**UNIVERSIDADE FEDERAL DO AGRESTE DE PERNAMBUCO**

**BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**IGOR MEDEIROS VANDERLEI**

**GENÁRIO CORREIA DE AZEVEDO**

**JOÃO VICTOR IANE GOIS PRESIDEU**

**SISTEMA DE RECOMENDAÇÃO DE FILMES DESENVOLVIDO COM GRAFOS**

**Garanhuns-PE**

**2023**

**IGOR MEDEIROS VANDERLEI**

**GENÁRIO CORREIA DE AZEVEDO**

**JOÃO VICTOR IANE PRESIDEU**

Trabalho desenvolvido na disciplina de Algoritmo e Estrutura de Dados II no curso de Bacharelado em Ciência da Computação, como requisito avaliativo da 2VA.

**Orientador:** Igor Medeiros Vanderlei

**Garanhuns– PE**

**2023**

**SISTEMA DE RECOMENDAÇÃO DE FILMES DESENVOLVIDO COM GRAFOS**

**RESUMO**

Este artigo apresenta um sistema personalizado de recomendação de filmes baseado em uma técnica de filtragem colaborativa e um grafo implementado em Java. O gráfico consiste em uma lista de usuários e uma lista de relações entre eles, sendo que cada usuário possui um nome, idade e uma lista de filmes que assistiu com suas respectivas pontuações. O sistema calcula a pontuação média de cada filme e retorna uma lista de recomendações com base nas pontuações mais altas. A técnica de filtragem colaborativa usada neste sistema identifica padrões de interesse entre os usuários e recomenda itens com base em suas opiniões e comportamentos anteriores, em vez de confiar apenas nas informações sobre os próprios filmes. O sistema também utiliza um grafo para localizar vizinhos de primeiro e segundo grau e sugerir filmes relacionados aos interesses do usuário, mesmo que nunca tenham assistido a esses filmes antes. O código é implementado em quatro classes, incluindo User, Relationship, Graph e Movie, sendo a classe App responsável pela execução do código.

**DESENVOLVIMENTO**

A recomendação de filmes personalizada é uma área em constante crescimento, especialmente com o aumento das plataformas de streaming e do acesso a uma grande variedade de filmes. Com um número elevado de opções disponíveis, torna-se difícil escolher um filme para assistir. É por isso que os sistemas de recomendação são cada vez mais importantes, pois eles ajudam os usuários a descobrirem novos filmes que possam ser do seu interesse.

O sistema de recomendação apresentado neste artigo é baseado em um grafo implementado em Java, utilizando a técnica de filtragem colaborativa. O grafo é composto por uma lista de usuários e uma lista de relações entre eles. Cada usuário tem um nome, idade e uma lista de filmes assistidos com suas respectivas pontuações. O grafo é utilizado para encontrar os vizinhos de primeiro e segundo grau a partir de um usuário, removendo aqueles que já assistiram aos mesmos filmes que o usuário atual. Em seguida, o código calcula a média das pontuações para cada filme e retorna uma lista de indicações com base nas pontuações mais altas.

Este sistema de recomendação é importante porque permite que os usuários descubram novos filmes que possam ser do seu interesse, intuídos pela proximidade e avaliações de usuários próximos. Além disso, a utilização de um grafo para encontrar vizinhos de primeiro e segundo grau permite que o sistema sugira filmes que estejam relacionados com os interesses do usuário, mesmo que eles ainda não tenham assistido a esses filmes antes.

Neste artigo, iremos explicar com mais detalhes como o sistema foi implementado e como ele pode ser utilizado.

Partindo da técnica de filtragem colaborativa usada em sistemas de recomendação para identificar padrões de interesse entre os usuários e recomendar itens (por exemplo, filmes, livros, músicas, produtos) com base em suas opiniões e comportamentos anteriores. Em vez de usar apenas informações sobre os filmes em si (como gênero, diretor, elenco, autor), a filtragem colaborativa aproveita a sabedoria coletiva dos usuários para determinar se determinado filme é provável que seja de interesse para um usuário com base nas opiniões de outros usuários com perfis semelhantes.

Neste sistema a filtragem se faz presente ao obter os vizinhos de primeiro e segundo grau do usuário e a remoção daqueles que já foram assistidos pelo usuário, calculando-se a média das pontuações para cada filme com base nas pontuações dadas pelos vizinhos restantes e retorna as indicações com base nestes valores. A ideia é que, se os vizinhos do usuário assistiram e gostaram de determinado filme que o usuário ainda não viu, é mais provável que o usuário também goste desse filme.

O código apresentado é uma implementação de um grafo, com o objetivo de obter sugestões de filmes para um usuário com base em seus vizinhos no grafo. O código está dividido em 4 classes, sendo, Usuario, Relacao, Grafo, Filme e uma classe App responsável pela execução do código.

A classe *“Usuario”* representa um usuário no sistema implementado, contendo informações básicas, como nome e idade, e uma lista de filmes assistidos por ele com as respectivas pontuações. A classe também possui métodos para adicionar filmes assistidos pelo usuário, obter a lista de vizinhos (outros usuários) e obter o caminho mais curto entre o usuário atual e outro usuário no grafo.

O método *obterCaminhoMaisCurto* utiliza o algoritmo de busca em largura (BFS - Breadth-First Search) para encontrar o caminho mais curto entre o usuário atual e o usuário de destino no grafo. A busca em largura (BFS) é um algoritmo de busca utilizado em grafos para visitar todos os vértices e arestas de um grafo de forma sistemática. A ideia por trás do algoritmo é visitar todos os vértices de um grafo a partir de um vértice inicial e expandir a busca de forma ordenada, ou seja, visitando todos os vértices vizinhos de um vértice antes de visitar os vértices adjacentes a estes. Esse método implementado no código, mantém um mapa de antecessores que registra o nó anterior que levou ao nó atual durante a busca, e outro mapa de distâncias que armazena a distância entre o usuário atual e cada um dos seus vizinhos, por fim, retorna uma lista de usuários representando o caminho mais curto entre o usuário atual e o usuário de destino.

O método getVizinhos retorna uma lista de usuários que estão conectados ao usuário atual no grafo. Ele itera sobre a lista de relações no grafo e adiciona os nós que são vizinhos do usuário atual na lista de vizinhos.

A classe *“Relacao”* define um objeto que representa uma relação entre dois usuários no grafo, contendo a origem, o destino e um peso associado a essa relação. Essa relação é definida por uma origem, um destino e um peso, que pode ser utilizado para representar a intensidade ou a distância da relação entre os dois usuários. Os atributos origem e destino representam os dois vértices que estão conectados por essa relação, enquanto o atributo peso representa o valor numérico associado a essa relação.

A classe *“Grafo”* representa o grafo de usuários e relações. Ela mantém uma lista de usuários e uma lista de relações entre os usuários. A classe também possui métodos para adicionar um novo usuário e uma nova relação, e para obter as recomendações de filmes para um determinado usuário.

O método *obterIndicacoes* da classe grafo retorna um mapa de filmes e suas respectivas pontuações médias, que são calculadas a partir das pontuações dadas pelos usuários que são vizinhos de primeiro e segundo grau do usuário fornecido. Primeiro, os vizinhos de primeiro e segundo grau são obtidos a partir do grafo. Em seguida, os vizinhos que já assistiram filmes que o usuário em questão também assistiu são removidos das listas de vizinhos. Em seguida, as pontuações dos filmes são calculadas para cada usuário vizinho que assistiu a um determinado filme, e uma lista dessas pontuações é mantida para cada filme.

O algoritmo começa adicionando o usuário atual a uma fila e inicializa uma lista de antecessores e uma lista de distâncias com o usuário atual. Em seguida, enquanto a fila não estiver vazia, o próximo usuário é removido da fila e os seus vizinhos são verificados. Se um vizinho ainda não tiver uma distância atribuída, a sua distância é definida como a distância atual + 1 e ele é adicionado à fila. O antecessor do vizinho é definido como o usuário atual. Quando todos os vizinhos foram visitados, a lista de antecessores contém o caminho mais curto do usuário atual ao usuário de destino. A lista de usuários que compõem o caminho mais curto é então construída a partir dessa lista de antecessores e retornada.

A classe comporta outros métodos, como o método *adicionarUsuario* que adiciona um usuário à lista de usuários do grafo; o método *adicionarRelacao* adiciona uma relação entre dois usuários, especificando a origem, o destino e o peso da relação; o método *getPesoRelacao* retorna o peso de uma relação entre dois usuários, após percorrer a lista de relações do grafo e verifica se existe uma relação entre os dois usuários especificados. Se existir, ele retorna o peso da relação. Caso contrário, ele retorna 1, e o método *getRelacoes* retorna a lista de relações do grafo. A classe também utiliza outras classes e interfaces Java, como *ArrayList, Comparator, HashMap, List, Map e PriorityQueue*, para implementar suas funcionalidades.

A classe *“Filme”* é uma classe em Java que representa um filme e possui várias propriedades como título, gênero, ano de lançamento, lista de atores e pontuação média. A classe possui um construtor que recebe como parâmetros todas essas propriedades para criar um objeto Filme. Além disso, a classe possui métodos get e set para cada uma dessas propriedades, permitindo a leitura e a atualização dos valores das mesmas.

A propriedade atores é uma lista de strings que contém o nome dos atores que participaram do filme. Essa propriedade é do tipo List<String>, que é uma interface Java que representa uma lista de elementos do tipo String. Isso significa que a lista de atores pode ser acessada e manipulada usando os métodos definidos na interface List, como add(), remove(), size(), etc.

A classe principal “*App*” é responsável por criar os filmes e usuários, adicioná-los ao grafo e definir os filmes assistidos por cada usuário. O programa faz as recomendações de filmes para cada usuário com base nas avaliações de outros usuários que assistiram a filmes semelhantes. Isso é feito calculando a similaridade entre os usuários com base em suas avaliações de filmes e recomendando os filmes mais bem avaliados pelos usuários com maior similaridade.