Inteligência Artificial

Sistema Inteligente

Trabalho Final

Jairo Kapta Berniger Litman

Marcelo Kenji Ichikawa

2025

TAREFA Nº 1

Sistema Inteligente com Deploy em Streamlit

Introdução

O objetivo principal deste trabalho é o desenvolvimento de um **sistema inteligente de recomendação de filmes**, integrando técnicas de processamento de linguagem natural e modelagem de preferências do usuário em uma aplicação interativa acessível via web.

O sistema oferece três modalidades complementares de recomendação:

1. **Recomendação por semelhança textual**: dado um filme selecionado, o sistema identifica títulos semanticamente próximos com base em gêneros, palavras-chave e sinopse.
2. **Recomendação por consulta livre**: o usuário fornece termos descritivos (ex: "viagem no tempo, distopia, resistência") e o sistema retorna filmes alinhados ao perfil temático desejado.
3. **Recomendação personalizada**: a partir do histórico de interações explícitas do usuário (marcações de "like" e "dislike"), o sistema constrói dinamicamente um perfil de preferência e ajusta as recomendações.

Além disso, a aplicação permite:

- Inclusão de novos filmes ao catálogo, com preenchimento de metadados essenciais (título, gêneros, palavras-chave e sinopse);
- Registro e remoção de avaliações em tempo real;

- Visualização de resultados com destaque para similaridade, gêneros e resumo dos filmes sugeridos.

Todo o sistema foi implementado em Python e disponibilizado por meio de uma interface web desenvolvida com **Streamlit**.

Dataset

O sistema foi desenvolvido com base no dataset *"Full TMDB Movies Dataset 2024 (1M Movies)"*, obtido da plataforma Kaggle [1]. Trata-se de um conjunto abrangente contendo metadados de aproximadamente mais de 1 milhão de filmes coletados da API do The Movie Database (TMDb), incluindo informações estruturadas e textuais relevantes para tarefas de recomendação.

Visando equilibrar diversidade, qualidade dos dados e desempenho computacional, realizamos um processo rigoroso de filtragem e pré-processamento:

1. **Filtragem por confiabilidade:** mantivemos apenas filmes com pelo menos 50 avaliações (`vote_count > 50`) e marcados como não adultos (`adult = False`), garantindo maior representatividade das notas e adequação geral ao público.
2. **Amostragem por popularidade:** ordenamos os filmes pela métrica `popularity` (indicador agregado de engajamento no TMDb) e selecionamos os 10.000 mais populares, priorizando conteúdos com maior reconhecimento e disponibilidade de metadados.
3. **Seleção de atributos relevantes:** foram mantidas apenas as colunas pertinentes à recomendação baseada em conteúdo.
4. **Tratamento de valores ausentes:** campos textuais (`genres`, `keywords`, `overview`) tiveram valores nulos substituídos por strings vazias, evitando falhas na vetorização textual.
5. **Ordenação e persistência:** o dataset final foi ordenado alfabeticamente por título e salvo como `movies_top10k.csv`.

Coluna	Descrição
`id`	Identificador único (TMDb)
`title`	Título original do filme
`genres`	Lista de gêneros (ex: "Action, Drama")
`keywords`	Palavras-chave descritivas (ex: "time travel, dystopia")
`overview`	Sinopse/resumo do enredo
`release_date`	Data de lançamento (formato YYYY-MM-DD)
`vote_average`	Média de avaliação (escala 0–10)
`vote_count`	Número de avaliações coletadas
`popularity`	Índice de popularidade normalizado (TMDb)
`runtime`	Duração em minutos
`poster_path`	Link para pôster

[1] https://www.kaggle.com/datasets/asaniczka/tmdb-movies-dataset-2023-930k-movies

Metodologia

A metodologia desenvolvida para este sistema de recomendação baseia-se na abordagem de *Content-Based Filtering* (Filtragem Baseada em Conteúdo). O sistema opera mediante a transformação de dados textuais não estruturados, especificamente sinopses e gêneros cinematográficos, em representações numéricas dentro de um espaço vetorial multidimensional.

A seguir, detalhamos os algoritmos implementados e a justificativa técnica para cada decisão arquitetural.

Vetorização e Representação Textual (TF-IDF)

O passo fundamental em Processamento de Linguagem Natural (NLP) consiste na conversão de texto subjetivo em vetores processáveis matematicamente. Para tal, empregamos o algoritmo **TF-IDF** (*Term Frequency-Inverse Document Frequency*).

Ao contrário de uma contagem simples de ocorrências (Bag of Words), o TF-IDF atribui um peso a cada termo, fundamentado na premissa de que a especificidade de uma palavra é inversamente proporcional à sua frequência global no conjunto de dados.

O cálculo pondera dois componentes estatísticos:

1. **TF (Term Frequency):** A frequência local do termo. Mede a relevância da palavra dentro do documento específico (sinopse) analisado.
2. **IDF (Inverse Document Frequency):** A raridade global do termo. Penaliza palavras que ocorrem em uma vasta quantidade de documentos no catálogo, pois estas oferecem baixo poder discriminativo para classificação.

Matematicamente, o peso w de um termo t em um documento d é dado por:

$$w_{t,d} = \mathrm{tf}(t,d) \times \log\left(\frac{N}{\mathrm{df}(t)}\right)$$

onde N é o total de filmes, $\mathrm{df}(t)$ é o número de documentos que contêm o termo e $\mathrm{tf}(t,d)$ é a frequência do termo t no documento d.

Análise de Caso

Para ilustrar a aplicação do algoritmo, considere a vetorização da sinopse do filme *"Alien: O Oitavo Passageiro"* em relação ao catálogo:

- **Artigos e Preposições (ex: "o", "de"):** Embora possuam alta frequência local (TF), aparecem em quase a totalidade dos documentos (IDF próximo de zero). O produto resultante tende a zero, permitindo que o modelo trate estes termos como ruído (*stop words*).
- **Termos Genéricos (ex: "filme", "cinema"):** Possuem frequência moderada, mas baixa capacidade de distinção temática. Recebem pesos baixos, influenciando minimamente o vetor final.
- **Termos Específicos (ex: "xenomorfo", "Nostromo"):** Apresentam alta frequência local neste documento específico e são raros no restante do catálogo (IDF alto). O resultado é um peso elevado, tornando estes termos as "assinaturas vetoriais" da obra.

Parametrização do Modelo

Para refinar a captura de nuances semânticas, o vetorizador foi configurado com os seguintes parâmetros:

- **N-grams (1, 2):** O modelo considera tanto unigramas (palavras isoladas) quanto bigramas (pares de palavras consecutivas). Isso permite diferenciar termos compostos, onde a união das palavras altera o sentido original (ex: "New York" é tratado como uma entidade geográfica única, distinta de "New" e "York" isoladamente).
- **Limitação de Dimensionalidade:** O espaço vetorial foi restringido às 5.000 *features* mais relevantes. Adicionalmente, aplicou-se um filtro de frequência mínima e máxima para eliminar termos excessivamente raros (possíveis erros de grafia) ou onipresentes (que não agregam valor semântico), otimizando o desempenho computacional e a precisão das recomendações.

Construção do Espaço Vetorial

A aplicação prática do algoritmo TF-IDF sobre o conjunto de filmes resulta na construção de uma **Matriz Documento-Termo**. Nesta estrutura, o vocabulário aprendido atua como as dimensões do espaço vetorial, enquanto cada filme é representado como um ponto neste hiperespaço.

O processo de conversão segue a seguinte lógica estrutural:

1. **Definição de Dimensões (Colunas):** O algoritmo varre todo o conjunto de dados e seleciona os 5.000 termos (unigramas ou bigramas) mais relevantes de acordo com os critérios de frequência estabelecidos. Cada um destes termos torna-se um eixo ortogonal no espaço vetorial.
2. **Mapeamento (Linhas):** Cada filme do catálogo ocupa uma linha na matriz. O valor da célula $M_{i,j}$ (filme i, termo j) corresponde ao peso w calculado pela fórmula TF-IDF.
3. **Esparsidade:** Dada a natureza da linguagem, um filme específico contém apenas uma pequena fração do vocabulário total de 5.000 palavras. Consequentemente, a vasta maioria das células da matriz é preenchida com zeros. Matematicamente, o sistema lida com uma **Matriz Esparsa**, o que otimiza drasticamente o armazenamento em memória e a velocidade das operações de álgebra linear.

Dessa forma, transformamos descrições semânticas subjetivas em vetores numéricos fixos $\boldsymbol{v} \in \mathbb{R}^{5000}$. Por exemplo, se a dimensão 42 corresponde à palavra "alien" e a dimensão 105 à palavra "espaço", o vetor do filme *Alien* terá valores positivos significativos nestas coordenadas, enquanto o vetor de um romance de época terá valor zero nas mesmas, posicionando-os em regiões opostas do espaço geométrico.

Cálculo de Similaridade (Cosseno)

Com os filmes representados em um espaço de 5.000 dimensões, a proximidade semântica entre eles é calculada através da **Similaridade de Cosseno**.

A escolha desta métrica em detrimento da Distância Euclidiana justifica-se pela natureza dos dados textuais. Enquanto a Distância Euclidiana mede a magnitude da diferença (o que penalizaria sinopses mais longas ou detalhadas), a Similaridade de Cosseno mede a **orientação** dos vetores.

Em análise de texto, o tamanho do documento não deve alterar sua classificação temática. Se dois filmes tratam de "Viagem Espacial", seus vetores apontarão para a mesma direção no hiperespaço, independentemente de uma sinopse ter 20 palavras e a outra 200.

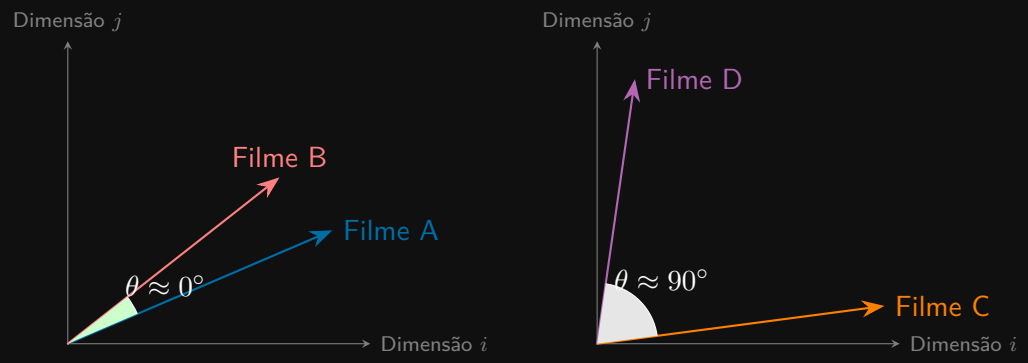
A métrica é obtida pelo produto escalar normalizado dos vetores A e B:

$$\text{sim}(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Interpretação dos Resultados

O resultado $\cos(\theta)$ varia entre 0 e 1 no contexto de TF-IDF (onde os pesos não são negativos):

- **Cosseno $\approx$ 1 (Vetores Colineares):** Indica alta similaridade semântica. *Exemplo: "Toy Story"* e *"Toy Story 2"* compartilham vocabulário central significativo, resultando em um ângulo próximo de zero.
- **Cosseno $\approx$ 0 (Vetores Ortogonais):** Indica ausência de correlação semântica. *Exemplo: "O Poderoso Chefão"* e *"Barbie"*. Como não compartilham termos relevantes, seus vetores são perpendiculares no espaço (90º), resultando em similaridade nula.
- **Valores Intermediários:** Representam interseções parciais de temas. *Exemplo:* Filmes de guerra espacial e filmes de guerra histórica podem apresentar similaridade mediana devido ao compartilhamento de termos bélicos, apesar da divergência no cenário (espaço vs. terra).



Alta Similaridade
$\cos(\theta) \approx 1$
(Ex: Toy Story 1 & 2)

Baixa Similaridade
$\cos(\theta) \approx 0$
(Ex: Poderoso Chefão & Barbie)

Perfil do Usuário (Adaptação de Rocchio)

Para viabilizar a personalização dinâmica, implementamos uma adaptação do **Algoritmo de Rocchio**. Originalmente concebido para sistemas de recuperação de informação, este método utiliza o conceito de *Feedback de Relevância* para refinar consultas.

No contexto deste projeto, o algoritmo constrói um perfil de usuário calculando um **vetor centróide**. Geometricamente, o vetor do usuário move-se pelo espaço vetorial aproximando-se das regiões temáticas dos itens avaliados positivamente e distanciando-se dos itens avaliados negativamente.

Formalização Matemática

Diferente da implementação clássica que modifica uma *query* inicial, nosso sistema constrói o vetor do usuário ($\boldsymbol{v}_{user}$) exclusivamente a partir do histórico de interações, conforme a equação:

$$\boldsymbol{v}_{user} = \alpha \cdot \left(\frac{1}{|L|} \sum_{i \in L} \boldsymbol{v}_i \right) - \beta \cdot \left(\frac{1}{|D|} \sum_{j \in D} \boldsymbol{v}_j \right)$$

Onde:

- L e D representam os conjuntos de filmes com *Likes* e *Dislikes*, respectivamente.
- $\boldsymbol{v}_i$ e $\boldsymbol{v}_j$ são os vetores TF-IDF dos filmes nesses conjuntos.
- A divisão por $|L|$ e $|D|$ normaliza os vetores, garantindo que o perfil represente a preferência média e não seja distorcido pela quantidade absoluta de interações.

Definição de Hiperparâmetros

Os coeficientes α e β controlam a influência do feedback positivo e negativo. Definimos:

- $\alpha = 1.0$ (Peso do Feedback Positivo)
- $\beta = 0.5$ (Peso do Feedback Negativo)

A decisão de manter $\beta < \alpha$ baseia-se na heurística de recomendação onde a indicação positiva é um sinal mais forte de preferência do que a negativa. Um peso negativo excessivo poderia restringir severamente o espaço de busca, eliminando gêneros inteiros devido à rejeição de um único título específico. Com essa configuração, o sistema prioriza a aproximação dos interesses do usuário, utilizando as rejeições apenas para refinar a precisão, sem causar exclusões excessivas.

Implementação e Aplicação do Perfil do Usuário

A operacionalização do modelo de Rocchio no sistema ocorre em duas etapas distintas: a construção do vetor sintético do usuário e a aplicação deste vetor nas funções de recuperação de informação.

Construção do Vetor de Preferência

O sistema processa o histórico de interações do usuário não como uma lista estática, mas como clusters de vetores no hiperespaço. O procedimento algorítmico segue uma sequência lógica de agregação e subtração vetorial:

1. **Segregação de Clusters:** O sistema itera sobre o dicionário de avaliações, separando os índices dos filmes em dois subconjuntos distintos: o conjunto de aprovação (L) e o conjunto de rejeição (D).
2. **Cálculo dos Centróides:** Para cada subconjunto, calcula-se a média aritmética de todos os vetores contidos nele. Isso resulta em dois vetores representativos:
    - **Centróide Positivo:** Representa o "filme médio ideal" baseando-se em tudo que o usuário gostou.
    - **Centróide Negativo:** Representa a média das características que o usuário tende a rejeitar.
3. **Síntese do Vetor do Usuário:** Aplica-se a equação de Rocchio (discutida na seção anterior) para fundir estes dois centróides. O vetor resultante aponta para uma região do espaço vetorial que maximiza a proximidade com os gostos do usuário enquanto aplica uma penalidade vetorial na direção dos desgostos.

É importante notar que o sistema trata a ausência de dados (start a frio) de forma robusta: se não houver rejeições, o vetor é formado apenas pela média dos likes; se não houver interação alguma, o perfil é considerado nulo e a personalização é desativada.

Estratégias de Recomendação

Uma vez calculado o vetor do usuário, o sistema o utiliza de duas maneiras distintas para gerar sugestões:

1. Recomendação Personalizada Pura Neste modo, o sistema ignora qualquer termo de busca externo. O vetor do usuário é tratado como se fosse o vetor de um filme ("query vector"). O algoritmo calcula a Similaridade de Cosseno entre este vetor de perfil e todos os outros 10.000 filmes do banco de dados.

O resultado é uma lista classificada dos filmes que estão geometricamente mais próximos do gosto consolidado do usuário. Por fim, aplica-se um filtro de exclusão para garantir que obras já avaliada ou assistidas não sejam recomendadas novamente, focando na descoberta de novos conteúdos.

2. Busca Híbrida Ponderada (Interpolada) Esta é uma aplicação sofisticada onde o perfil do usuário atua como um viés (*bias*) sobre uma busca textual ou por similaridade de item.

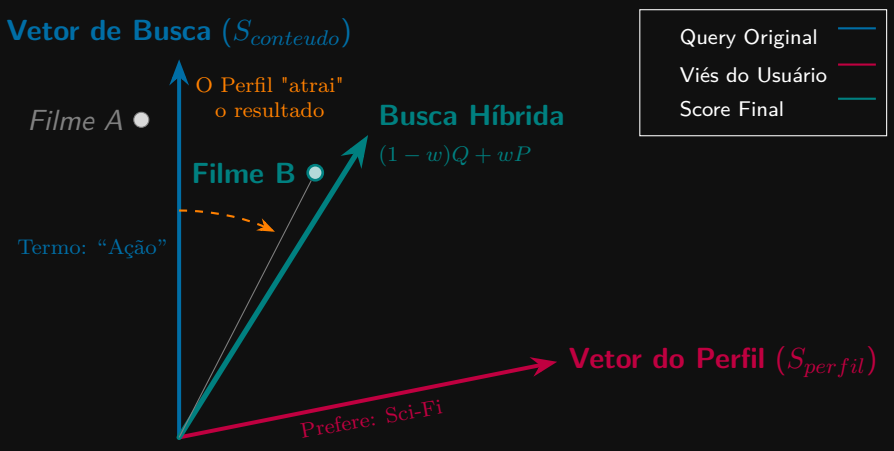
Quando o usuário busca por um filme específico ou por palavras-chave, o sistema calcula dois scores de similaridade independentes:

- $S_{conteudo}$: A relevância do filme em relação ao termo pesquisado.
- S_{perfil}: A afinidade do filme com o histórico do usuário.

O score final é obtido através de uma interpolação linear controlada por um parâmetro de peso (w):

$$Score_{final} = (1 - w) \cdot S_{conteudo} + w \cdot S_{perfil}$$

Esta abordagem permite que o sistema desempate resultados. Se o usuário buscar por "Ação", o sistema retornará filmes de ação, mas priorizará aqueles que também alinham com o perfil do usuário (ex: se o usuário gosta de Ficção Científica, o sistema priorizará "Ação Sci-Fi" sobre "Ação Comédia"), oferecendo uma experiência de busca contextualmente personalizada.



Implementação

A arquitetura do sistema foi desenvolvida em Python, estruturada em torno de uma classe central denominada `MovieRecommender`. Esta classe atua como o orquestrador principal, gerenciando desde a ingestão de dados e treinamento do modelo vetorial até a lógica de inferência para as recomendações.

A implementação apoia-se nas bibliotecas *pandas* para manipulação tabular de dados e *scikit-learn* para a álgebra linear e cálculos de métricas de distância. Abaixo, descrevemos os módulos funcionais do sistema.

Pipeline de Dados e Persistência

O sistema adota uma estratégia híbrida de persistência de dados para garantir flexibilidade e manutenibilidade:

1. **Dados Estáticos (CSV):** O catálogo base de filmes é carregado de um arquivo CSV imutável. Este dataset contém as metadados essenciais (título, gêneros, sinopse).
2. **Dados Dinâmicos (JSON):** Para permitir a personalização e a evolução do sistema, as interações do usuário (avaliações de "Like/Dislike") e novos filmes adicionados manualmente são armazenados em arquivos JSON locais.

Durante a inicialização (`__init__`), o sistema realiza a fusão desses dados. O *dataframe* principal é construído concatenando o catálogo base com os filmes inseridos pelo usuário, garantindo que o modelo considere ambas as fontes de informação de forma transparente.

Engenharia de Features e Treinamento

O processo de treinamento do modelo, encapsulado no método `fit` ocorre através da construção de uma matriz TF-IDF.

A etapa crítica aqui é a **Concatenação de Features**. O algoritmo não analisa apenas a sinopse. Ele cria um campo sintético denominado `combined_features`, que agrega:

- A lista de gêneros;
- As palavras-chave (*keywords*);
- A sinopse completa (*overview*).

Essa fusão textual enriquece o contexto. Por exemplo, se uma sinopse não menciona explicitamente a palavra "Futuro", mas o gênero é "Sci-Fi", a concatenação garante que o vetor resultante capture essa dimensão semântica.

Motores de Recomendação

A implementação oferece três modalidades distintas de recomendação, desenhadas para cobrir diferentes intenções de busca do usuário:

1. Recomendação Contextual (*Item-Item*)

Implementada no método `recommend_by_movie`, esta função recebe um filme de referência e busca os vizinhos mais próximos no espaço vetorial.

- **Lógica:** Calcula o cosseno entre o vetor do filme de entrada e todos os outros vetores da matriz.
- **Hibridização:** O sistema permite uma ponderação através do parâmetro `profile_weight`. Isso possibilita misturar a similaridade do filme com o gosto pessoal do usuário. Se o peso for maior que zero, o *score* final é uma média ponderada entre "o que se parece com este filme" e "o que se parece com o perfil do usuário".

2. Recomendação por Palavras-Chave (*Query-Item*)

No método `recommend_by_keywords`, o sistema vetoriza uma consulta textual arbitrária (ex: "viagem no tempo, robôs") usando o mesmo vocabulário aprendido durante o treinamento. O vetor resultante é comparado contra a base de dados, permitindo buscas semânticas que vão além da correspondência exata de palavras.

3. Recomendação Personalizada (*User-Item*)

O método `recommend_personal` ignora o contexto atual e foca exclusivamente no histórico do usuário. Ele gera o vetor de perfil (baseado na média de likes e dislikes explicada na Metodologia) e retorna os itens com maior alinhamento vetorial global, excluindo automaticamente títulos que o usuário já assistiu ou avaliou.

Gerenciamento de Estado e Re-Treinamento

Um diferencial da implementação é a capacidade de atualização em tempo de execução. Métodos como `add_new_movie` ou `save_rating` não apenas alteram os arquivos JSON, mas também acionam gatilhos de atualização.

Quando um novo filme é inserido, o sistema reconstrói o *dataframe* e executa novamente o método `fit`. Isso garante que o novo título seja imediatamente vetorizado e passe a ser recomendável nas próximas consultas, sem a necessidade de reiniciar a aplicação. Da mesma forma, novas avaliações recaluculam instantaneamente o vetor do usuário, tornando a personalização responsiva.

Análise de Desempenho e Métricas

A avaliação de sistemas de recomendação difere da avaliação de modelos preditivos tradicionais (como classificação ou regressão), pois a "correção" de uma sugestão possui um componente subjetivo inerente. Para validar a solução proposta, consideramos duas dimensões principais: a qualidade da recuperação da informação e a eficiência computacional.

Métricas de Qualidade e Análise de Casos

Dada a natureza não supervisionada da filtragem baseada em conteúdo, a validação do sistema não ocorre por uma matriz de confusão binária simples, mas sim pela análise da relevância subjetiva

das recomendações. Para quantificar esse desempenho, adotamos a métrica de **Precisão no Top-k (P@k)**.

Esta métrica avalia a proporção de itens relevantes dentro das k primeiras sugestões oferecidas pelo algoritmo, conforme a equação:

$$P@k = \frac{\text{Itens Relevantes no Top } k}{k}$$

Abaixo, analisamos o comportamento do algoritmo em dois cenários de teste distintos ($k = 5$), baseados em execuções reais do sistema.

Cenário 1: Consistência Temática (Caso "Apocalypse Now")

Ao solicitar recomendações para o clássico de guerra *Apocalypse Now*, o sistema retornou os seguintes resultados no Top 5:

1. *The Green Berets* (Similaridade: 41%)
2. *Full Metal Jacket* (Similaridade: 40%)
3. *Casualties of War* (Similaridade: 37%)
4. *Uncommon Valor* (Similaridade: 37%)
5. *Missing in Action* (Similaridade: 36%)

Análise: O algoritmo demonstrou robustez excepcional neste cenário. Todos os cinco filmes recomendados pertencem estritamente ao gênero de Guerra, com forte correlação ao subgênero da Guerra do Vietnã. A sobreposição de vocabulário bélico (termos como "soldier", "war", "vietnam", "mission") garantiu uma clusterização perfeita.

- **Resultado:** $P@5 = 1.0$ (100% de precisão).

Cenário 2: Ruído Semântico (Caso "Toy Story")

Para a animação infantil *Toy Story*, o sistema apresentou o seguinte comportamento:

1. *Toy Story 3* (Similaridade: 47%)
2. *Toy Story 4* (Similaridade: 45%)
3. *Toy Story 2* (Similaridade: 43%)
4. *Small Soldiers* (Similaridade: 35%)
5. *The 40 Year Old Virgin* (Similaridade: 34%)

Análise: As três primeiras posições foram ocupadas pelas sequências da franquia, o que é o comportamento ideal (máxima relevância). O quarto item, *Small Soldiers*, mantém a coerência temática (brinquedos que ganham vida).

Contudo, o quinto item, *The 40 Year Old Virgin*, representa um **Falso Positivo**. Embora seja uma comédia adulta, é possível que a sinopse do filme, que contém palavras-chave como "Andy", "enganou" o algoritmo TF-IDF. O modelo matemático pode ter identificado a coincidência léxica da palavra "Andy", a criança de *Toy Story* chama-se Andy igual ao protagonista de *The 40 Year Old Virgin*, mas falhou em capturar a distinção de público-alvo (Infantil vs. Adulto).

- **Resultado:** $P@5 = 0.8$ (80% de precisão).

Conclusão das Métricas

Os testes indicam que o sistema possui alta precisão para nichos com vocabulário denso e específico (como Guerra ou Sci-Fi). Entretanto, observa-se uma ligeira degradação de desempenho quando termos polissêmicos ou objetos (como "brinquedos") aparecem em contextos narrativos divergentes, evidenciando as limitações naturais de uma abordagem puramente baseada em frequência de termos (TF-IDF) sem análise de sentimento ou contexto profundo.

Limitações e Possibilidades de Evolução

Limitações do Modelo Atual

1. O Problema do Início Frio (Cold Start)

Identificamos duas vertentes deste problema clássico:

- **Novo Usuário:** Sem histórico de avaliações, o vetor do perfil do usuário ($\boldsymbol{v}_{user}$) é nulo. O sistema não consegue personalizar recomendações até que ocorra a primeira interação.
- **Novo Item:** Embora o sistema consiga recomendar um filme novo imediatamente após sua inserção (vantagem sobre a Filtragem Colaborativa), ele depende inteiramente da qualidade da sinopse inserida. Uma descrição pobre ou vaga resulta em um vetor fraco, tornando o filme "invisível" nas recomendações.

2. Limitação Semântica do TF-IDF

O algoritmo baseia-se na *coincidência léxica*, não no significado semântico profundo. *Exemplo:* Se um filme contém a palavra "automóvel" e outro a palavra "carro", o TF-IDF pode não capturar a similaridade, a menos que ambas as palavras ocorram nos documentos. O sistema não "entende" que são sinônimos, apenas compara strings.

Possibilidades de Evolução

Para mitigar as limitações supracitadas, propomos as seguintes evoluções para iterações futuras do projeto:

1. **Word Embeddings:** Substituir o TF-IDF por modelos de linguagem baseados em Redes Neurais (Transformers). Isso permitiria capturar relações semânticas complexas. O sistema entenderia que "Rei" e "Rainha" estão matematicamente próximos, resolvendo a limitação da coincidência léxica.
2. **Sistema Híbrido:** Implementar uma camada de *Collaborative Filtering* (Filtragem Colaborativa). Ao cruzar dados de múltiplos usuários, o sistema poderia recomendar itens baseado no padrão de consumo de pessoas parecidas, introduzindo diversidade e resolvendo o problema da superespecialização.

3. **Redução de Dimensionalidade:** Aplicar técnicas de decomposição de matrizes para reduzir o espaço vetorial de 5.000 para, por exemplo, 100 componentes principais latentes. Isso aumentaria a velocidade de processamento e ajudaria a agrupar conceitos similares, melhorando a generalização do modelo.