, B. **Procedures for performing systematic reviews**. Department of Computer Science, Keele University, UK, 2004. 28 p.
- KOENIGSTEIN, N.; DROR, G.; KOREN, Y. Yahoo! Music Recommendations: Modeling Music Ratings with Temporal Dynamics and Item Taxonomy. In: **Proceedings of the 5th ACM Conference on Recommender Systems**. New York, NY, USA: ACM, 2011. p. 165–172.
- LECUN, Y. et al. Object recognition with gradient-based learning. In: **Lecture Notes in Computer Science**. [S.l.: s.n.], 1999. v. 1681, p. 319–345.
- LI, Z. et al. Learning from history and present: Next-item recommendation via discriminatively exploiting user behaviors. In: **Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2018. (KDD '18), p. 1734–1743.
- LIU, J.; WU, C. Deep learning based recommendation: A survey. In: **Lecture Notes in Electrical Engineering**. [S.l.]: Springer, Singapore, 2017. v. 424, p. 451–458.
- LIU, Z.; XIE, X.; CHEN, L. Context-aware academic collaborator recommendation. In: **Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [S.l.: s.n.], 2018.
- LOPS, P.; GEMMIS, M. de; SEMERARO, G. Content-based Recommender Systems: State of the Art and Trends. In: RICCI, F. et al. (Ed.). **Recommender Systems Handbook**. 1. ed. [S.l.]: Springer US, 2011. cap. 3.
- MAYERL, M. et al. Language models for next-track music recommendation. In: **CEUR Workshop Proceedings**. [S.l.: s.n.], 2019. v. 2367, p. 15–19.
- MIKOLOV, T. et al. Efficient estimation of word representations in vector space. In: **Workshop Track Proceedings of the 1st International Conference on Learning Representations, ICLR 2013**. [S.l.: s.n.], 2013.
- MIKOLOV, T. et al. Distributed representations of words and phrases and their compositionality. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2013.
- NING, X.; DESROSIERS, C.; KARYPIS, G. A comprehensive survey of neighborhood-based recommendation methods. In: **Recommender Systems Handbook, Second Edition**. [S.l.: s.n.], 2015. p. 37–76.

- PALMISANO, C.; TUZHILIN, A.; GORGOGNONE, M. Using context to improve predictive modeling of customers in personalization applications. **IEEE Transactions on Knowledge and Data Engineering**, v. 20, n. 11, p. 1535–1549, 2008.
- PASCANU, R.; MIKOLOV, T.; BENGIO, Y. **On the difficulty of training recurrent neural networks**. 2013. 2347–2355 p.
- PAZZANI, M. J.; BILLSUS, D. Content-based recommendation systems. In: BRUSILOVSKY, P.; KOBASA, A.; NEJDL, W. (Ed.). **Lecture Notes in Computer Science**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. v. 4321 LNCS, p. 325–341.
- RAZA, S.; DING, C. Progress in context-aware recommender systems - An overview. **Computer Science Review**, v. 31, p. 84–97, 2019.
- RESNICK, P. et al. GroupLens: An open architecture for collaborative filtering of netnews. In: **Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work**. [S.l.: s.n.], 1994. p. 175–186.
- RESNICK, P.; VARIAN, H. R. Recommender Systems. **Communications of the ACM**, ACM, New York, NY, USA, v. 40, n. 3, p. 56–58, 1997.
- RICCI, F.; SHAPIRA, B.; ROKACH, L. Recommender systems: Introduction and challenges. In: **Recommender Systems Handbook, Second Edition**. [S.l.: s.n.], 2015. p. 1–34.
- ROGERS, T. T.; MCCLELLAND, J. L. Parallel distributed processing at 25: Further explorations in the microstructure of cognition. In: RUMELHART, D. E.; MCCLELLAND, J. L.; PDP Research Group, C. (Ed.). **Cognitive Science**. Cambridge, MA, USA: MIT Press, 2014. v. 38, n. 6, cap. Distribute, p. 1024–1077.
- ROJAS, R. The Biological Paradigm. In: **Neural Networks**. [S.l.]: Springer Berlin Heidelberg, 1996. p. 3–27.
- RUMELHART, D. E.; MACCLELLAND, J. L. Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Foundations. In: RUMELHART, D. E.; MCCLELLAND, J. L.; PDP Research Group, C. (Ed.). **MIT Press**. Cambridge, MA, USA: MIT Press, 1986. v. 1, n. June, cap. Learning I, p. 318 — 362.
- SANTANA, I. A. P. et al. Music4All: A New Music Database and Its Applications. In: **Proceedings of the 27th International Conference on Systems, Signals and Image Processing (IWSSIP 2020)**. [S.l.: s.n.], 2020.
- SARWAR, B. et al. Item-based collaborative filtering recommendation algorithms. In: **Proceedings of the 10th International Conference on World Wide Web**. [S.l.: s.n.], 2001. p. 285–295.
- SCHAFER, J. B. et al. Collaborative filtering recommender systems. In: **Lecture Notes in Computer Science**. [S.l.: s.n.], 2007.
- SHANI, G.; GUNAWARDANA, A. Evaluating Recommendation Systems. In: RICCI, F. et al. (Ed.). **Recommender Systems Handbook**. 1. ed. Boston, MA: Springer US, 2011. cap. 8, p. 257–297.

- SMIRNOVA, E.; VASILE, F. Contextual sequence modeling for recommendation with Recurrent Neural Networks. In: **ACM International Conference Proceeding Series**. [S.l.: s.n.], 2017. Part F130153, p. 2–9.
- SUTSKEVER, I. Training recurrent neural networks. University of Toronto, 2013.
- TAN, J.; WAN, X.; XIAO, J. A Neural Network Approach to Quote Recommendation in Writings. In: **Proceedings of the 25th ACM International on Conference on Information and Knowledge Management**. New York, NY, USA: ACM, 2016. (CIKM '16), p. 65–74.
- TAN, Y. K.; XU, X.; LIU, Y. Improved recurrent neural networks for session-based recommendations. In: **ACM International Conference Proceeding Series**. [S.l.: s.n.], 2016.
- TANG, X.; WAN, X.; ZHANG, X. Cross-language context-aware citation recommendation in scientific articles. In: **Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval**. [S.l.: s.n.], 2014. p. 817–826.
- TILAHUN, B.; AWONO, C.; BATCHAKUI, B. A Survey of State-of-the-art: Deep Learning Methods on Recommender System. **International Journal of Computer Applications**, v. 162, n. 10, p. 17–22, 2017.
- UNGER, M. Latent context-aware recommender systems. In: **Proceedings of the 9th ACM Conference on Recommender Systems**. New York, NY, USA: ACM, 2015. (RecSys '15), p. 383–386.
- VÖTTER, M. et al. Autoencoders for next-track-recommendation. In: **CEUR Workshop Proceedings**. [S.l.: s.n.], 2019. v. 2367, p. 20–25.
- WANG, D.; DENG, S.; XU, G. Sequence-based context-aware music recommendation. **Information Retrieval Journal**, Springer Netherlands, v. 21, n. 2-3, p. 230–252, jun 2018.
- WANG, D. et al. Learning music embedding with metadata for context aware recommendation. In: **Proceedings of the 2016 ACM International Conference on Multimedia Retrieval**. New York, NY, USA: ACM, 2016. (ICMR '16), p. 249–253.
- WANG, S.; CAO, L.; WANG, Y. **A Survey on Session-based Recommender Systems**. 2019.
- WERBOS, P. J. Backpropagation Through Time: What It Does and How to Do It. **Proceedings of the IEEE**, v. 78, n. 10, p. 1550–1560, oct 1990.
- XIA, H. et al. Leveraging ratings and reviews with gating mechanism for recommendation. In: **International Conference on Information and Knowledge Management, Proceedings**. [S.l.: s.n.], 2019. p. 1573–1582.
- XIE, M. et al. Learning Graph-based POI Embedding for Location-based Recommendation. In: **Proceedings of the 25th ACM International on Conference on Information and Knowledge Management**. New York, NY, USA: ACM, 2016. (CIKM '16), p. 15–24.
- XU, W.; RUDNICKY, A. Can artificial neural networks learn language models? In: **Proceedings of the 6th International Conference on Spoken Language Processing**. [S.l.: s.n.], 2000.

YANG, C. et al. Bridging collaborative filtering and semi-supervised learning: A neural approach for POI recommendation. In: **Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. [S.l.: s.n.], 2017.

ZHANG, L.; LIU, P.; GULLA, J. A. A deep joint network for session-based news recommendations with contextual augmentation. In: **Proceedings of the 29th ACM Conference on Hypertext and Social Media**. New York, New York, USA: ACM Press, 2018. p. 201–209.

ZHANG, S. et al. Deep learning based recommender system: A survey and new perspectives. **ACM Computing Surveys**, v. 52, n. 1, 2019.

ZHAO, Q. et al. From preference into decision making: Modeling user interactions in recommender systems. In: **Proceedings of the 13th ACM Conference on Recommender Systems**. [S.l.: s.n.], 2019. p. 29–33.

ZHAO, S. et al. Geo-Teaser: Geo-Temporal Sequential Embedding Rank for Point-of-interest Recommendation. In: **Proceedings of the 26th International Conference on World Wide Web Companion**. New York, New York, USA: ACM Press, 2017. p. 153–162.

APÊNDICE A – PROTOCOLO DA REVISÃO SISTEMÁTICA

Diversos métodos de pesquisas podem ser utilizados para se entender como está o estado da arte de determinada área de pesquisa. É possível pesquisar *online* por artigos, pesquisar em bibliotecas de universidades, ou encontrar as melhores revistas e conferências e estudar seus melhores artigos. Uma revisão sistemática é um método para revisar, identificar, avaliar e entender a pesquisa disponível em uma determinada área. Uma revisão sistemática, no entanto, revisa apenas trabalhos que são chamados de primários: trabalhos que modificam o estado da arte de uma área de pesquisa.

Como descrito por Kitchenham (2004), uma revisão sistemática possui três fases: planejamento, condução e relatório. A primeira fase é responsável por definir um protocolo de pesquisa, que contém as estratégias de pesquisa e as questões essenciais de pesquisa que o protocolo deve responder. A segunda fase é responsável por executar o protocolo definido na primeira fase e extrair os resultados das fontes de dados. O último passo consiste em interpretar e reportar os resultados.

O objetivo da revisão sistemática realizada neste trabalho é entender quais modelos estão sendo utilizados para obter *embeddings* para sistemas de recomendação sensíveis ao contexto. Como visto em (KITCHENHAM, 2004), toda revisão sistemática deve conter questões que as referências obtidas devem responder. Nesse trabalho, as questões são as seguintes:

- Quais modelos estão sendo utilizados para gerar *embeddings* para sistemas de recomendação sensíveis ao contexto?
- Em que domínios de pesquisa esses modelos estão sendo utilizados?

Dada a definição das questões de pesquisa, a fase de planejamento da revisão sistemática começou com uma revisão dos trabalhos secundários já publicados sobre o assunto. Trabalhos secundários são trabalhos que, assim como uma revisão sistemática, compilam trabalhos primários (mas sem a necessidade de um protocolo). Existem diversos trabalhos que

sintetizam trabalhos feitos com sistemas de recomendação tradicionais utilizando técnicas de *deep learning* (ZHANG et al., 2019; LIU; WU, 2017; TILAHUN et al., 2017), porém não foi encontrado na literatura trabalhos com o mesmo objetivo de pesquisa dessa revisão sistemática. Alguns trabalhos foram feitos sobre sistemas de recomendação sensíveis ao contexto com *deep learning* (RAZA; DING, 2019; WANG et al., 2019), porém não focam no contexto de aquisição de *embeddings*.

Após definir as questões do protocolo, foram criadas as consultas que foram utilizadas para pesquisar nas fontes de dados pelos trabalhos. Essas consultas foram criadas com os seguintes termos, que deveriam ser encontrados nos trabalhos: *deep learning*, *context-aware recommender systems* e *embeddings*. As fontes de dados possuíam pequenas variações na sintaxe para criar a consulta, porém uma consulta genérica pode ser definida como a seguinte:

(“*deep learning*” AND “*context-aware recommender systems*”) OR (“*embeddings*” AND “*context-aware recommender systems*”)

Essa consulta foi utilizada nas principais fontes de dados que possuem trabalhos na área de ciência da computação. As fontes de dados escolhidas foram as seguintes: ACM Digital Library¹, IEEE Xplore Digital Library², ScienceDirect³ and Springer Link⁴.

O protocolo deve definir também os critérios de seleção/exclusão que são utilizados na fase de condução da revisão sistemática. Esses critérios determinam se os trabalhos são apropriados para serem revisados e, caso os trabalhos não estejam de acordo com os critérios, eles são excluídos da revisão. Os critérios de exclusão dessa revisão sistemática são os seguintes:

- Trabalhos que são secundários;
- Trabalhos que possuem acesso restrito ou não estão escritos em Inglês;
- Trabalhos que possuem menos de 4 páginas, pôsteres, apresentações e tutoriais;
- Trabalhos que tem como objetivo melhorar um sistema de recomendação sensível ao contexto, mas que não usam nenhum modelo para adquirir *embeddings*.

A fase de condução da revisão sistemática compreende a coleta dos trabalhos das fontes de dados, junto com a extração de dados dos trabalhos. A Figura 27 sumariza a fase

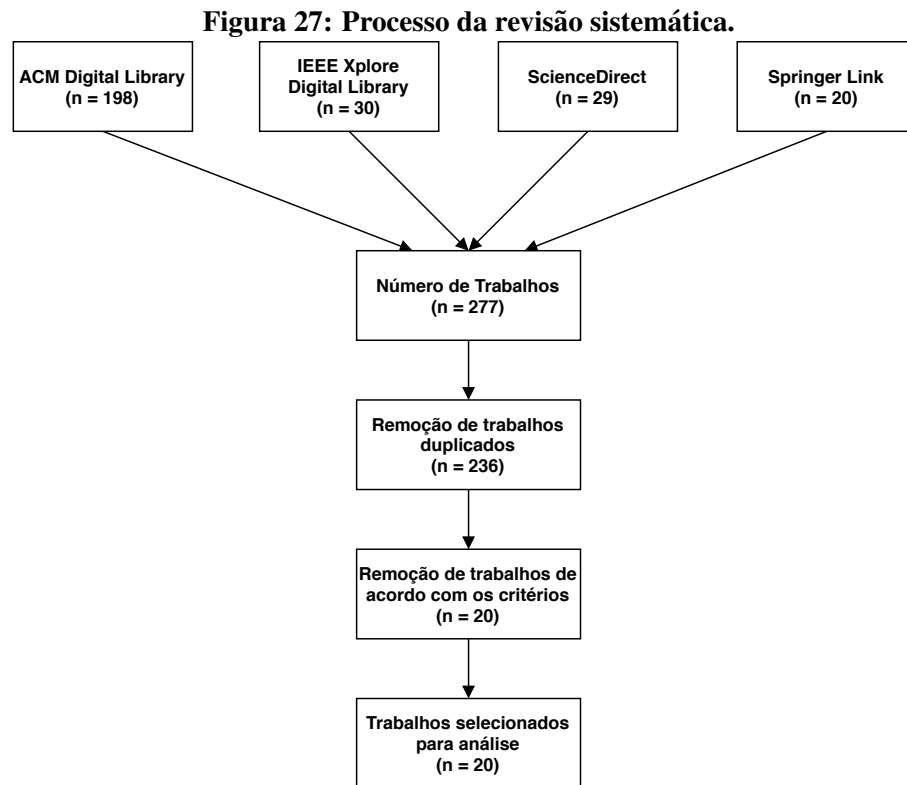
¹<https://dl.acm.org>

²<https://ieeexplore.ieee.org>

³<https://www.sciencedirect.com>

⁴<https://link.springer.com>

de condução, com os processos aplicados de remoção de trabalhos duplicado e remoção de trabalhos de acordo com os critérios de exclusão. Após a fase de condução, 20 trabalhos foram selecionados para análise de um total de 277 trabalhos obtidos de 4 fontes de dados diferentes. Os resultados obtidos com a fase de relatório foram reportados na Seção 3.



Fonte: A autoria própria.