UNIVERSIDADE FEDERAL DO CEARÁ

CAMPUS DE SOBRAL

CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

EVECLEISON ALBUQUERQUE DO NASCIMENTO

ESTUDO COMPARATIVO ENTRE SISTEMAS DE RECOMENDAÇÃO TRADICIONAIS E BASEADOS EM DEEP LEARNING: ANÁLISE DE DESEMPENHO E EFICIÊNCIA COMPUTACIONAL

SOBRAL

2025

EVECLEISON ALBUQUERQUE DO NASCIMENTO

ESTUDO COMPARATIVO ENTRE SISTEMAS DE RECOMENDAÇÃO TRADICIONAIS E BASEADOS EM DEEP LEARNING: ANÁLISE DE DESEMPENHO E EFICIÊNCIA COMPUTACIONAL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus de Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Iális Cavalcante de Paula Júnior.

SOBRAL

2025

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

N194e Nascimento, Evecleison Albuquerque do.
Estudo comparativo entre sistemas de recomendação tradicionais e baseados em deep learning : análise de desempenho e eficiência computacional / Evecleison Albuquerque do Nascimento. – 2025.
60 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Campus de Sobral, Curso de Engenharia da Computação, Sobral, 2025.
Orientação: Prof. Dr. Iális Cavalcante de Paula Júnior.

1. Sistemas de recomendação. 2. Modelos tradicionais. 3. Deep learning. 4. Desempenho preditivo. 5. Eficiência computacional. I. Título.

CDD 621.39

EVECLEISON ALBUQUERQUE DO NASCIMENTO

ESTUDO COMPARATIVO ENTRE SISTEMAS DE RECOMENDAÇÃO TRADICIONAIS E BASEADOS EM DEEP LEARNING: ANÁLISE DE DESEMPENHO E EFICIÊNCIA COMPUTACIONAL

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia de Computação do Campus de Sobral da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia de Computação.

Aprovada em: 04/08/2025.

BANCA EXAMINADORA

Prof. Dr. Iális Cavalcante de Paula Júnior (Orientador)
Universidade Federal do Ceará (UFC)

Profa. Dra. Jermana Lopes de Moraes
Universidade Federal do Ceará (UFC)

Prof. Me. David Nascimento Coelho
Universidade Federal do Ceará (UFC)

A Deus, detentor de toda a glória, por me guiar nessa jornada. À minha família, por todo amor e apoio. Dedico este trabalho aos meus pais, que sempre acreditaram em mim e estiveram presentes em cada passo da minha caminhada acadêmica, apoiando-me constantemente.

AGRADECIMENTOS

Agradeço a Deus pela vida que me concedeu, por me guiar nessa jornada e pela minha família. Aos meus pais, Carlos Alberto do Nascimento e Maria Eveline Silva de Albuquerque, sou grato de coração pelo apoio constante e pelos ensinamentos.

Ao meu orientador, Prof. Dr. Iális Cavalcante de Paula Júnior, pela excelente orientação, pelos ensinamentos, conselhos e pelas valiosas contribuições durante este trabalho.

Aos professores da banca, Profa. Dra. Jermana Lopes de Moraes e Prof. Me. David Nascimento Coelho, agradeço por doarem seu tempo e pelas contribuições a este trabalho. Agradeço aos professores da UFC pelo ensino e pelas orientações que contribuíram para a minha formação.

À coordenação do curso de Engenharia de Computação, muito obrigado pelo cuidado em garantir o bom funcionamento do curso ao longo desses anos, sempre orientando e esclarecendo dúvidas quando necessário.

Por fim, a todos os funcionários que fazem a UFC funcionar diariamente e tornam possível a existência desse ambiente de ensino, deixo aqui minha sincera gratidão.

"O que sabemos é uma gota; o que ignoramos é um oceano." (Atribuída a Isaac Newton)

RESUMO

Sistemas de recomendação são ferramentas digitais usadas em plataformas online para personalizar a experiência do usuário e facilitar a descoberta de itens relevantes. Este trabalho realiza um estudo comparativo entre abordagens tradicionais e baseadas em *deep learning* para recomendação de filmes, utilizando o conjunto de dados *The Movies Dataset*. Foram avaliados dois modelos tradicionais – FunkSVD e TF-IDF – e dois modelos baseados em *deep learning* – AutoRec e DistilRoBERTa com *fine-tuning* via *triplet loss*. A avaliação considerou métricas preditivas (Precision@10, Recall@10 e NDCG@10) e de eficiência computacional (tempo de treinamento, tempo de inferência e pico de memória RAM). O FunkSVD apresentou o melhor desempenho em Precision@10 (0,7365) e Recall@10 (0,4871), superando tanto o AutoRec quanto o DistilRoBERTa nesses aspectos. Embora o DistilRoBERTa tenha atingido o maior NDCG@10 (0,8336), também apresentou o maior tempo de treinamento (22,17 minutos). Já o TF-IDF teve o menor consumo de memória, mas apresentou os piores resultados preditivos. Os resultados indicam que, apesar do potencial dos modelos baseados em *deep learning*, abordagens tradicionais como o FunkSVD podem oferecer desempenho competitivo com menor custo computacional. Em determinados contextos, os métodos mais simples podem inclusive superar soluções mais complexas, especialmente quando há escassez de dados ou restrições de recursos computacionais.

Palavras-chave: sistemas de recomendação; modelos tradicionais; deep learning; desempenho preditivo; eficiência computacional.

ABSTRACT

Recommender systems are digital tools used on online platforms to personalize user experiences and facilitate the discovery of relevant items. This work presents a comparative study between traditional and deep learning-based approaches for movie recommendation, using The Movies Dataset. Two traditional models – FunkSVD and TF-IDF – and two deep learning-based models – AutoRec and DistilRoBERTa with triplet loss fine-tuning – were evaluated. The evaluation considered predictive metrics (Precision@10, Recall@10, and NDCG@10) and computational efficiency (training time, inference time, and peak RAM usage). FunkSVD achieved the best performance in Precision@10 (0.7365) and Recall@10 (0.4871), outperforming both AutoRec and DistilRoBERTa in these aspects. Although DistilRoBERTa reached the highest NDCG@10 (0.8336), it also had the longest training time (22.17 minutes). TF-IDF showed the lowest memory consumption but had the worst predictive results. The findings indicate that, despite the potential of deep learning-based models, traditional approaches like FunkSVD can offer competitive performance with lower computational cost. In certain contexts, simpler methods may even outperform more complex solutions, especially when data is scarce or computational resources are limited.

Keywords: recommender systems; traditional models; deep learning; predictive performance; computational efficiency.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AE	*Autoenconder*
API	*Application Programming Interface*
BERT	*Bidirectional Encoder Representations from Transformers*
DCG	*Discounted Cumulative Gain*
DL	*Deep Learning*
FN	Falso Negativo
FP	Falso Positivo
GPU	*Graphics Processing Unit*
IA	Inteligência Artificial
IDCG	*Ideal Discounted Cumulative Gain*
ML	*Machine Learning*
NDCG	*Normalized Discounted Cumulative Gain*
NLP	*Natural Language Processing*
NSP	*Next Sentence Prediction*
RAM	*Random Access Memory*
RoBERTa	*Robustly Optimized BERT Approach*
SVD	*Singular Value Decomposition*
TF-IDF	*Term Frequency – Inverse Document Frequency*
TMDB	*The Movie Database*
TP	*True Positive*

SUMÁRIO

1 INTRODUÇÃO

A internet se desenvolveu em um ambiente para serviços *online* de considerável proporção, modificando profundamente a forma da comunicação da sociedade, pois possibilitou que o ser humano tivesse acesso a diversas informações e opções de consumo. Diante disso, a variedade de itens disponibilizados *online* requer um sistema que ajude a definir as preferências do indivíduo. Nesse contexto, os sistemas de recomendação são ferramentas de filtragem de informações que proporcionam serviços individualizados e uma experiência adaptada para os usuários. Eles aliviam a sobrecarga de informações, diminuindo o esforço dos usuários em encontrar itens, e melhoram as prestações de serviços, gerando uma importante fonte de receita às plataformas digitais. Os sistemas de recomendação são utilizados na indústria e no cotidiano, em várias áreas, como sites de compras *online*, serviços de *streaming*, lojas de aplicativos móveis e redes sociais (ZHANG *et al.*, 2023).

Ao longo dos anos, os sistemas de recomendação tradicionais foram usados para oferecer aos usuários produtos e serviços individualizados, mas a maioria deles tem dificuldade em lidar com o enorme volume, a complexidade e a dinâmica das informações. Diante disso, vários estudos foram realizados para melhorar os sistemas de recomendação utilizando *Deep Learning* (DL) (ZHANG *et al.*, 2020).

O DL vem remodelando as arquiteturas de recomendação, gerando mais possibilidades para aprimorar o desempenho desses sistemas. Avanços recentes em soluções baseadas nessa abordagem alcançaram destaque considerável, ultrapassando os obstáculos dos modelos convencionais e obtendo alta qualidade na recomendação. Essa técnica consegue capturar as relações não lineares e não triviais entre usuário e item, proporcionando a codificação de abstrações mais complexas, como a identificação de padrões presentes nos próprios dados, provenientes de diversas fontes amplamente disponíveis, tais como informações contextuais, textuais e visuais (ZHANG *et al.*, 2017).

Entretanto, os modelos baseados em DL introduzem uma considerável complexidade computacional, exigindo muitos ajustes de hiperparâmetros e, geralmente, carecem de transparência, ou seja, não fornecem meios de compreender como as decisões foram feitas, podendo dificultar sua implementação em sistemas de tempo real de grande escala (SCRIVANO, 2025).

Para compreender o desenvolvimento das pesquisas, um estudo feito por Dacrema et al. (2021) comparou os resultados obtidos em sistemas de recomendação baseados em filtragem colaborativa usando DL com um conjunto de *baselines* (linhas de base), que são modelos simples

que servem de referência para avaliar se novos métodos realmente representam algum avanço. A partir da análise dos trabalhos avaliados por Dacrema et al., verificou-se que nenhum dos métodos baseados em DL se mostrou consistentemente melhor do que as abordagens tradicionais já existentes, como a fatoração de matrizes e modelos lineares. O estudo também identificou problemas na prática de pesquisa atual, como a escolha inadequada de *baselines*, falta de ajustes nos hiperparâmetros e falta de reprodutibilidade (DACREMA *et al.*, 2021).

Um dos fatores que contribuem para essa falta de reprodutibilidade é a baixa disponibilidade e qualidade do código-fonte associado aos trabalhos científicos. Embora se perceba uma tendência crescente entre os pesquisadores em publicar o código-fonte de seus trabalhos, essa ainda não é uma prática comum nas publicações científicas. Além disso, mesmo quando o código é disponibilizado, ele pode estar incompleto, não incluindo, por exemplo, o código de pré-processamento dos dados, o ajuste de parâmetros ou os procedimentos exatos de avaliação (DACREMA *et al.*, 2019).

Portanto, é fundamental que os estudos demonstrem o código-fonte, detalhem o pré-processamento dos dados e forneçam informações claras sobre os detalhes de implementação. Além disso, diversos trabalhos, como os de Coutinho e Marcacini (2022), Matos (2021) e Jin et al. (2024), analisaram apenas o desempenho preditivo, sem abordar a eficiência computacional, evidenciando uma carência na literatura.

Por isso, é importante a realização de um estudo comparativo que avalie conjuntamente o desempenho preditivo e a eficiência computacional, detalhando o processo de implementação para garantir solidez e reprodutibilidade dos resultados. Essa análise possibilita compreender os pontos fortes e limitações de diferentes abordagens de recomendação, auxiliando na escolha da abordagem mais adequada ao contexto computacional.

Diante disso, este trabalho busca responder ao seguinte problema de pesquisa: como os modelos tradicionais de recomendação (FunkSVD e TF-IDF) se comparam aos modelos baseados em DL (AutoRec e DistilRoBERTa) em termos de desempenho preditivo e eficiência computacional?

Então, o objetivo geral desta pesquisa é comparar modelos tradicionais de recomendação e modelos baseados em DL, identificando os contextos em que cada abordagem apresenta melhores resultados.

Com esse propósito, definiram-se alguns objetivos específicos: explicar os conceitos fundamentais dos sistemas de recomendação, incluindo suas diferentes categorias; mostrar as

particularidades, pontos fortes e pontos fracos dos modelos utilizados e avaliar tanto a precisão das previsões quanto o uso dos recursos computacionais por cada algoritmo.

A metodologia é de natureza explicativa, experimental, bibliográfica, com abordagem quantitativa. Foram implementados modelos representativos dos sistemas de recomendação tradicionais e baseados em DL, utilizando a base de dados *The Movies Dataset*, publicamente acessível na plataforma Kaggle. A análise dos resultados foi realizada com base em métricas de desempenho e eficiência computacional, visando garantir a objetividade, reprodutibilidade e solidez dos resultados.

Este trabalho está organizado em cinco capítulos. O Capítulo 1 é esta introdução. No Capítulo 2, são explorados os conceitos centrais que sustentam teoricamente o estudo. O Capítulo 3 apresenta a metodologia. O Capítulo 4 consiste nos resultados obtidos. Por fim, o Capítulo 5 demonstra as conclusões do estudo e sugere caminhos para pesquisas futuras.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda conceitos importantes e o desenvolvimento teórico dos sistemas de recomendação. Cada seção explica pontos relevantes dos sistemas de recomendação que serão utilizados neste trabalho.

2.1 Sistemas de recomendação

Os sistemas de recomendação são utilizados em plataformas de comércio eletrônico, serviços de *streaming* e em outras aplicações para atender aos diversos interesses dos usuários. Eles realizam a predição de itens ou produtos que provavelmente serão do interesse dos usuários com base em seu histórico, preferências e comportamento (JIN *et al.*, 2024).

Diante disso, a seleção de uma abordagem de recomendação considera as informações disponíveis sobre os usuários e os itens. No caso de filmes, por exemplo, há diversos dados descritivos, como título, diretores e elenco, que são informações que compõem o conteúdo do item. Além disso, sistemas de recomendação podem usar avaliações fornecidas pelos usuários, que podem ser explícitas ou implícitas (MATOS, 2021).

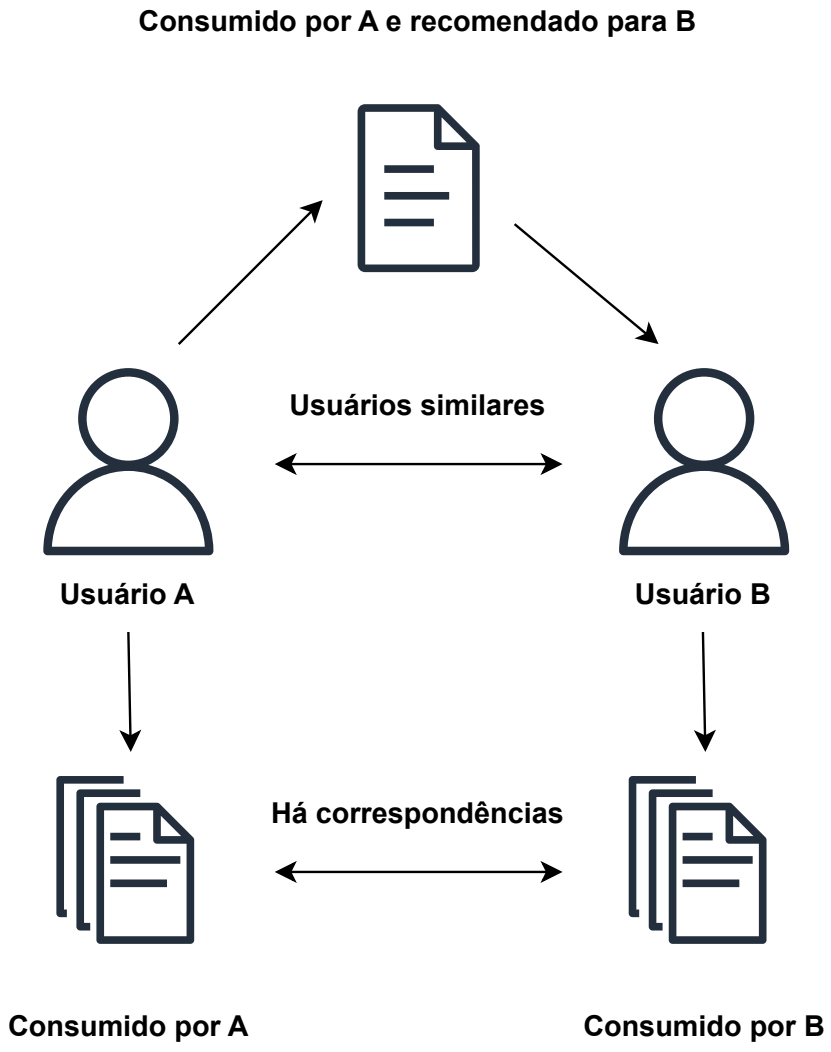
O *feedback* explícito consiste na manifestação direta das preferências dos usuários, por exemplo, o modelo de avaliação por estrelas feito por alguns serviços e o YouTube fornecendo os botões de *like* e *dislike*. Contudo, nem sempre é viável obter o *feedback* explícito, pois muitos relutam em avaliar as recomendações. Já o *feedback* implícito indica indiretamente a opinião do usuário por meio da sua atividade na plataforma, por isso o implícito está frequentemente disponível. Há várias formas de *feedback* implícito, como movimentos de *mouse*, histórico de compras e de navegação. Nesse contexto, se um usuário leu muitos livros de um autor, pode-se deduzir que ele gosta desse autor, mas o fato de ele ter lido um livro não indica necessariamente uma visão positiva dessa obra. Assim, no *feedback* implícito, ocorre apenas uma dedução das preferências do usuário (ZHANG *et al.*, 2023).

Portanto, a forma como essas informações são obtidas e tratadas tem impacto direto na escolha da abordagem de recomendação. Nesse contexto, os sistemas podem ser classificados em três categorias: filtragem colaborativa, filtragem baseada em conteúdo e filtragem híbrida, que serão detalhadas a seguir.

2.1.1 Filtragem colaborativa

A filtragem colaborativa consiste na ideia de que, se dois usuários tiverem considerações similares sobre os mesmos itens, logo tenderão a ter a mesma opinião sobre outros itens. Para determinar isso, há uma análise e uma comparação entre as preferências dos usuários, determinando assim o padrão de semelhança entre eles para recomendar os itens que um usuário gostou para outro com interesses parecidos (MATOS, 2021). Isso é representado pela Figura 1.

Figura 1 – Filtragem colaborativa



Fonte: Elaborado pelo autor.

Ademais, a filtragem colaborativa precisa de informações que definam as características de itens e usuários, por isso dados pré-existentes são utilizados. Eles podem ser adquiridos de várias maneiras, por exemplo, pelo cadastramento de dados por empresas que investem em análise de dados e na determinação das características dos itens. Além disso, podem ser

fornecidos explicitamente pelos usuários, como, por exemplo, na avaliação de filmes. Outra maneira de obtenção é o uso de dados de consumo, que é um *feedback* implícito, como no caso da contagem de músicas por usuário (MASSON, 2016).

Há vários modelos de filtragem colaborativa. A maioria deles se concentra em utilizar as correlações entre itens ou entre usuários para realizar a previsão, sendo que alguns usam ambos. Dessa forma, eles podem ser de dois tipos: métodos baseados em memória e métodos baseados em modelo. Os métodos baseados em memória também são conhecidos como filtragem colaborativa baseada em vizinhança. Eles estavam entre os primeiros algoritmos de filtragem trabalhados colaborativamente e se caracterizavam pela previsão de itens baseada na combinação de itens com os vizinhos mais próximos. Essas vizinhanças podem ser definidas em um desses dois caminhos: baseada no usuário e baseada em itens. Já no método baseado em modelos, o *Machine Learning* (ML) e a mineração de dados são usados no contexto de modelos preditivos. Os parâmetros do modelo são ajustados por meio de uma estrutura de otimização, e alguns exemplos de tais modelos incluem árvores de decisão e modelos de fatores latentes (AGGARWAL, 2016).

Os modelos de filtragem colaborativa têm como desvantagem o *cold start*, que é um problema que acontece quando há uma adição de novos itens e usuários a um modelo. Dessa forma, ocorre a dificuldade de classificação de um novo usuário, pois, no momento da inserção no sistema, ele é constituído de um número insuficiente de informações associadas aos outros usuários. Há também a escalabilidade do sistema, que ocorre em plataformas com milhares de itens e usuários na base de dados, exigindo uma capacidade computacional maior pela empresa (MATOS, 2021).

Além disso, a filtragem colaborativa apresenta dificuldades quando aplicada a conjuntos de dados esparsos. Em matrizes com poucas interações entre usuários e itens, a qualidade das recomendações tende a ser comprometida, já que esse método depende de um volume significativo de avaliações para identificar padrões de similaridade e preferências de forma eficaz (OLIVEIRA, 2012).

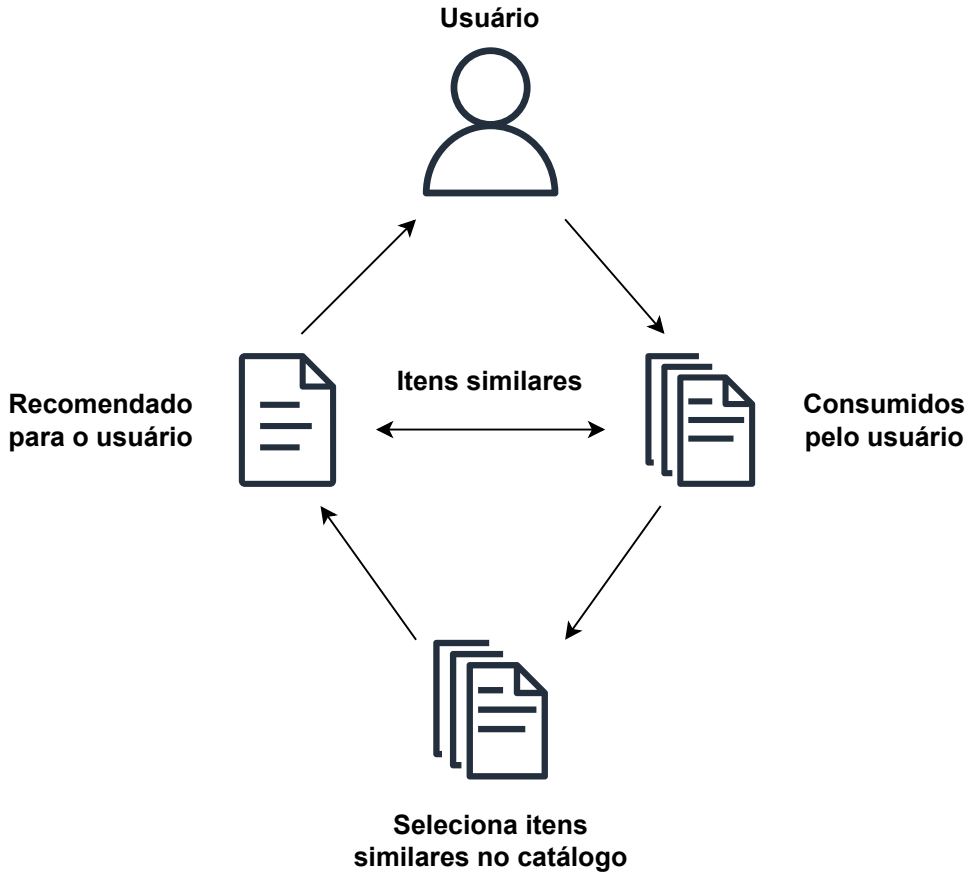
2.1.2 Filtragem baseada em conteúdo

A filtragem baseada em conteúdo faz recomendações com base nas características dos próprios itens. Para isso, analisa-se o comportamento de consumo e as avaliações dos usuários em conjunto com as informações descritivas dos itens, de modo a sugerir aqueles que

tenham semelhanças com os que o usuário já consumiu (AGGARWAL, 2016).

No caso de filmes, há, como exemplo de informações, o gênero, elenco, diretor e a sinopse. O sistema de recomendações equipara esses dados com o histórico de preferências do usuário para apresentar filmes semelhantes ao que ele já consumiu. Assim, se um indivíduo viu filmes de ação com um determinado autor, a plataforma vai sugerir outros filmes com a mesma temática ou com o mesmo autor (MINGRONI, 2024). A Figura 2 demonstra esse processo de recomendação.

Figura 2 – Filtragem baseada em conteúdo



Fonte: Elaborado pelo autor.

O primeiro passo para um sistema de recomendação baseado em conteúdo é o de pré-processamento e extração de *features*, que coleta, em uma base de dados, uma enorme quantidade de informações não estruturadas sobre itens diversos que são determinantes para a predição das pontuações. Na segunda etapa, é construído o perfil do usuário baseado no conteúdo, verificando o histórico de avaliações do usuário, relacionando conceitualmente as opiniões do usuário às características de cada item. Dessa forma, o modelo desenvolvido no passo anterior é usado para realizar recomendações sobre itens, configurando na etapa de filtragem e recomendação.

Há, como exemplos desse tipo de filtragem, os métodos que usam árvores de decisão e *word embeddings* (MATOS, 2021).

A filtragem baseada em conteúdo tem a vantagem de não precisar de informações de outros usuários. Além disso, recomenda itens novos sem problemas de *cold start* ou de esparsidade, que ocorrem quando um item inédito é colocado no conjunto e não possui avaliações prévias. Ademais, apresenta explicações sobre os itens recomendados e a razão de serem mencionados. Uma das desvantagens desse método é que o conteúdo deve ser codificado por atributos de fácil compreensão, porque ele é analisado automaticamente para determinar as categorias, então uma descrição incompleta ou imprecisa do item dificulta essa categorização. Assim, há uma ineficiência se o item não for muito informativo (OLIVEIRA, 2012).

Outro problema é a superespecialização, pois a filtragem baseada em conteúdo não possui um método inerente para encontrar algo inesperado e tende a recomendar itens com um grau limitado de novidade. Se, por exemplo, um usuário avaliou somente filmes dirigidos por Stanley Kubrick, serão recomendados apenas esse tipo de filme. Portanto, uma abordagem em conteúdo, mesmo que altamente precisa, dificilmente encontraria algo novo, limitando o escopo de aplicações para as quais ela poderia ser adequada (RICCI *et al.*, 2011).

2.1.3 Filtragem híbrida

A filtragem híbrida consiste na combinação de duas ou mais técnicas de recomendação para alcançar um desempenho melhor. Esse método tem como objetivo extinguir as desvantagens presentes nas técnicas quando usadas individualmente. Um sistema híbrido comum combina a filtragem colaborativa com a filtragem baseada em conteúdo. Essa junção pode focar nos itens recomendados por ambos e colocá-los em uma lista. Outra opção disponível é implementar a filtragem colaborativa e depois aperfeiçoar a lista de itens recomendados por intermédio da filtragem baseada em conteúdo (COUTINHO; MARCACINI, 2022).

Nesse contexto, a filtragem híbrida de cascata é caracterizada pela aplicação de um processo de refinamento iterativo no desenvolvimento de uma ordem de preferência entre diferentes itens. Logo, as recomendações de uma técnica são refinadas por outra técnica de recomendação. Assim, a primeira técnica cria uma lista de recomendações que será aprimorada pela próxima técnica de recomendação (ISINKAYE *et al.*, 2015).

A filtragem híbrida busca aproveitar os benefícios da filtragem colaborativa e da baseada em conteúdo, compensando as limitações de cada uma (REATEGUI; CAZELLA, 2005).

Essas vantagens são mostradas na Figura 3.

Figura 3 – Filtragem híbrida

Fonte: Reategui e Cazella, 2005.

Aplicando a filtragem híbrida no contexto de recomendações de filmes, viabilizam-se recomendações mais diversificadas e precisas. Nesse contexto, a filtragem baseada em conteúdo foca nas características dos filmes, enquanto a filtragem colaborativa verifica os interesses coletivos sobre os filmes, construindo uma experiência personalizada. Esse método resolve problemas como o *cold start* e a superespecialização. Assim, esse sistema integra a avaliação dos usuários, o histórico de visualizações e dados específicos do filme, gerando uma recomendação equilibrada que consiste em conteúdos populares e novos conteúdos, de acordo com o perfil do usuário. Abordagens adicionais podem ser utilizadas para adicionar outros dados, como o histórico de compras e interações em dispositivos conectados, a fim de aprimorar as recomendações, incluindo mais uma camada de contextualização. No entanto, a filtragem híbrida apresenta problemas, como a demanda por um balanceamento preciso para evitar recomendações irrelevantes e a complexidade computacional (MINGRONI, 2024).

Para melhorar a identificação e comparação de similaridades entre itens, utilizam-se *embeddings*, que representam os dados em espaços vetoriais compactos.

2.2 Embeddings

Determinar a similaridade entre dois itens pode ser difícil, especialmente dependendo do tipo de dado utilizado para caracterizá-los. Isso torna a escolha dos atributos mais adequados uma tarefa complexa, podendo levar a resultados incorretos na identificação de semelhanças.

Diante disso, os *embeddings* simplificam e otimizam esse processo. O *embedding* é um vetor de *features* do objeto, ou seja, é uma indicação de *features* em um espaço vetorial compacto e de tamanho fixo, contendo características semânticas sobre o objeto (SANTANA, 2019).

Assim, os *embeddings* armazenam um número grande de informações úteis em menos espaço, porque são otimizados na sua formação, tendo a dimensionalidade do espaço reduzida. O vetor pode possuir qualquer tamanho, independentemente do número de *features* e tipo de dados usados, propiciando operações como concatenação, comparação e manipulação. Além disso, os vetores são densos, isto é, eles são constituídos de valores reais que representam uma posição em um espaço vetorial de N dimensões, permitindo o conceito de distância e semelhança entre eles (SANTANA, 2019).

Qualquer tipo de dado, como imagem, áudio e texto, pode ter um *embedding* criado para representá-lo vetorialmente. Essa funcionalidade é interessante para um sistema de recomendações, pois, seja qual for o tipo de item a ser recomendado, o algoritmo não se modifica (SANTANA, 2019).

A seguir, serão apresentados dois tipos fundamentais de *embeddings* utilizados no processamento de texto: os *word embeddings* e os *sentence embeddings*, que permitem capturar informações semânticas.

2.2.1 Word embeddings

Os *embeddings* mais populares são os *word embeddings*, que codificam um texto em um vetor de características. Em tarefas de *Natural Language Processing* (NLP), o uso de *word embeddings* é essencial para estruturar a linguagem da forma mais natural possível, contendo o contexto e a semântica embutidos na palavra. A principal característica dos *word embeddings* é a capacidade de modelar a semelhança de palavras semanticamente, onde a distância entre os vetores indica um sentido em um contexto específico. Por exemplo, a distância vetorial entre as palavras *queen* e *king* é igual à distância vetorial entre as palavras *woman* e *man* (SANTANA, 2019).

Portanto, *word embeddings* são técnicas de modelagem da linguagem em que palavras de um texto são representadas por vetores de valor real em um espaço vetorial. Isso possibilita que palavras com semântica similar sejam facilmente identificadas e comparadas numericamente e vetorialmente. Implementando esse método para a recomendação de filmes, pode-se adquirir dados textuais relacionados ao que o usuário gosta, por exemplo, informações

de uma sinopse, e construir um modelo que infere seus interesses. Para definir se um filme não assistido pode ser da preferência de um usuário, basta analisar as informações textuais dele e compará-las ao modelo construído. Caso esses dados sejam consideravelmente similares, o filme é recomendável (MATOS, 2021).

2.2.2 Sentence embeddings

Os *sentence embeddings* representam sentenças de texto em um espaço vetorial de n dimensões, de maneira que palavras semanticamente semelhantes ou relacionadas estejam próximas umas das outras durante o treinamento. Eles permitem distintas representações para uma palavra conforme o contexto em que aparece (JUNIOR *et al.*, 2023).

Isso possibilita que o modelo relacione palavras com seu contexto e também com sentenças inteiras, tornando sua interpretação menos literal e mais semelhante à interpretação humana. Nesse contexto, quando o ser humano lê um texto, ele não interpreta apenas palavra por palavra isoladamente, mas também considera o contexto geral para dar sentido ao que está sendo lido (SOUSA, 2022).

Assim, considerando que as sentenças estão representadas por vetores, torna-se necessário um método para mensurar o grau de similaridade entre eles, sendo a similaridade do cosseno uma das métricas mais utilizadas para esse fim.

2.3 Métrica de similaridade: a similaridade do cosseno

A similaridade do cosseno é uma métrica de similaridade entre dois vetores no espaço vetorial, que calcula a medida do cosseno do ângulo compreendido entre eles, conforme mostrado na equação (SOUSA, 2022):

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|}. \tag{2.1}$$

O termo $A \cdot B$ indica o produto escalar dos vetores A e B, obtido pela soma dos produtos dos valores correspondentes de cada vetor. Já $\|A\|$ e $\|B\|$ representam os comprimentos desses vetores, calculados pela raiz quadrada da soma dos quadrados de seus elementos (MINGRONI, 2024).

O valor obtido com esse cálculo varia entre -1 e 1, indicando o grau de similaridade entre os vetores, sendo que quanto mais próximo de 1, maior a semelhança entre eles. O principal

benefício dessa medida é a possibilidade de usar vetores de n dimensões e, como o cálculo é feito a partir do ângulo, dois vetores podem ser considerados similares mesmo que ambos sejam de grandezas distintas ou estejam em posições diferentes no espaço vetorial (SOUSA, 2022).

2.4 Modelos tradicionais de recomendação

Apesar do avanço de técnicas mais recentes, modelos tradicionais de recomendação ainda são usados devido à sua simplicidade de implementação, exigirem menor poder computacional e alguns conseguirem ser usados em contextos computacionais limitados ou com poucos dados. Esta seção apresenta os modelos tradicionais de recomendação implementados neste trabalho, descrevendo suas características, vantagens e limitações.

2.4.1 Fatoração de matriz

A fatoração de matriz representa usuários e itens por meio de vetores de fatores inferidos a partir de padrões de classificação de itens. Nesse contexto, a correspondência acentuada entre os fatores gera uma recomendação. Esse método oferece flexibilidade para modelar variadas situações da vida real. Os sistemas de recomendação precisam de vários tipos de entrada, que podem ser inseridos em uma matriz, com uma dimensão indicando os usuários e a outra dimensão indicando os itens. Assim, os dados mais convenientes são os de *feedback* explícito, que incluem informações dos usuários. A Netflix, por exemplo, coleta avaliações com estrelas para filmes. O *feedback* explícito resulta em uma matriz esparsa, pois é mais provável que um usuário avalie apenas uma pequena porcentagem dos itens disponíveis. Quando o *feedback* explícito está indisponível, os sistemas de recomendação podem deduzir as preferências do usuário utilizando *feedback* implícito, que indica indiretamente a opinião do usuário, analisando o seu comportamento. O *feedback* implícito, na maioria das vezes, denota a ausência ou a presença de um evento, por isso normalmente compreende uma matriz densamente preenchida (KOREN *et al.*, 2009).

Modelos de fatoração matricial mapeiam usuários e itens através de um conjunto de fatores latentes de dimensionalidade f, de modo que as interações usuário-item são modeladas como produtos internos de duas matrizes. Assim, cada item i está associado a um vetor $q_i \in \mathbb{R}^f$, e cada usuário u está associado a um vetor $p_u \in \mathbb{R}^f$. Para um determinado item i, os elementos de q_i medem até que ponto o item possui esses fatores, como o gênero e autor de uma obra

literária. Para um dado usuário, os elementos de p_u medem a extensão de interesse que o usuário possui em relação aos fatores correspondentes dos itens considerados (KOREN *et al.*, 2009).

Diante disso, o produto escalar resultante $q_i^\top p_u$ obtém a interação entre o usuário u e o item i, indicando o nível de interesse geral do usuário em relação às características do item. Dessa forma, a avaliação estimada que o usuário atribui ao item é determinada por (KOREN *et al.*, 2009):

$$\hat{r}_{ui} = q_i^\top p_u. \tag{2.2}$$

O grande desafio computacional presente nesse cenário é o mapeamento de cada item e usuário para fatorar vetores q_i e $p_u \in \mathbb{R}^f$. Após o sistema de recomendação terminar esse mapeamento, ele pode predizer a avaliação que um usuário dará a um item usando a Equação 2.2. Esse modelo está relacionado à *Singular Value Decomposition* (SVD), uma técnica que identifica fatores semânticos latentes na obtenção da informação. O uso de SVD na filtragem colaborativa requer a fatoração da matriz de classificação. Isso pode gerar problemas devido à ausência considerável de valores, ocasionada pela dispersão na matriz de classificação (KOREN *et al.*, 2009).

Além disso, considerar apenas os aspectos das poucas entradas fornecidas pode resultar em *overfitting*. Há a abordagem da imputação para o preenchimento da matriz; entretanto, a imputação pode ser problemática, porque a imputação imprecisa pode distorcer os dados consideravelmente. Para resolver isso, pode-se modelar apenas as classificações observadas por intermédio de um modelo regularizado. Assim, o treinamento do modelo de fatoração de matriz pode ser baseado na minimização do erro quadrático médio entre as notas de classificação previstas e as notas de classificação reais. Portanto, a função objetivo é definida de acordo com a equação (KOREN *et al.*, 2009):

$$\min_{q,p} \sum_{(u,i)\in\kappa} \left(r_{ui} - q_i^\top p_u\right)^2 + \lambda \left(\|q_i\|^2 + \|p_u\|^2\right). \tag{2.3}$$

Na Equação 2.3, k é o conjunto dos pares (u,i) , pelos quais $\hat{r}_{ui}$ é conhecido, ou seja, é o conjunto de treinamento. O sistema aprende se modificando com as classificações observadas, contudo, o objetivo é generalizar essas classificações anteriores para depois prever as futuras classificações. O termo λ denota o grau de regularização (KOREN *et al.*, 2009).

2.4.2 FunkSVD

Para resolver o problema da baixa eficiência computacional do SVD e a necessidade de preencher valores ausentes, foi proposto o FunkSVD. Embora esse algoritmo seja simples, ele é amplamente utilizado e elevou os algoritmos de recomendação baseados em modelo a um novo patamar (XIAOCHEN; QICHENG, 2022).

Essa versão do modelo de fatoração de matrizes foi proposta por Simon Funk em um famoso *blog post*, no qual ele apresentou a ideia de fatorar a matriz de interações entre usuários e itens. O método ganhou popularidade com a competição Netflix Prize de 2006 (ZHANG *et al.*, 2023).

Seja $\mathbf{R} \in \mathbb{R}^{m \times n}$ a matriz de interação usuário-item, que representa as avaliações explícitas entre m usuários e n itens. Essa matriz pode ser decomposta em uma matriz latente de usuários $\mathbf{P} \in \mathbb{R}^{m \times k}$ e em uma matriz latente de itens $\mathbf{Q} \in \mathbb{R}^{n \times k}$, onde $k \ll m, n$ é o número de fatores latentes. Cada linha $\mathbf{p}_u$ da matriz $\mathbf{P}$ representa o vetor de características latentes do usuário u, enquanto cada linha $\mathbf{q}_i$ da matriz $\mathbf{Q}$ representa o vetor de características latentes do item i. Os elementos de $\mathbf{q}_i$ descrevem em que grau o item possui certas propriedades, como por exemplo gênero ou idioma, no caso de um filme. Da mesma forma, os elementos de $\mathbf{p}_u$ indicam o nível de interesse do usuário u nessas propriedades. Esses fatores latentes podem refletir atributos explícitos ou até dimensões não interpretáveis diretamente. A predição das avaliações pode então ser obtida através do produto matricial:

$$\hat{R} = PQ^{\top}, \tag{2.4}$$

onde $\hat{R} \in \mathbb{R}^{m \times n}$ é a matriz estimada de avaliações, com a mesma dimensão da matriz original $\mathbf{R}$ (ZHANG *et al.*, 2023).

Nesse contexto, Simon Funk tornou conhecida, em seu *blog post*, uma otimização da Equação 2.3 baseada em descida do gradiente estocástico. Esse algoritmo percorre todas as avaliações no conjunto de treinamento. Para cada amostra no conjunto de treinamento, o sistema prediz $\hat{r}_{ui}$, e calcula o erro de predição correspondente (KOREN *et al.*, 2009):

$$e_{ui} = r_{ui} - q_i^{\top} p_u. \tag{2.5}$$

Em seguida, ele atualiza os parâmetros proporcionalmente à taxa de aprendizado γ, na direção oposta ao gradiente, resultando em (KOREN *et al.*, 2009):

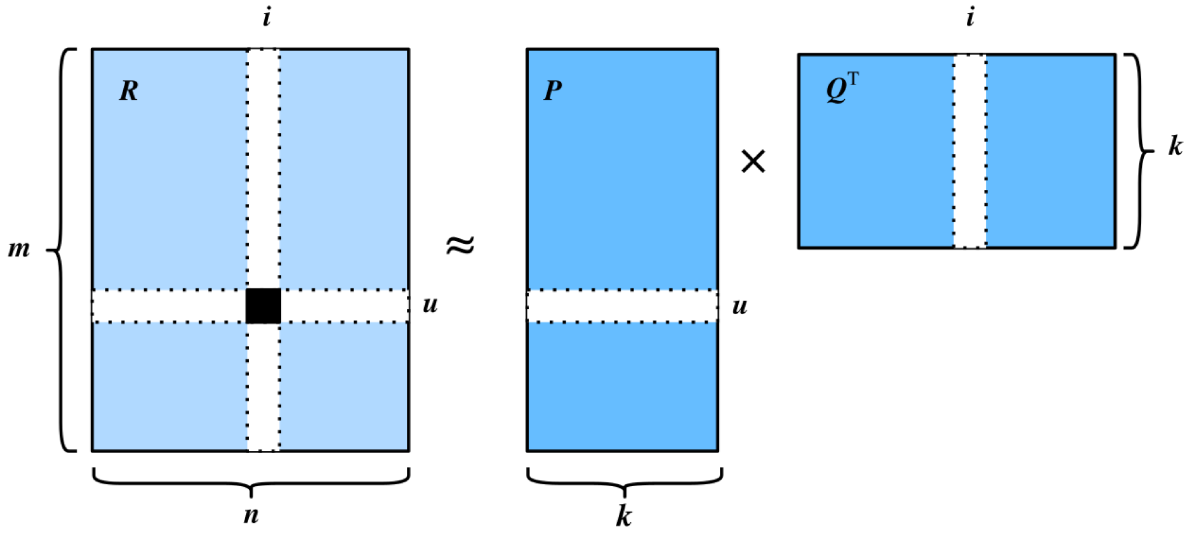
$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i), \tag{2.6}$$

$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u). \tag{2.7}$$

Essa abordagem possui uma facilidade de implementação e um tempo de execução relativamente rápido (KOREN *et al.*, 2009).

Embora o algoritmo FunkSVD resolva o problema do algoritmo SVD em preencher os valores ausentes, à medida que o volume de dados aumenta e a esparsidade dos dados aumenta, a precisão do algoritmo diminuirá. Além disso, o uso do método de descida de gradiente para atualizar os valores dos parâmetros pode causar oscilações iterativas, reduzindo assim a precisão (XIAOCHEN; QICHENG, 2022). A figura a seguir ilustra de forma intuitiva o modelo FunkSVD.

Figura 4 – Ilustração intuitiva do modelo FunkSVD



Fonte: adaptado de ZHANG et al. (2023, p. 792).

Na Figura 4, **R** é a matriz de interação com m usuários e n itens representando avaliações explícitas. Além disso, **P** é uma matriz latente de usuário e **Q** é uma matriz latente de item (ZHANG *et al.*, 2023).

2.4.3 TF-IDF

O *Term Frequency – Inverse Document Frequency* (TF-IDF) e a similaridade do cosseno são técnicas consolidadas na construção de sistemas de recomendação. O TF-IDF especifica a relevância de um termo específico dentro de um conjunto de textos, representada na forma de um vetor numérico (SANGEETHA *et al.*, 2025).

A frequência de termos (TF, do inglês *term frequency*) indica a frequência relativa de um termo em relação ao total de palavras no documento. Isso consiste em calcular a razão entre o número de ocorrências do termo e o total de palavras no documento, conforme a equação (CHINY *et al.*, 2022):

$$\mathrm{tf}_i = \frac{n_i}{\sum_k n_k}. \tag{2.8}$$

Na Equação 2.8, tf_i representa a frequência do termo i, n_i é o número de ocorrências desse termo no documento e $\sum_k n_k$ corresponde ao total de termos presentes no documento.

Já a frequência inversa de documento (IDF, do inglês *inverse document frequency*) é uma medida da importância de um termo no *corpus* como um todo, onde *corpus* é um conjunto organizado de textos. A IDF corresponde à divisão entre o número total de documentos do corpus e o número de documentos em que o termo está presente. Além disso, consiste em calcular o logaritmo dessa razão, conforme a equação (CHINY *et al.*, 2022):

$$\mathrm{idf}_i = \log\left(\frac{|D|}{|\{d_j : t_i \in d_j\}|}\right). \tag{2.9}$$

Na Equação 2.9, idf_i representa a frequência inversa de documento do termo i, $|D|$ é o número total de documentos no corpus e $|\{d_j : t_i \in d_j\}|$ corresponde à quantidade de documentos d_j que contêm o termo t_i. O operador log é utilizado para suavizar o valor, evitando que termos muito raros recebam peso excessivamente alto.

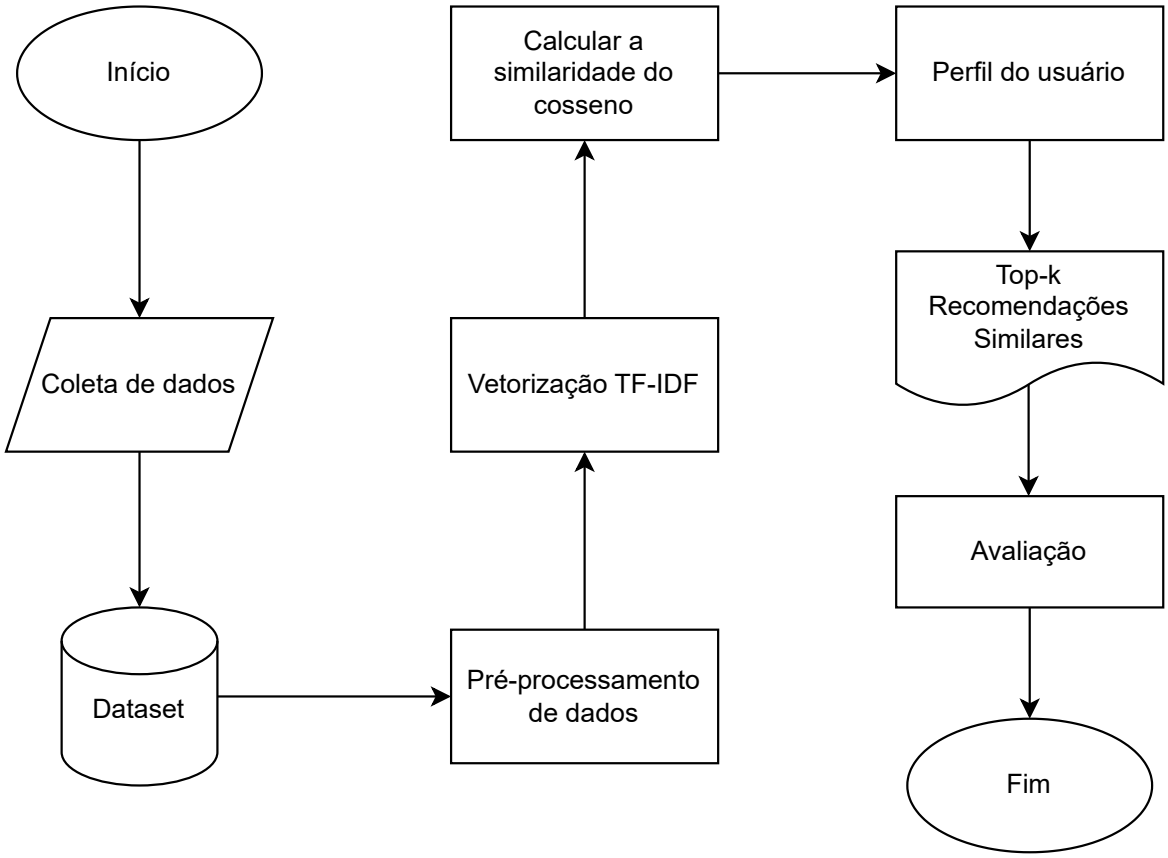
O peso do TF-IDF é calculado pela multiplicação das duas medidas obtidas anteriormente, como demonstrado na Equação 2.10. Assim, quanto maior o peso, mais relevante é a palavra em questão dentro do *corpus* (CHINY *et al.*, 2022).

$$\mathrm{tfidf}_i = \mathrm{tf}_i \times \mathrm{idf}_i \tag{2.10}$$

Nesta pesquisa, foi adotada uma abordagem de filtragem baseada em conteúdo, utilizando o modelo TF-IDF e a similaridade do cosseno. Essa escolha se justifica pela capacidade do método de antecipar as preferências dos usuários e gerar recomendações de filmes com base no conteúdo extraído de seus históricos ou interações (PERMANA; WIBOWO, 2023).

A Figura 5 ilustra o fluxo típico de funcionamento de um sistema de recomendação baseado em TF-IDF e similaridade do cosseno.

Figura 5 – Fluxograma do Sistema de Recomendação



Fonte: Adaptado de PERMANA; WIBOWO (2023).

Com base na Figura 5, o processo inicia-se com a coleta de dados, que podem estar organizados em *datasets* contendo informações sobre itens e usuários. Em seguida, realiza-se o pré-processamento, no qual os dados passam por limpeza e filtragem para garantir consistência e qualidade. Na etapa de representação, aplica-se a vetorização TF-IDF, que converte descrições textuais em vetores numéricos ponderados. A similaridade entre itens é então estimada por meio da métrica de similaridade do cosseno, permitindo identificar aqueles com características próximas. Com base nas avaliações positivas de cada usuário, constrói-se um perfil que reflete suas preferências, possibilitando gerar uma lista de recomendações contendo os top-k itens mais similares. Por fim, o sistema passa por uma etapa de avaliação para verificar seu desempenho e efetividade.

A vantagem do TF-IDF é a sua simplicidade e por precisar de apenas um volume baixo de dados de treinamento para implementar o sistema de recomendação. Isso lhe permite ser facilmente implementado e utilizado em diversos sistemas computacionais (CHINY *et al.*, 2022).

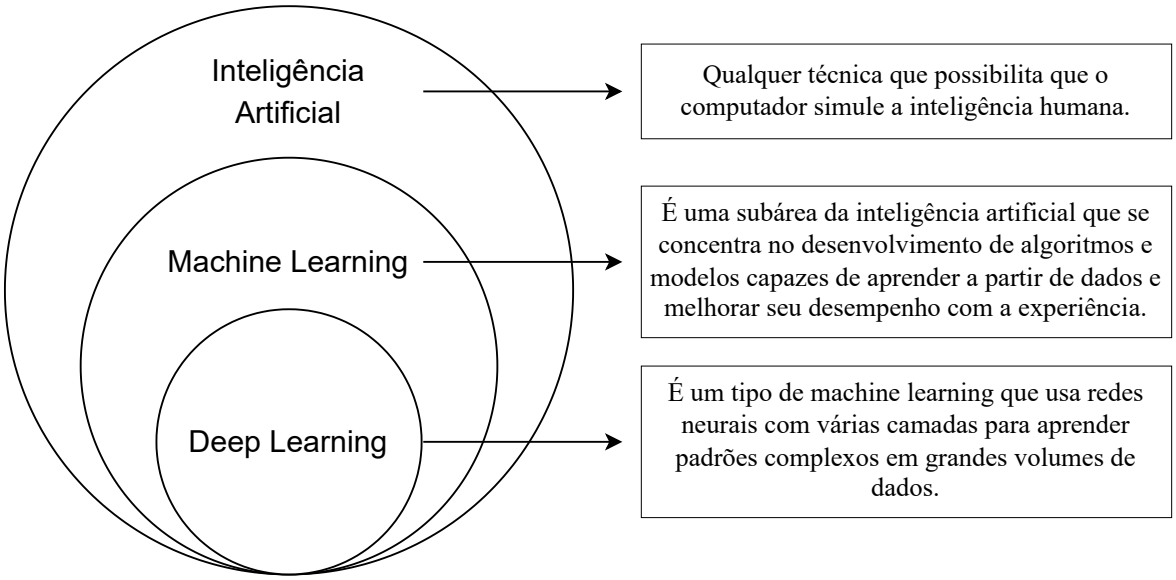
2.5 Modelos baseados em deep learning

O uso de DL em sistemas de recomendação tem se tornado cada vez mais comum e continua em constante evolução. Nesta seção, são apresentados os conceitos fundamentais dessa área, bem como os modelos baseados em DL implementados neste trabalho.

2.5.1 Deep learning

Antes de tudo, é importante conhecer alguns termos frequentemente usados quando o assunto é Inteligência Artificial (IA), ML e DL. Para facilitar a compreensão da relação entre esses três, pode-se representá-los visualmente como áreas sobrepostas, onde uma está contida dentro da outra, conforme ilustrado na Figura 6 (SOUSA, 2022).

Figura 6 – Relação entre as áreas de IA



Fonte: Adaptado de SOUSA (2022).

A IA é uma área que abrange um conjunto de técnicas que simulam a inteligência humana. O ML é um subconjunto da IA que aprende com os dados disponíveis para executar tarefas específicas, identificando padrões. Já o DL consiste de redes neurais profundas, ou seja, de múltiplas camadas (NGUYEN *et al.*, 2019).

Os sistemas de recomendação que utilizam essa abordagem transformaram significativamente a área, ao incorporar redes neurais para processar *embeddings* e oferecer recomendações altamente personalizadas com base no comportamento do usuário. A escolha cuidadosa da arquitetura dessas redes é essencial para o desenvolvimento desses sistemas (LI *et al.*, 2023).

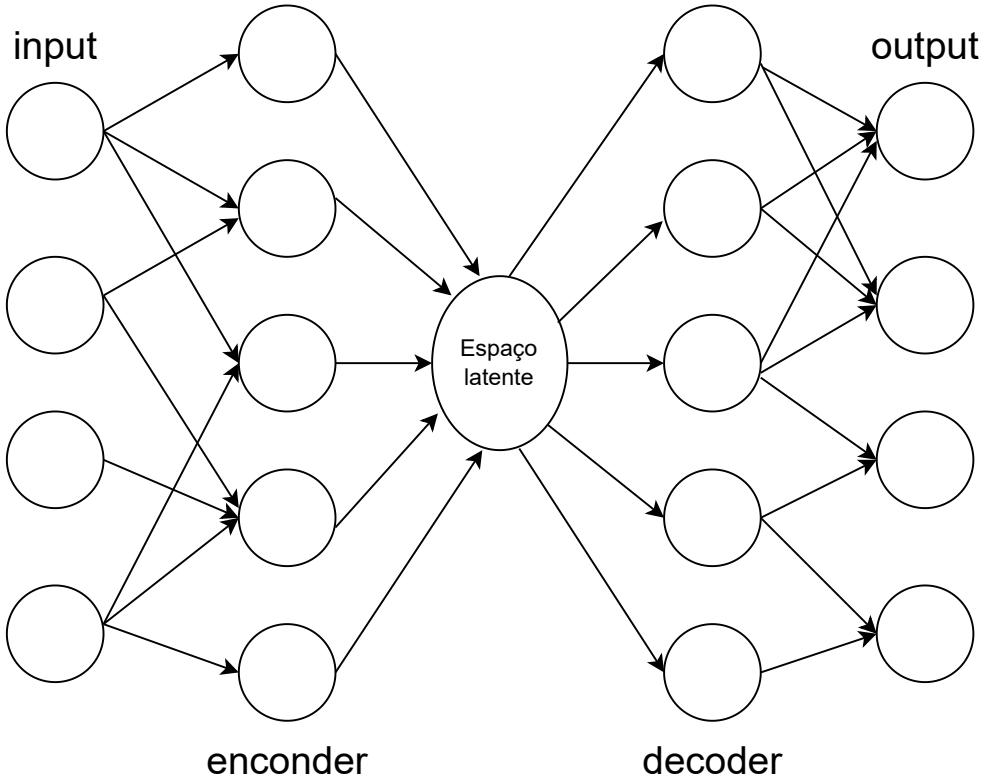
Essa técnica consegue capturar relações não lineares e complexas entre usuário e item, possibilitando a codificação de abstrações mais complexas por meio das camadas superiores das redes (ZHANG *et al.*, 2017).

Dessa forma, o uso de DL em sistemas de recomendação tem se mostrado fundamental para aprimorar a personalização e a eficácia das recomendações, pois permite representar de forma mais precisa as preferências dos usuários.

2.5.2 AutoRec

Um *Autoenconder* (AE) é um tipo de rede neural projetado para reconstruir a entrada na saída. Para isso, ele utiliza uma camada intermediária oculta, que representa um código comprimido da entrada original. Essa rede é composta por duas partes principais. A primeira é o codificador, responsável por transformar a entrada em uma representação reduzida. A segunda é o decodificador, que utiliza essa representação para tentar reconstruir a entrada original. A Figura 7 apresenta a arquitetura básica de um *autoenconder* (GOODFELLOW *et al.*, 2016).

Figura 7 – Arquitetura básica de uma rede neural



Fonte: Elaborada pelo autor.

Na filtragem colaborativa baseada em avaliações, considera-se um conjunto com

m usuários e n itens, representados por uma matriz de avaliações parcialmente preenchida $R \in \mathbb{R}^{m \times n}$. Cada usuário $u \in U = \{1, \ldots, m\}$ pode ser descrito por um vetor parcialmente observado $r^{(u)} = (R_{u1}, \ldots, R_{un}) \in \mathbb{R}^n$, que contém suas avaliações para os itens, onde algumas podem estar ausentes. De maneira semelhante, cada item $i \in I = \{1, \ldots, n\}$ é representado por um vetor $r^{(i)} = (R_{1i}, \ldots, R_{mi}) \in \mathbb{R}^m$, que agrega as avaliações que os usuários deram a esse item, também com possíveis valores ausentes (SEDHAIN *et al.*, 2015).

O AutoRec, que pode ser tanto baseado em itens quanto em usuários, é capaz de receber como entrada um vetor parcialmente observado $r^{(i)}$ (ou $r^{(u)}$), projetá-lo em um espaço latente de menor dimensão e, em seguida, reconstruí-lo na saída. O intuito é preencher as avaliações ausentes, isto é, prever as notas não observadas com o objetivo de gerar recomendações. Formalmente, dado um conjunto S de vetores em $\mathbb{R}^d$ e um inteiro positivo $k \in \mathbb{N}^+$, um AE busca resolver a função de reconstrução descrita a seguir (SEDHAIN *et al.*, 2015):

$$h(r; \theta) = f\left(W \cdot g(Vr + \mu) + b\right). \tag{2.11}$$

Nesse contexto, $h(r; \theta)$ representa a reconstrução da entrada $r \in \mathbb{R}^d$. As funções $f(\cdot)$ e $g(\cdot)$ são funções de ativação aplicadas na saída e na camada oculta, respectivamente. O conjunto de parâmetros do modelo é dado por $\theta = \{W, V, \mu, b\}$, onde $W \in \mathbb{R}^{d \times k}$ e $V \in \mathbb{R}^{k \times d}$ são matrizes de transformação (pesos) e $\mu \in \mathbb{R}^k$ e $b \in \mathbb{R}^d$ são vetores de viés. Esse modelo é uma rede neural do tipo AE, que possui apenas uma camada oculta com dimensão k no espaço latente. O treinamento do AE é realizado utilizando o algoritmo de retropropagação (*backpropagation*), ajustando os parâmetros θ para minimizar o erro de reconstrução (SEDHAIN *et al.*, 2015).

O modelo AutoRec baseado em itens utiliza o *autoencoder* descrito pela Equação 2.11 que é aplicado sobre o conjunto de vetores ${r^{(i)}}_{i=1}^{n}$. Primeiro, leva-se em consideração o fato de que cada $r^{(i)}$ é parcialmente observado, atualizando durante a retropropagação apenas os pesos associados às entradas observadas, como é comum nas abordagens de fatoração de matrizes. Segundo, aplica-se uma regularização aos parâmetros aprendidos a fim de evitar *overfitting* nas avaliações observadas. Formalmente, a função objetivo para o modelo AutoRec baseado em itens, fica definida da seguinte forma, assumindo um parâmetro de regularização $\lambda > 0$ (SEDHAIN *et al.*, 2015):

$$\min_{\theta} \sum_{i=1}^{n} \left\| r^{(i)} - h(r^{(i)}; \theta) \right\|^2 + \frac{\lambda}{2} \left(\|W\|_F^2 + \|V\|_F^2 \right). \tag{2.12}$$

Nesse contexto, $\|\cdot\|_{\mathcal{O}}^2$ indica que o erro de reconstrução é calculado apenas sobre as avaliações observadas dos vetores de entrada, ignorando as posições onde os dados estão ausentes (SEDHAIN *et al.*, 2015).

O AutoRec baseado em usuários segue o mesmo princípio do AutoRec baseado em itens, porém recebe como entrada vetores de classificações de usuários, em vez de itens. Cada vetor $r(u) \in \mathbb{R}^n$ representa as avaliações que um usuário u atribuiu aos itens, sendo, em sua maioria, valores ausentes. O modelo é treinado para reconstruir esse vetor e prever as avaliações faltantes, utilizando apenas as entradas observadas durante o *backpropagation*. A função de perda é semelhante à do AutoRec baseado em item, mas aplicada ao conjunto de vetores de usuários. O AutoRec baseado em usuário é útil quando há mais itens do que usuários ou quando o foco está em capturar o perfil comportamental de cada usuário.

2.5.3 Transformes

Mecanismos de atenção foram inspirados em como os seres humanos prestam atenção a estímulos — de forma voluntária (quando algo é focado voluntariamente) e involuntária (quando algo chama a atenção automaticamente). Em modelos de atenção, representa-se a atenção voluntária como consultas (*queries*), que definem o que está procurando ou tentando focar. Já a atenção involuntária é representada pelas chaves (*keys*), que estão associadas às entradas sensoriais, ou seja, aos dados que estão disponíveis. Cada entrada sensorial também tem um valor (*value*), que é a informação real que será processada. O mecanismo de atenção compara cada consulta com todas as chaves e calcula a contribuição de cada valor. Esse processo permite que o modelo "preste atenção" nos dados mais relevantes, de acordo com o contexto da consulta. Isso faz com que os mecanismos de atenção sejam muito mais adaptáveis e sensíveis ao contexto do que camadas tradicionais, como as densas ou aquelas que resumem informações, como as camadas de agregação espacial (ZHANG *et al.*, 2023).

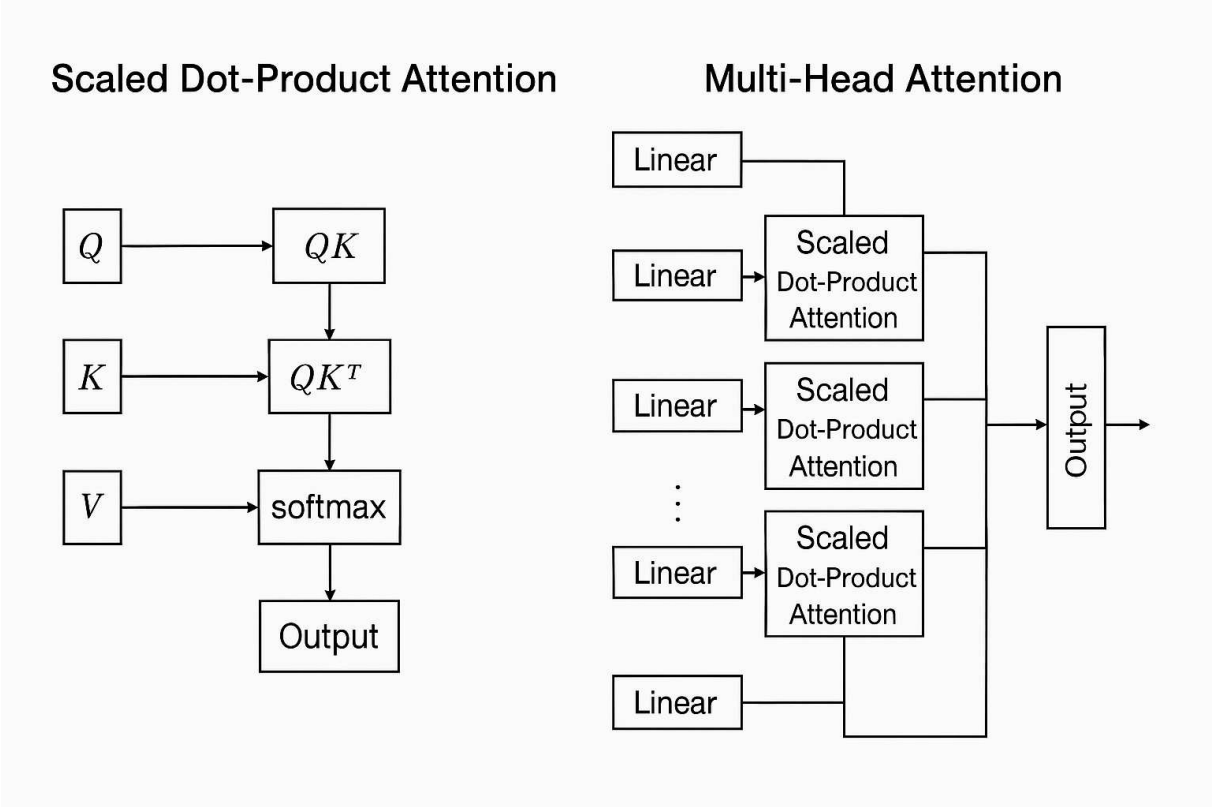
Portanto, uma função de atenção pode ser definida como o mapeamento de uma consulta e um conjunto de pares de valores-chave para uma saída, onde a consulta, as chaves e a saída são vetores. A saída é determinada pela soma ponderada dos valores, em que o peso de cada valor é obtido por meio de uma função de compatibilidade entre a consulta e a chave correspondente. Essa atenção mencionada é conhecida como *Scaled Dot-Product Attention* e pode ser visualizada na Figura 8. A entrada é composta por consultas e chaves de dimensão d_k e valores da dimensão d_v. Portanto, são realizados os produtos escalares entre a consulta e as

chaves, dividindo-se cada um por $\sqrt{d_k}$ e aplicando-se uma função *softmax* para adquirir os pesos dos valores. Efetivamente, o que ocorre é o cálculo da função de atenção em um conjunto de consultas, agregadas simultaneamente em uma matriz Q. Além disso, as chaves e os valores também são agrupados nas matrizes K e V. Assim, o cálculo da matriz de resultados é realizado (VASWANI *et al.*, 2017):

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{2.13}$$

Em vez de empregar uma única função de atenção utilizando chaves, valores e consultas com dimensão d_{model} − sendo d_{model} a dimensão dos vetores de entrada e saída no Transformer – é possível realizar projeções lineares das consultas, chaves e valores h vezes, cada uma com dimensões d_k, d_k e d_v, respectivamente. Dessa forma, a função de atenção é aplicada separadamente a cada uma das versões transformadas de consultas, chaves e valores, resultando em vetores de saída com dimensão d_v. Depois, eles são concatenados e novamente projetados, resultando nos valores finais, conforme demonstrado na Figura 8 (VASWANI *et al.*, 2017).

Figura 8 – À esquerda: *Scaled Dot-Product Attention* à direita: *Multi-Head Attention*



Fonte: Imagem gerada por IA via DALL·E (OpenAI, 2025), com prompt elaborado pelo autor.

Essa nova técnica é chamada de atenção *multi-head*, possibilitando que atenda, em

conjunto, aos dados de diferentes representações de subespaços em posições distintas (VASWANI *et al.*, 2017):

$$\text{MultiHead}(Q,K,V) = \text{Concat}(\text{head}_1,\ldots,\text{head}_h)W^O, \tag{2.14}$$

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V). \tag{2.15}$$

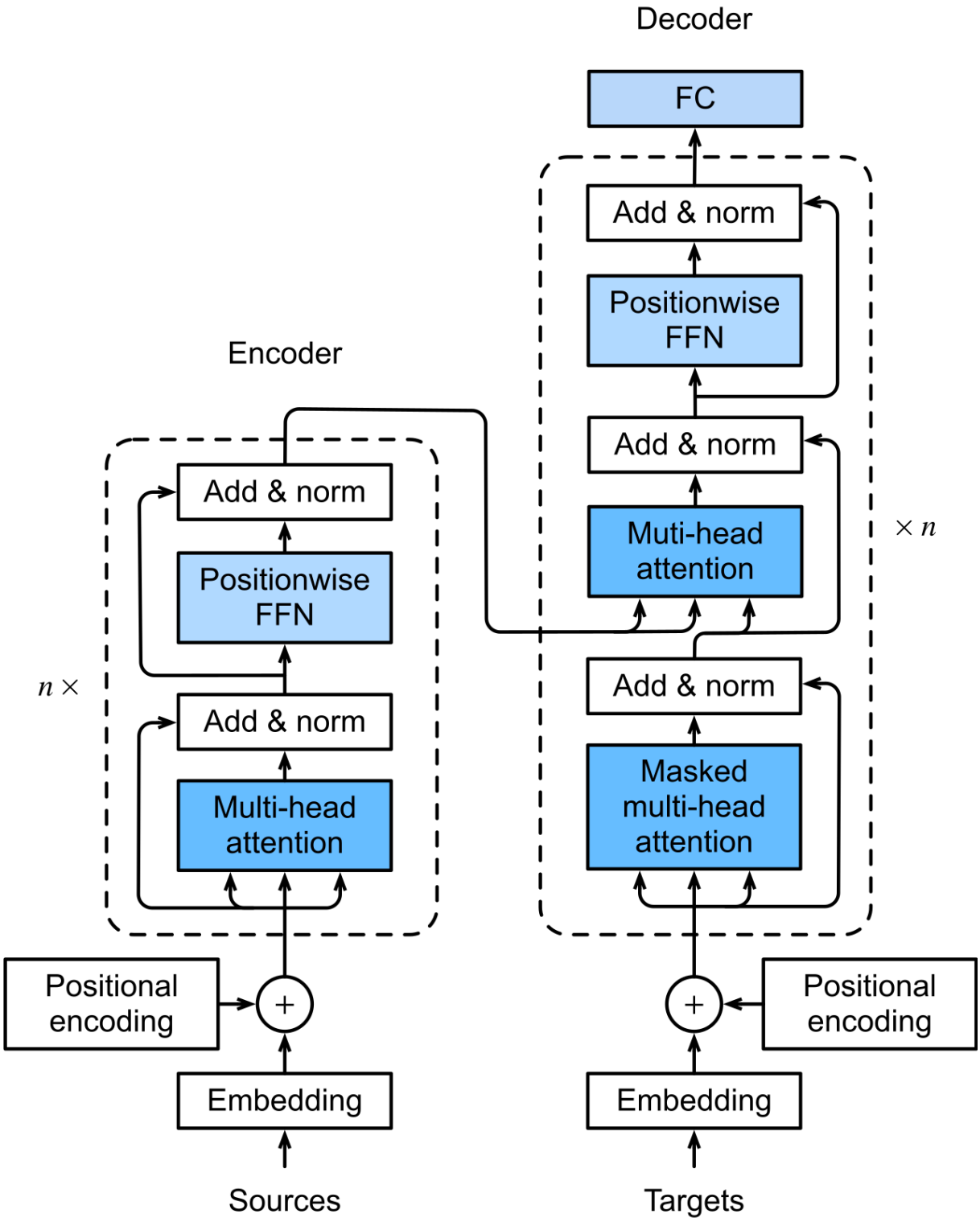
Onde os parâmetros $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ são matrizes utilizadas para realizar as projeções. Nesse contexto, os *embeddings* são utilizados para converter a entrada em *tokens* e *tokens* de saída para vetores de dimensão d_{model}. A saída do decodificador é projetada por uma camada linear aprendida e, em seguida, normalizada por uma função *softmax*, resultando em uma distribuição de probabilidade sobre o vocabulário para predição do próximo *token*. Como exemplo da arquitetura codificador-decodificador, a estrutura geral do Transformer (transformador) é mostrada na Figura 9 (VASWANI *et al.*, 2017).

Nota-se que o transformador é formado por dois componentes principais: o codificador e o decodificador. Antes de passarem por esses módulos, os *embeddings* das sequências de entrada (origem) e de saída (destino) recebem informações de posição por meio da codificação posicional. Em seguida, esses dados são enviados ao codificador e ao decodificador, que consistem em camadas empilhadas com mecanismos baseados em autoatenção (ZHANG *et al.*, 2023).

O codificador converte uma sequência de entrada de símbolos $(x_1,\ldots,x_n)$ em uma sequência de representações contínuas $z = (z_1,\ldots,z_n)$. A partir dessas representações, o decodificador constrói uma nova sequência de saída $(y_1,\ldots,y_m)$, gerando os símbolos um de cada vez. A cada estágio, o modelo é autorregressivo, pois os símbolos formados anteriormente são consumidos como entrada adicional ao gerar a próxima saída. O *Transformer* possui essa arquitetura geral, utilizando um mecanismo de autoatenção empilhada e camadas conectadas e específicas para o codificador e o decodificador, demonstradas nas metades esquerda e direita da Figura 9, respectivamente (VASWANI *et al.*, 2017).

O codificador é constituído por uma pilha de $N = 6$ camadas idênticas. Cada camada possui 2 subcamadas. A primeira subcamada aplica um mecanismo de autoatenção e a segunda é formada por uma rede *feed-forward* posicional, simples e totalmente conectada. Ambas as subcamadas são precedidas por conexões residuais e, depois disso, passam por uma etapa de

Figura 9 – A arquitetura do transformador



Fonte: ZHANG et al. (2023).

normalização. O decodificador é composto também por uma pilha de $N = 6$ camadas idênticas, cada uma com 2 subcamadas. Além dessas, o decodificador adiciona uma terceira subcamada, responsável por aplicar o mecanismo de atenção com múltiplos focos sobre a saída gerada pelo

codificador. Similarmente ao codificador, são utilizadas conexões residuais em torno de cada uma das subcamadas, seguidas pela normalização da camada. Além disso, ocorre a modificação da subcamada de autoatenção do decodificador para impedir que as posições atinjam as posições subsequentes. Isso, e o fato de que os *embeddings* de saída são deslocados em uma posição, assegura que as predições para a posição i dependam somente das saídas em posições inferiores a i (VASWANI *et al.*, 2017).

Como esse modelo não possui recorrência nem convolução, para implementar a ordem da sequência é necessário inserir algumas informações sobre a posição relativa ou absoluta dos *tokens* na sequência. Para essa finalidade, adicionam-se codificações posicionais aos *embeddings* de entrada nas bases das pilhas do codificador e do decodificador. Essas codificações possuem a mesma dimensão d_{model} dos *embeddings*, o que permite a soma entre elas. Há várias opções de codificações posicionais; entretanto, neste trabalho são usadas funções seno e cosseno de frequências distintas, como mostrado a seguir (VASWANI et al., 2017):

$$PE(\text{pos}, 2i) = \sin\left(\frac{\text{pos}}{10000^{2i \cdot d_{\text{model}}}}\right), \tag{2.16}$$

$$PE(\text{pos}, 2i+1) = \cos\left(\frac{\text{pos}}{10000^{2i \cdot d_{\text{model}}}}\right). \tag{2.17}$$

Em que *pos* representa a posição e i indica a dimensão. Isso significa que cada dimensão da codificação posicional é representada por uma função senoide. Os comprimentos de onda geram uma progressão geométrica de 2π a $10000 \cdot 2\pi$. Essa função foi escolhida, pois, hipoteticamente, ela possibilita que o modelo aprenda facilmente a identificar as posições relativas, já que, para qualquer deslocamento fixo k, $PE_{\text{pos}+k}$ pode ser representado como uma função linear de PE_{pos} (VASWANI *et al.*, 2017).

2.5.4 BERT

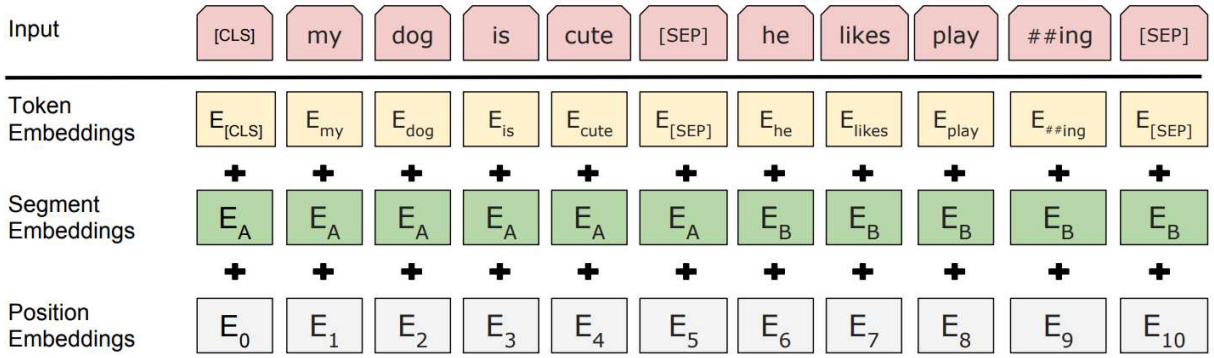
Bidirectional Encoder Representations from Transformers (BERT), ou Representações Bidirecionais Codificadoras baseadas em Transformers, é estruturado em duas fases principais: pré-treinamento e *fine-tuning* (ajuste fino). Sua arquitetura, como o próprio nome indica, é um codificador Transformer bidirecional multicamadas, baseado na elaboração original descrita em Vaswani et al. (2017). Na etapa de pré-treinamento, o modelo é exposto a grandes

volumes de dados não rotulados, realizando tarefas específicas para aprender representações linguísticas gerais. Em seguida, no *fine-tuning*, os parâmetros aprendidos são reutilizados como base e refinados com dados rotulados, específicos de cada tarefa final. Embora diferentes tarefas posteriores utilizem modelos ajustados separadamente, todos partem de uma mesma base de parâmetros pré-treinados. Uma característica que diferencia o BERT é sua arquitetura unificada, que se mantém praticamente inalterada entre o pré-treinamento e as aplicações nas tarefas finais, garantindo consistência e versatilidade em diversos contextos (DEVLIN *et al.*, 2019).

Para que o BERT seja capaz de lidar com diferentes tipos de tarefas posteriores (*downstream*), sua representação de entrada foi projetada para identificar de forma clara tanto uma única sentença quanto um par de sentenças (como em pares pergunta-resposta), reunidas em uma única sequência de *tokens*. Ao longo deste trabalho, uma *sentença* pode ser qualquer trecho contínuo de texto, e não necessariamente uma sentença linguística completa. Uma *sequência* refere-se à cadeia de *tokens* de entrada para o modelo, que pode conter uma única sentença ou duas sentenças agrupadas (DEVLIN *et al.*, 2019).

O BERT emprega *embeddings* para codificar as informações fornecidas como entrada. Toda sequência inicia com um *token* especial de classificação, denominado [CLS], cujo vetor oculto final serve como uma representação global da sequência em tarefas de classificação. Quando há duas sentenças, elas são concatenadas em uma única sequência, separadas por um *token* especial [SEP]. Além disso, cada *token* recebe um *embedding* de segmento aprendido, que indica se ele pertence à primeira ou à segunda sentença (sentença A ou B). A representação de entrada de cada *token* é construída a partir da soma de três componentes: o *embedding* do próprio *token*, o *embedding* de posição e o *embedding* de segmento correspondente. Essa estrutura pode ser visualizada na Figura 10 (DEVLIN *et al.*, 2019).

Figura 10 – Representação de uma entrada BERT



Fonte: Devlin et al. (2019).

O pré-treinamento do BERT envolve duas tarefas não supervisionadas: a modelagem de linguagem mascarada, em que o modelo aprende a prever palavras ocultas aleatoriamente dentro de uma sequência de texto, e a previsão da próxima frase, que permite ao modelo capturar relações entre sentenças consecutivas (ZHANG *et al.*, 2023). Essas tarefas são fundamentais para que o modelo aprenda representações linguísticas profundas, que posteriormente são refinadas no *fine-tuning* para tarefas específicas.

O pré-treinamento do BERT requer a preparação de um conjunto de dados no formato adequado para atender às duas tarefas utilizadas nesse processo. O BERT original foi pré-treinado com a junção dos conjuntos de textos BookCorpus e Wikipedia em inglês. Esses conjuntos de dados são de grande escala, contendo aproximadamente 800 milhões de palavras no BookCorpus e cerca de 2,5 bilhões de palavras na Wikipedia (ZHANG *et al.*, 2023).

O processo de *fine-tuning* do BERT é considerado simples, uma vez que o mecanismo de autoatenção presente na arquitetura Transformer permite que o modelo seja adaptado a diversas tarefas posteriores, independentemente de envolverem textos individuais ou pares de texto, bastando modificar as entradas e saídas conforme a tarefa. Em aplicações que utilizam pares de textos, a abordagem tradicional consiste em codificar os textos separadamente antes de aplicar a atenção cruzada bidirecional. Diferentemente disso, o BERT integra essas etapas ao utilizar a autoatenção sobre a sequência concatenada dos dois textos, o que possibilita que a interação entre as sentenças ocorra de forma bidirecional de maneira natural (DEVLIN *et al.*, 2019).

Durante o *fine-tuning*, as entradas e saídas específicas de cada tarefa são incorporadas ao modelo, e todos os parâmetros são atualizados de forma conjunta. As sentenças *A* e *B*, usadas no pré-treinamento, correspondem, nas diferentes tarefas, a pares de sentenças em tarefas de paráfrase, a pares hipótese-premissa em inferência textual, a pares pergunta e trecho em tarefas de perguntas e respostas, ou ainda a pares compostos por um texto e uma entrada nula em tarefas como classificação de texto ou marcação de sequência. No momento da saída, as representações geradas para cada *token* são encaminhadas para uma camada final em tarefas que exigem previsões *token* a *token*, como é o caso da marcação de sequência ou da resposta a perguntas. Já a representação do *token* [CLS] é utilizada em tarefas de classificação, como inferência textual ou análise de sentimentos. Em comparação ao pré-treinamento, o *fine-tuning* exige consideravelmente menos recursos computacionais (DEVLIN *et al.*, 2019).

2.5.5 RoBERTa

O *Robustly Optimized BERT Approach* (RoBERTa) é uma variação do modelo BERT desenvolvida com o objetivo de aprimorar o processo de pré-treinamento e buscando a obtenção de melhores resultados em tarefas de NLP. O BERT original estava subtreinado, e diversas decisões no seu pré-treinamento podiam ser melhoradas. O modelo RoBERTa adota um esquema de mascaramento dinâmico, que gera diferentes máscaras de *tokens* a cada etapa do treinamento, diferentemente as máscaras fixas usadas no BERT. Além disso, o modelo remove a tarefa de *Next Sentence Prediction* (NSP), demonstrando que sua exclusão não compromete o desempenho. O RoBERTa é treinado com lotes maiores, um número maior de passos de otimização e uma quantidade significativamente maior de dados (aproximadamente 160 GB), combinando múltiplas fontes textuais para ampliar a diversidade do treinamento. Os resultados indicam que o RoBERTa supera o BERT em *benchmarks* padrão, mantendo boa performance mesmo após extenso treinamento (LIU *et al.*, 2019).

2.5.6 DistilBERT

O DistilBERT é uma versão compacta do modelo BERT, projetada para ser significativamente mais eficiente, mantendo uma performance similar. Com 40% a menos de tamanho e 60% mais rápido, ele conserva 97% das capacidades de compreensão linguística do modelo original. A principal abordagem por trás da criação do DistilBERT é a técnica de destilação de conhecimento, onde um modelo menor, chamado aluno, aprende a imitar as previsões de um modelo maior, o professor. Isso é feito aproveitando a probabilidade das *soft targets* geradas pelo modelo professor, o que permite que o modelo menor seja treinado de maneira mais eficiente, mesmo com uma arquitetura reduzida. O DistilBERT possui metade das camadas do BERT, o que reduz significativamente os custos computacionais sem afetar muito a precisão. Além disso, remove certos componentes da arquitetura, como as incorporações do tipo de *token* e o *pooler*, contribuindo para maior eficiência. Essa versão mais leve é mais indicada para aplicações em dispositivos com recursos limitados, oferecendo uma solução mais rápida e econômica para tarefas de NLP (SANH *et al.*, 2020).

2.5.7 DistilRoBERTa

O DistilRoBERTa é uma versão mais leve do modelo RoBERTa-base, obtida por meio da técnica de destilação de conhecimento, que visa reduzir o tamanho e a complexidade do modelo original, mantendo boa performance. Esse modelo possui (Hugging Face, 2019):

- 6 camadas (metade das 12 camadas do RoBERTa-base);
- 768 dimensões;
- 12 cabeças de atenção;
- e aproximadamente 82 milhões de parâmetros, em comparação com os 125 milhões do RoBERTa-base.

Essa redução torna o DistilRoBERTa mais rápido, aproximadamente o dobro da velocidade do modelo original, facilitando sua aplicação em cenários onde recursos computacionais ou tempo são limitados. O modelo é sensível a maiúsculas e minúsculas, o que é importante para muitas tarefas linguísticas. Ele é especialmente indicado para tarefas que envolvem análise de sentenças completas, como classificação de sequência, classificação de *tokens* e resposta a perguntas, podendo ser ajustado para diferentes aplicações (Hugging Face, 2019).

2.6 Métricas de avaliação

Neste trabalho, utilizam-se algumas das métricas de desempenho no contexto de sistemas de recomendação mais conhecidas. Entre elas estão aquelas que avaliam a qualidade do ranking de itens recomendados, como a *precision*, o *recall* e o *Normalized Discounted Cumulative Gain* (NDCG). Além disso, também são considerados aspectos de eficiência computacional, como tempo de treinamento, tempo de inferência e consumo de memória. As próximas subseções detalham cada uma dessas métricas.

2.6.1 Precision

Precision é uma métrica que mede a proporção de recomendações relevantes entre todas as recomendações realizadas, ou seja, indica a capacidade do sistema de recomendar itens que realmente agradam ao usuário. Assim, quanto mais próxima de 1 estiver a *precision*, maior será essa proporção. Essa métrica é formalmente definida como (MINGRONI, 2024):

$$\text{Precision} = \frac{TP}{TP+FP}. \tag{2.18}$$

True Positive (TP) (Verdadeiro Positivo) representa os itens que foram recomendados e que, de fato, são relevantes. Já Falso Positivo (FP) corresponde aos itens recomendados que não são do interesse do usuário (MINGRONI, 2024).

2.6.2 Recall

O *recall* mensura a capacidade do sistema de recomendar itens relevantes, considerando todos os itens que poderiam ser recomendados. Essa métrica reflete a abrangência do sistema em selecionar opções que são do interesse do usuário. Assim, valores altos de *recall* demonstram a competência do sistema em captar uma ampla variedade de itens do interesse do usuário. O *recall* é calculado por meio da seguinte equação (MINGRONI, 2024):

$$\text{Recall} = \frac{TP}{TP+FN}. \tag{2.19}$$

Na Equação 2.19, TP representa os verdadeiros positivos, enquanto Falso Negativo (FN) indica os itens que não foram recomendados, mas que eram relevantes para o usuário (MINGRONI, 2024).

2.6.3 NDCG

A utilização de uma função de decaimento exponencial para a utilidade indica que apenas os itens mais bem posicionados na lista recebem atenção dos usuários, enquanto aqueles em posições inferiores tendem a ser negligenciados. No entanto, essa suposição não é válida para todos os contextos, especialmente em sistemas de recomendação de notícias, nos quais é comum que os usuários explorem diversos itens, mesmo em posições mais baixas. Nesses cenários, recomenda-se aplicar uma taxa de desconto menos agressiva. Uma métrica apropriada para esses casos é o *Discounted Cumulative Gain* (DCG). Nessa métrica, o fator de desconto para um item j é definido por $\log_2(v_j+1)$, sendo v_j a posição do item j no conjunto de teste I_u. O DCG é, então, calculado conforme a seguinte equação (AGGARWAL, 2016):

$$\text{DCG} = \frac{1}{m}\sum_{u=1}^{m}\sum_{j\in I_u}\frac{g_{uj}}{\log_2(v_j+1)}. \tag{2.20}$$

Neste contexto, g_{uj} representa a utilidade (ou ganho) associada ao consumo do item j pelo usuário u. Essa utilidade costuma ser expressa por uma função exponencial da relevância do item, como nos casos de avaliações positivas ou acertos do usuário (AGGARWAL, 2016):

$$g_{uj} = 2^{rel_{uj}} - 1. \tag{2.21}$$

O termo rel_{uj} corresponde à relevância real do item j para o usuário u, determinada por meio de uma função heurística baseada em avaliações ou interações. Frequentemente, utilizam-se diretamente os valores brutos das avaliações. É prática comum restringir o cálculo do DCG a uma lista de recomendação de tamanho limitado L, em vez de considerar todos os itens disponíveis (AGGARWAL, 2016):

$$\text{DCG} = \frac{1}{m} \sum_{u=1}^{m} \sum_{j \in I_u, v_j \leq L} \frac{g_{uj}}{\log_2(v_j + 1)}. \tag{2.22}$$

O princípio aqui é que as listas de recomendação apresentadas aos usuários geralmente não excedem o tamanho L. Para normalizar esse valor, utiliza-se o NDCG, definido como a razão entre o DCG e seu valor ideal, conhecido como *Ideal Discounted Cumulative Gain* (IDCG) (AGGARWAL, 2016):

$$\text{NDCG} = \frac{\text{DCG}}{\text{IDCG}}. \tag{2.23}$$

O cálculo do IDCG segue o mesmo procedimento do DCG, com a diferença de que utiliza a ordenação ideal dos itens, baseada na relevância real, como critério de posicionamento na lista (AGGARWAL, 2016). Portanto, o NDCG avalia a qualidade do ranqueamento considerando tanto a relevância dos perfis quanto suas posições na lista, valorizando mais os perfis relevantes que aparecem nos primeiros lugares (SANGEETHA *et al.*, 2025).

2.6.4 Tempo de treinamento

O tempo de treinamento é um aspecto importante, já que a maioria dos sistemas de recomendação passa por uma etapa de aprendizado separada da fase de teste. Por exemplo, em um sistema baseado em fatoração de matrizes, é necessário calcular os fatores latentes. O tempo total gasto nesse processo é utilizado como uma das métricas de avaliação. Normalmente,

o treinamento é feito offline, e por isso, em cenários reais, considera-se aceitável que ele dure algumas horas (AGGARWAL, 2016).

2.6.5 Tempo de inferência

O tempo de inferência refere-se ao intervalo necessário para que um modelo gere uma previsão ou recomendação a partir de dados já processados, após a conclusão do treinamento. Esta métrica é fundamental em sistemas que operam em tempo real ou que exigem baixa latência nas respostas, como plataformas de *streaming*, *e-commerce* e redes sociais. O tempo de inferência depende tanto da complexidade do modelo quanto da quantidade de dados processados no momento da consulta (AGGARWAL, 2016).

2.6.6 Consumo de memória

O consumo de memória representa a quantidade de recursos computacionais, especialmente a *Random Access Memory* (RAM) e, em cenários que utilizam aceleração por *hardware*, memória da *Graphics Processing Unit* (GPU), exigida durante os processos de treinamento e inferência. Segundo Aggarwal (2016), esse fator impacta diretamente a escolha da infraestrutura e a escalabilidade de sistemas de recomendação tradicionais. No contexto de modelos baseados em DL, o consumo de memória se torna ainda mais crítico, devido à necessidade de armazenamento de *embeddings* densos, parâmetros de redes profundas e *buffers* intermediários, elevando significativamente os custos operacionais (NAUMOV *et al.*, 2019).

3 METODOLOGIA

Este capítulo descreve a metodologia adotada neste trabalho, detalhando os procedimentos seguidos, o tipo de pesquisa realizada, a implementação dos modelos, os critérios de avaliação e os parâmetros experimentais utilizados. O objetivo é assegurar a reprodutibilidade e a validade dos resultados obtidos.

3.1 Tipo de pesquisa

Este trabalho é uma pesquisa de natureza explicativa, pois procura compreender as diferenças de desempenho entre diferentes abordagens de recomendação. Este estudo é fundamentado em fontes secundárias, priorizando publicações recentes ou autores consolidados. Essa pesquisa bibliográfica abrangeu artigos científicos, livros, teses, monografias e documentos acadêmicos indexados em bases como o Google Acadêmico. Além disso, este trabalho é caracterizado como uma pesquisa experimental e adota uma abordagem quantitativa, já que foram implementados algoritmos de recomendação em um ambiente controlado para a coleta de dados relacionados ao desempenho preditivo e à eficiência computacional. Os dados foram apresentados e comparados, permitindo uma análise objetiva dos resultados.

3.2 Dataset

Para a realização da avaliação experimental dos algoritmos de recomendação, foi utilizado o *dataset The Movies Dataset*, publicamente acessível na plataforma Kaggle e disponibilizado por Rounak Banik. Esse *dataset* foi escolhido porque apresenta dados explícitos e informações textuais, como sinopses, permitindo a implementação da filtragem colaborativa e da filtragem baseada em conteúdo. Além disso, o volume de dados é apropriado para avaliar o desempenho de modelos tradicionais e os baseados em DL, mas também para verificar a escalabilidade dos modelos.

Esse *dataset* é uma combinação das informações coletadas do *The Movie Database* (TMDB) e do *dataset* MovieLens do grupo de pesquisa GroupLens. Os detalhes dos filmes foram extraídos da *Application Programming Interface* (API) aberta do TMDB, mas não são endossados nem certificados por ele. O *dataset* é constituído por metadados de 45.000 filmes e possui filmes lançados até julho de 2017. Ele é composto por diversas tabelas, contendo informações como: países, empresas de produção, idiomas, datas de lançamento, pôsteres,

receita, orçamento, palavras-chave do enredo, equipe técnica, elenco, contagem de votos e médias de votos do TMDB. O *dataset* contém 26 milhões de avaliações feitas por 270 mil usuários para os 45 mil filmes. Além disso, os *ratings* (avaliações) vão de 1 a 5 e foram coletados no site oficial do GroupLens (BANIK, 2017).

3.3 Pré-processamento dos dados

Foi implementada uma função de pré-processamento unificada para todos os modelos, que inclui a leitura e limpeza dos dados do The Movies Dataset. Foram consideradas apenas avaliações únicas por par usuário-filme, e aplicou-se uma filtragem iterativa para manter usuários e itens com pelo menos 100 interações, garantindo assim a representatividade e densidade da matriz de interações.

Para os dados dos filmes, foram extraídas e limpas informações textuais dos campos *genres*, *keywords*, *overview* e *title*, criando-se uma nova coluna *text* que serviu de base para os modelos baseados em conteúdo. Dessa forma, essa coluna é o resultado da concatenação de diferentes atributos dos filmes: título, gênero, palavras-chave e sinopse. O objetivo dessa combinação é fornecer uma descrição rica e informativa de cada item, utilizada posteriormente como entrada para os vetorizadores.

Para o modelo baseado em TF-IDF, foi realizado um processamento linguístico clássico com as seguintes etapas:

- Normalização e limpeza: remoção de pontuações e conversão para minúsculas.
- Tokenização: divisão do texto em palavras.
- Remoção de *stopwords*: exclusão de palavras frequentes sem valor semântico relevante (ex.: "the", "and", "in").
- Lematização: as palavras foram lematizadas de acordo com sua classe gramatical (substantivo, verbo, adjetivo etc.), utilizando o WordNetLemmatizer. Isso permite reduzir variações morfológicas a uma forma base comum (ex.: "running"$\rightarrow$ "run").

Esse pré-processamento do texto foi aplicada aos textos formados na coluna *text* de cada filme, resultando em documentos textuais otimizados para vetorização com o TF-IDF. Para o modelo DistilRoBERTa, não foi aplicado pré-processamento textual tradicional, uma vez que os modelos do tipo Transformer são treinados com tokenização própria e já são capazes de lidar com aspectos como pontuação e *stopwords*. Assim, a coluna *text* original foi utilizada como entrada para a tokenização do modelo.

3.4 Ambiente de desenvolvimento

Para a implementação dos sistemas de recomendação e a comparação entre as diferentes abordagens, utilizou-se o Google Colab, uma plataforma online que permite criar e executar código Python em notebooks interativos, com suporte gratuito a GPUs e TPUs e integração com o Google Drive, o que facilita o acesso e o armazenamento de dados (MINGRONI, 2024). O desenvolvimento envolveu bibliotecas voltadas à manipulação e ao processamento de dados, avaliação de desempenho dos modelos e visualização de resultados. Os experimentos foram realizados em um ambiente com GPU NVIDIA L4 (22,5 GB VRAM) — utilizada especificamente para os modelos AutoRec e DistilRoBERTa —, CPU Intel Xeon 2,20 GHz, 53 GB de memória RAM e 235,7 GB de armazenamento.

3.4.1 Bibliotecas e ferramentas utilizadas

As bibliotecas utilizadas foram escolhidas por sua eficiência e robustez para tarefas de recomendação, processamento textual e aprendizado profundo. A seguir, são listadas as principais bibliotecas, com suas respectivas versões:

- **pandas** (v2.2.2): usada para carregamento, manipulação e filtragem dos conjuntos de dados. Essencial para o pré-processamento dos dados tabulares e integração entre avaliações e metadados dos filmes.
- **numpy** (v1.26.4): utilizada para operações matemáticas e manipulação de arrays. Foi empregada no cálculo de métricas como NDCG, precisão e recall, além de suporte em operações vetoriais gerais.
- **scikit-surprise** (v1.1.4): biblioteca especializada em sistemas de recomendação baseados em filtragem colaborativa. Foi utilizada para a implementação do modelo *FunkSVD*.
- **scikit-learn** (v1.4.2): usada para cálculo de similaridade de cosseno, divisão de conjuntos de treino e teste (*train_test_split*) e em partes da avaliação dos modelos baseados em conteúdo.
- **sentence-transformers** (v2.7.0): biblioteca que fornece modelos pré-treinados para transformação de texto em embeddings semânticos. Utilizada com o modelo *paraphrase-distilroberta-base-v1* e no treinamento supervisionado com *Triplet Loss*.
- **torch** (PyTorch v2.3.0+cu121): backend de aprendizado profundo usado com *sentence-transformers*. Responsável pela manipulação de tensores, treinamento do modelo Distil-

RoBERTa e uso da GPU.

- **tensorflow** (v2.16.1) e **keras** (v3.3.3): usadas para a construção e treinamento do modelo *AutoRec*, baseado em autoencoders. Responsáveis por toda a arquitetura de rede neural e funções de otimização.
- **spacy** (v3.7.4): utilizada para lematização e pré-processamento textual dos dados de conteúdo (gêneros, palavras-chave e sinopse). O modelo `en_core_web_lg` foi empregado para representar semanticamente o texto.
- **psutil** (v5.9.8): usada para monitoramento do uso de memória RAM ao longo da execução dos modelos, especialmente durante o treinamento e inferência.
- **tqdm** (v4.66.4): adiciona barras de progresso a loops de execução, especialmente durante a geração dos trios de treinamento e na avaliação por usuário.
- **gc**, **os**, **time**, **random**, **ast**: bibliotecas da própria linguagem Python. Foram usadas para controle do sistema de arquivos, gerenciamento manual de memória, controle de tempo de execução e manipulação de estruturas de dados.

3.4.2 Código e reprodutibilidade

Para garantir a transparência e a reprodutibilidade dos experimentos, todo o código-fonte desenvolvido neste trabalho foi disponibilizado publicamente no GitHub: https://github.com/evecleison/recommender-systems-tcc. A divisão dos dados foi realizada de forma aleatória, utilizando 80% das interações para treino e 20% para teste, mantendo a proporcionalidade por usuário. O processo foi repetido cinco vezes com diferentes sementes para cálculo de médias e desvios padrão, assegurando robustez estatística.

3.5 Modelos implementados

Neste trabalho, foram implementados quatro modelos de sistemas de recomendação para comparar as diferentes abordagens. Dois deles utilizam métodos tradicionais, enquanto os outros dois usam técnicas de DL. Eles foram escolhidos de modo a representar a filtragem colaborativa e a filtragem baseada em conteúdo. Os modelos adotados foram: FunkSVD, TF-IDF, AutoRec e DistilRoBERTa. A configuração dos parâmetros e hiperparâmetros de cada um desses modelos é detalhada na seção seguinte, considerando as particularidades de cada abordagem e o objetivo de garantir uma comparação justa entre elas.

3.6 Ajuste de hiperparâmetros

O ajuste de hiperparâmetros foi conduzido com base em referências da literatura e em testes empíricos realizados durante os experimentos. A escolha de cada configuração visou balancear o desempenho preditivo dos modelos com a viabilidade computacional, considerando os recursos disponíveis no ambiente Colab. A seguir, são descritas as configurações adotadas para cada modelo avaliado:

O modelo FunkSVD foi implementado com a biblioteca `scikit-surprise`, utilizando os seguintes hiperparâmetros principais:

- Número de fatores latentes: 100
- Número de épocas: 20
- Taxa de aprendizado: 0,005

O modelo *TF-IDF*, embora não envolva treinamento supervisionado, requer a definição de parâmetros que afetam diretamente a representação vetorial dos itens. Utilizou-se o *TfidfVectorizer* com os seguintes ajustes:

- `ngram_range`: (1, 2)
- `min_df`: 3
- `max_df`: 0,8

O AutoRec foi configurado com uma arquitetura simples de autoencoder denso, com os seguintes hiperparâmetros:

- Dimensão da camada codificadora (*encoding_dim*): 500
- Regularização L2: 1×10^{-5}
- Taxa de dropout: 0,1
- Taxa de aprendizado: 0,001

No DistilRoBERTa, realizou-se o ajuste fino dos parâmetros utilizando a função de custo *Triplet Loss*, com os seguintes hiperparâmetros:

- Número de épocas: 10
- Tamanho do *batch*: 32
- Warmup steps: 100
- Taxa de aprendizado: 5×10^{-5}

Hiperparâmetros não mencionados explicitamente foram mantidos com os valores padrão das bibliotecas utilizadas.

3.7 Métricas de avaliação

Para avaliar o desempenho preditivo, foi usado um conjunto de métricas de avaliação amplamente aceitas e consolidadas: Precision@K, Recall@K e NDCG@K. Elas foram calculadas considerando o valor de $k = 10$, escolha justificada pelo comportamento típico dos usuários, que tendem a interagir com os itens nas primeiras posições da lista de recomendações, além de ser um padrão comum na literatura para facilitar comparações entre estudos.

Para o cálculo dessas métricas, considerou-se como acerto o caso em que o item recomendado ao usuário recebeu dele uma avaliação maior ou igual a 4, interpretando-se que o usuário gostou do filme. Caso contrário, a avaliação foi tratada como indiferente ou negativa.

Além das métricas preditivas, a eficiência computacional foi avaliada por meio do tempo de treinamento, do tempo de inferência e do consumo de memória, considerando o pico de uso da memória RAM.

4 RESULTADOS E DISCUSSÃO

Neste capítulo, são apresentados os resultados obtidos a partir da implementação e avaliação dos modelos de recomendação propostos: FunkSVD, TF-IDF, AutoRec e DistilRoBERTa. Para cada métrica, são apresentadas médias e desvios padrão calculados a partir de cinco execuções com diferentes divisões aleatórias dos dados, garantindo a reprodutibilidade.

4.1 Desempenho preditivo

As métricas utilizadas para avaliar a qualidade das recomendações foram Precision@10, Recall@10 e NDCG@10. A Tabela 1 apresenta os resultados obtidos.

Tabela 1 – Métricas de desempenho preditivo (média ± desvio padrão)

Modelo	**Precision@10**	**Recall@10**	**NDCG@10**
FunkSVD	0,7365 ± 0,0014	0,4871 ± 0,0008	0,7952 ± 0,0012
TF-IDF	0,4898 ± 0,0012	0,3064 ± 0,0008	0,7328 ± 0,0006
AutoRec	0,7235 ± 0,0026	0,4785 ± 0,0012	0,7841 ± 0,0023
DistilRoBERTa	0,6148 ± 0,0052	0,3948 ± 0,0039	0,8336 ± 0,0045

Fonte: Elaborado pelo autor.

Entre os modelos avaliados, o FunkSVD apresentou o melhor desempenho em termos de *precision* e *recall*, superando ligeiramente o AutoRec. No grupo dos modelos baseados em conteúdo, o destaque foi o DistilRoBERTa, que, embora não tenha superado os modelos colaborativos nas métricas tradicionais, obteve o maior valor de NDCG@10. Esse resultado indica uma capacidade superior de ordenar corretamente os itens mais relevantes no topo da lista de recomendações.

O modelo TF-IDF apresentou os piores resultados em todas as métricas preditivas refletindo as limitações de métodos baseados em conteúdo textual simples, especialmente quando comparados a representações semânticas mais sofisticadas como as geradas por *transformers*.

Os resultados demonstram que os modelos baseados em fatoração matricial continuam sendo eficazes. O FunkSVD, por exemplo, alcançou Precision@10 de 0,7365 e Recall@10 de 0,4871, com desempenho muito próximo ao do AutoRec, que também demonstrou alta capacidade preditiva. Ambos superaram os modelos baseados exclusivamente em conteúdo textual, confirmando a robustez das abordagens colaborativas, sobretudo em cenários com dados de interações ricas.

4.2 Eficiência computacional

A Tabela 2 apresenta uma comparação entre os modelos quanto ao tempo de treinamento, tempo médio de inferência por usuário e pico de consumo de memória RAM. Como o modelo baseado em TF-IDF não requer etapa de treinamento, ele foi omitido da tabela.

Tabela 2 – Métricas de eficiência computacional (média ± desvio padrão)

Modelo	**Treinamento (min)**	**Tempo de Inferência (ms)**	**Pico de RAM (GB)**
FunkSVD	0,74 ± 0,00	0,350 ± 0,009	2,93 ± 0,001
TF-IDF	–	1,117 ± 0,006	1,10 ± 0,013
AutoRec	0,99 ± 0,01	0,361 ± 0,050	5,90 ± 0,100
DistilRoBERTa	22,17 ± 0,21	4,097 ± 0,008	3,32 ± 0,063

Fonte: Elaborado pelo autor.

Do ponto de vista da eficiência computacional, o FunkSVD foi o modelo com menor tempo de treinamento e de inferência, tornando-se uma solução altamente eficiente para sistemas com restrições de recursos. Sua média de tempo de inferência foi inferior a 0,4 ms por usuário, destacando-o como uma excelente opção para aplicações em tempo real.

O modelo TF-IDF, embora inferior em desempenho preditivo, apresentou o menor consumo de memória RAM, com pico de aproximadamente 1,1 GB. Essa característica pode ser vantajosa em sistemas embarcados ou com hardware limitado. No entanto, seu tempo de inferência foi mais elevado que os modelos baseados em *deep learning*.

O DistilRoBERTa alcançou um excelente desempenho em ordenação (NDCG@10), mas apresentou o maior tempo de treinamento, com média superior a 22 minutos por execução, além de um tempo de inferência por usuário de aproximadamente 4 ms. Esse custo está associado à complexidade dos *embeddings* gerados por modelos baseados em *transformers*, além do uso de estratégias como *triplet loss*.

O AutoRec manteve um bom equilíbrio entre desempenho preditivo e custo computacional. Embora mais exigente que o FunkSVD, foi consideravelmente mais leve que o DistilRoBERTa em termos de tempo de treinamento e inferência, apresentando um bom compromisso entre performance e recursos consumidos. O modelo também teve o maior uso de memória RAM, com pico médio de quase 6 GB, devido a utilização intensiva de GPU e operações vetoriais.

4.3 Análise geral e considerações

Os resultados mostram que os modelos colaborativos ainda são mais eficazes em termos de *precision* e *recall* do que os modelos baseados em conteúdo, especialmente quando há dados suficientes de usuários e interações. Contudo, o uso de representações semânticas com o DistilRoBERTa permitiu alcançar uma ordenação mais precisa dos itens relevantes.

Observa-se também que modelos tradicionais, como o FunkSVD, podem alcançar desempenho comparável ou até superior ao de modelos de DL quando bem ajustados, especialmente em conjuntos de dados esparsos, pois modelos mais complexos geralmente requerem grandes volumes de dados para alcançar seu potencial máximo.

Modelos avançados como o DistilRoBERTa demonstram grande capacidade de gerar recomendações mais bem ordenadas, especialmente ao explorar o conteúdo textual. No entanto, seu uso demanda maior cautela, considerando fatores como tempo de treinamento, uso de memória e necessidade de infraestrutura computacional robusta. Em contrapartida, técnicas tradicionais como o FunkSVD continuam sendo altamente eficazes em cenários com restrições de recursos ou em aplicações que exigem respostas rápidas e em larga escala.

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, foi realizada uma análise comparativa entre modelos tradicionais e modelos baseados em DL para sistemas de recomendação, utilizando como referência um único conjunto de dados. Os modelos FunkSVD e TF-IDF representaram abordagens tradicionais, enquanto AutoRec e DistilRoBERTa foram utilizados como representantes de técnicas mais recentes baseadas em DL.

Os resultados obtidos demonstraram que modelos colaborativos, como o FunkSVD, ainda se destacam pela sua eficácia preditiva, simplicidade e alta eficiência computacional. Por outro lado, modelos baseados em representações semânticas, como o DistilRoBERTa, apresentaram desempenho preditivo e boa ordenação dos itens mais relevantes, embora com maior custo computacional. O AutoRec mostrou-se uma alternativa intermediária promissora, conciliando bom desempenho e custo aceitável.

Além disso, verificou-se há casos em que modelos tradicionais, como o FunkSVD, podem alcançar desempenho melhor do que modelos de DL se os hiperparâmetros forem bem escolhidos ou em cenários de dados esparsos.

Como trabalhos futuros, diversas frentes de investigação podem ser exploradas para ampliar e aprofundar esta análise. Entre elas, destaca-se a adoção de métricas complementares, como Cobertura, Diversidade e Novidade, que permitem avaliar aspectos qualitativos das recomendações além da precisão. Também é recomendável aplicar os modelos em diferentes conjuntos de dados – inclusive de domínios distintos – a fim de verificar sua generalização e robustez.

Adicionalmente, a análise de estratégias de filtragem híbrida, que combinam múltiplas fontes de informação, pode revelar ganhos significativos em desempenho. Outros métodos de recomendação podem ser analisados.

Questões práticas e relevantes como o problema de *cold start*, a escalabilidade dos modelos em ambientes reais, a atualização incremental de recomendações com novos dados (*incrementalidade*), a análise por segmentos de usuários e a interpretabilidade dos modelos são outras direções promissoras a serem exploradas em estudos posteriores.

Dessa forma, este trabalho contribui para a compreensão dos pontos fortes e limitações das abordagens analisadas, ao mesmo tempo em que estabelece uma base sólida para pesquisas futuras no campo dos sistemas de recomendação.

REFERÊNCIAS

AGGARWAL, C. C. **Recommender Systems: The Textbook**. New York: Springer, 2016.

BANIK, R. **The Movies Dataset**. 2017. Disponível em: https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset. Acesso em: 27 maio 2025.

CHINY, M.; CHIHAB, M.; BENCHAREF, O.; CHIHAB, Y. Netflix recommendation system based on tf-idf and cosine similarity algorithms. In: **Proceedings of the 2nd International Conference on Big Data, Modelling and Machine Learning (BML'21)**. Kenitra, Morocco: SciTePress, 2022. p. 15–20. Disponível em: https://www.researchgate.net/publication/360856267_Netflix_Recommendation_System_based_on_TF-IDF_and_Cosine_Similarity_Algorithms. Acesso em: 9 jun. 2025.

COUTINHO, J. G.; MARCACINI, R. M. **Um estudo sobre sistemas de recomendação baseados em Transformers**. 2022. Trabalho de Conclusão de Curso (MBA) – Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos. Disponível em: https://bdta.abcd.usp.br/item/003126073. Acesso em: 31 maio 2025.

DACREMA, M. F.; BOGLIO, S.; CREMONESI, P.; JANNACH, D. A troubling analysis of reproducibility and progress in recommender systems research. **ACM Transactions on Information Systems**, v. 39, p. 1–49, 1 2021. Disponível em: https://www.researchgate.net/publication/348283357_A_Troubling_Analysis_of_Reproducibility_and_Progress_in_Recommender_Systems_Research. Acesso em: 23 maio 2025.

DACREMA, M. F.; CREMONESI, P.; JANNACH, D. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In: ASSOCIATION FOR COMPUTING MACHINERY. **Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19)**. Copenhagen, Denmark: ACM, 2019. p. 101–109. Disponível em: https://www.researchgate.net/publication/348356197_Are_We_Really_Making_Much_Progress_A_Worrying_Analysis_of_Recent_Neural_Recommendation_Approaches. Acesso em: 23 maio 2025.

DEVLIN, J.; CHANG, M.-W.; LEE, K.; TOUTANOVA, K. Bert: Pre-training of deep bidirectional transformers for language understanding. In: **Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies**. Minneapolis, Minnesota: Association for Computational Linguistics, 2019. p. 4171–4186. Disponível em: https://arxiv.org/abs/1810.04805. Acesso em: 19 jun. 2025.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. **Deep Learning**. Cambridge, Massachusetts: MIT Press, 2016. Disponível em: http://www.deeplearningbook.org. Acesso em: 15 jun. 2025.

Hugging Face. **distilroberta-base**. 2019. Disponível em: https://huggingface.co/distilbert/distilroberta-base. Acesso em: 21 jul. 2025.

ISINKAYE, F.; FOLAJIMI, Y.; OJOKOH, B. Recommendation systems: Principles, methods and evaluation. **Egyptian Informatics Journal**, v. 16, n. 3, p. 261–273, 2015. ISSN 1110-8665. Disponível em: https://www.sciencedirect.com/science/article/pii/S1110866515000341. Acesso em: 1 jun. 2025.

JIN, Z.; YE, F.; NEDJAH, N.; ZHANG, X. A comparative study of various recommendation algorithms based on e-commerce big data. **Electronic Commerce Research and Applications**, v. 68, p. 101461, 2024. ISSN 1567-4223. Disponível em: https://www.sciencedirect.com/science/article/pii/S1567422324001066. Acesso em: 30 maio 2025.

JUNIOR, V. O. dos S.; BRANCO, J. A. C.; OLIVEIRA, M. A. de; SILVA, T. L. C. da; CRUZ, L. A.; MAGALHãES, R. P. **A Natural Language Understanding Model COVID-19 based for chatbots**. 2023. Disponível em: https://irislab.ce.gov.br/wp-content/uploads/2023/01/A_Natural_Language_Understanding_Model_COVID_19_based_for_chatbots__BIBE_.pdf. Accesso em: 9 jun. 2025.

KOREN, Y.; BELL, R.; VOLINSKY, C. Matrix factorization techniques for recommender systems. **Computer**, IEEE Computer Society, v. 42, n. 8, p. 30–37, 2009.

LI, C.; ISHAK, I.; IBRAHIM, H.; ZOLKEPLI, M.; SIDI, F.; LI, C. Deep learning-based recommendation system: Systematic review and classification. **IEEE Access**, v. 11, p. 113790–113835, 2023. Disponível em: https://ieeexplore.ieee.org/abstract/document/10274963. Acesso em: 15 jun. 2025.

LIU, Y.; OTT, M.; GOYAL, N.; DU, J.; JOSHI, M.; CHEN, D.; LEVY, O.; LEWIS, M.; ZETTLEMOYER, L.; STOYANOV, V. Roberta: A robustly optimized bert pretraining approach. **arXiv**, 2019. Disponível em: https://arxiv.org/abs/1907.11692. Acesso em: 20 jul. 2025.

MASSON, M. M. **Cold Start em recomendação de músicas utilizando Deep Learning**. 61 p. Dissertação (Dissertação (Mestrado em Informática) – Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2016. Disponível em: https://www2.dbd.puc-rio.br/pergamum/tesesabertas/1212372_2016_completo.pdf. Acesso em: 30 maio 2025.

MATOS, P. C. **Estudo comparativo de algoritmos de sistemas de recomendação de filmes**. 2021. 46 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2021. Disponível em: https://www.maxwell.vrac.pucrio.br/57546/57546.PDF. Acesso em: 30 maio 2025.

MINGRONI, G. S. **Sistema de recomendação de filmes baseado em filtragem**. 2024. 47 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Universidade Estadual Paulista, Bauru, 2024. Disponível em: https://repositorio.unesp.br/entities/publication/bcb85a6b-d10b-485a-9f1b-bf36396c2fc6. Acesso em: 28 maio 2025.

NAUMOV, M.; MUDIGERE, D.; SHI, H.-J. M.; HUANG, J.; SUNDARAMAN, N.; PARK, J.; WANG, X.; GUPTA, U.; WU, C.-J.; AZZOLINI, A. G.; DZHULGAKOV, D.; MALLEVICH, A.; CHERNIAVSKII, I.; LU, Y.; KRISHNAMOORTHI, R.; YU, A.; KONDRATENKO, V.; PEREIRA, S.; CHEN, X.; CHEN, W.; RAO, V.; JIA, B.; XIONG, L.; SMELYANSKIY, M. **Deep learning recommendation model for personalization and recommendation systems**. 2019. ArXiv preprint arXiv:1906.00091. Disponível em: https://arxiv.org/abs/1906.00091. Acesso em: 20 jun. 2025.

NGUYEN, G.; DLUGOLINSKY, S.; BOBAK, M.; TRAN, V.; GARCIA, A. L.; HEREDIA, I.; MALíK, P.; HLUCHý, L. Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey. **Artificial Intelligence Review**, v. 52, p. 77–124, 06 2019.

Disponível em: https://www.researchgate.net/publication/329990977_Machine_Learning_and_Deep_Learning_frameworks_and_libraries_for_large-scale_data_mining_a_survey. Acesso em: 14 jun. 2025.

OLIVEIRA, J. Sistemas de recomendação. In: MOTTA, C.; GARCIA, A. C.; VIVACQUA, A.; SANTORO, F.; OLIVEIRA, J. (Ed.). **Sistemas Colaborativos**. 1. ed. Rio de Janeiro: Elsevier Editora Ltda., 2012. cap. 15, p. 230–244. Disponível em: https://www.researchgate.net/publication/328228374_Sistemas_de_Recomendacao. Acesso em: 30 maio 2025.

PERMANA, A. H. J. P. J.; WIBOWO, A. T. Movie recommendation system based on synopsis using content-based filtering with tf-idf and cosine similarity. **International Journal on Information and Communication Technology (IJoICT)**, v. 9, n. 2, p. 1–14, Dez. 2023. Disponível em: https://socjs.telkomuniversity.ac.id/ojs/index.php/ijoict/article/view/747. Acesso em: 9 jun. 2025.

REATEGUI, E. B.; CAZELLA, S. C. Sistemas de recomendação. In: **Anais do 25º Congresso da Sociedade Brasileira de Computação: A universalidade da computação: um agente de inovação e conhecimento**. São Leopoldo: Unisinos, 2005. p. 306–348. Disponível em: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ff3756d0e7d480dd098b334df5006a740d11ce06. Acesso em: 1 jun. 2025.

RICCI, F.; ROKACH, L.; SHAPIRA, B. (Ed.). **Recommender Systems Handbook**. New York: Springer, 2011.

SANGEETHA, N.; THANGARAJ, H.; VASHISHT, V.; JOSHI, E.; VERMA, K.; KATARIYA, D. **A BERT Based Hybrid Recommendation System For Academic Collaboration**. 2025. Preprint. 2025. Disponível em: https://arxiv.org/abs/2502.15223. Acesso em: 9 jun. 2025.

SANH, V.; DEBUT, L.; CHAUMOND, J.; WOLF, T. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. **arXiv preprint arXiv:1910.01108**, 2020. Disponível em: https://arxiv.org/abs/1910.01108. Acesso em: 20 jul. 2025.

SANTANA, M. **Deep Learning para Sistemas de Recomendação (Parte 3) — Recomendação por Similaridade**. 2019. Disponível em: https://medium.com/data-hackers/deep-learning-para-sistemas-de-recomenda%C3%A7%C3%A3o-parte-3-recomenda%C3%A7%C3%A3o-por-similaridade-d788c126d808. Acesso em: 28 dez. 2024.

SCRIVANO, A. **A Comparative Study of Recommender Systems under Big Data Constraints**. 2025. Disponível: https://arxiv.org/abs/2504.08457. Acesso em: 22 maio 2025.

SEDHAIN, S.; MENON, A.; SANNER, S.; XIE, L. Autorec: Autoencoders meet collaborative filtering. In: **Proceedings of the 24th International Conference on World Wide Web**. [*S. l.*: *s. n.*], 2015. p. 111–112. Disponível em: https://www.researchgate.net/publication/311491420_AutoRec_Autoencoders_Meet_Collaborative_Filtering. Acesso em: 15 jun. 2025.

SOUSA, U. S. de. **Uma proposta de avaliação de embeddings de palavras por similaridade**. 2022. 45 p. TCC (Graduação em Sistemas e Mídias Digitais) - Universidade Federal do Ceará, Fortaleza, 2022. Disponível em: https://repositorio.ufc.br/handle/riufc/74839. Acesso em: 9 jun. 2025.

VASWANI, A.; SHAZEER, N.; PARMAR, N.; USZKOREIT, J.; JONES, L.; GOMEZ, A. N.; KAISER, L.; POLOSUKHIN, I. Attention is all you need. In: **Advances in Neural Information Processing Systems (NeurIPS 2017)**. Long Beach, CA, USA: Curran Associates, Inc., 2017. v. 30, p. 5998–6008. Disponível em: https://papers.nips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. Acesso em: 18 jun. 2025.

XIAOCHEN, Y.; QICHENG, L. Parallel algorithm of improved funksvd based on gpu. **IEEE Access**, v. 10, p. 26002–26010, 2022. Disponível em: https://ieeexplore.ieee.org/abstract/document/9729225. Acesso em: 5 jun. 2025.

ZHANG, A.; LIPTON, Z. C.; LI, M.; SMOLA, A. J. **Dive into Deep Learning**. E-book. Cambridge, UK: Cambridge University Press, 2023. Disponível em: https://d2l.ai. Acesso em: 19 maio 2025.

ZHANG, G.; LIU, Y.; JIN, X. A survey of autoencoder-based recommender systems. **Frontiers of Computer Science**, v. 14, n. 2, p. 430–450, 2020. Disponível em: https://doi.org/10.1007/s11704-018-8052-6. Acesso em: 22 maio 2025.

ZHANG, S.; YAO, L.; SUN, A.; TAY, Y. Deep learning based recommender system: A survey and new perspectives. **ACM Computing Surveys**, jul. 2017. Disponível em: https://www.researchgate.net/publication/318671349_Deep_Learning_Based_Recommender_System_A_Survey_and_New_Perspectives. Acesso em: 22 maio 2025.