

DATA ANALYTICS

DADOS GERADOS POR HUMANOS

AULA 05

SUMÁRIO

O QUE VEM POR AÍ?	3
HANDS ON	4
SAIBA MAIS.....	5
O QUE VOCÊ VIU NESTA AULA?	20
REFERÊNCIAS.....	21

EMSE

O QUE VEM POR AÍ?

Nesta aula, aplicaremos os conhecimentos obtidos ao longo da disciplina para construir a base de um sistema de recomendação. Veremos desde sistemas simples até os mais complexos, usando técnicas de vetorização de textos que vimos em aulas anteriores para extrair informações de sinopses de filmes e fazer recomendações baseadas nelas.



HANDS ON

Seguindo o mesmo modelo da aula passada, mesclaremos conteúdo teórico e prático, usando o Jupyter notebook para desenvolver nossos códigos.

EMEND

SAIBA MAIS

Introdução

Usar sistemas de recomendação e métodos de Machine Learning em produção não é somente sobre implementar o melhor algoritmo, e sim entender seus usuários e seu domínio.

A definição de sistemas de recomendação é simples e vamos fornecê-la futuramente, mas, por enquanto, focaremos em quais problemas ela se propõe a resolver e como é utilizada. Nessa aula, vamos:

- Entender a tarefa que um sistema de recomendação está tentando emular;
- Desenvolver um entendimento do que são recomendações personalizadas e não-personalizadas;
- Desenvolver uma taxonomia de como descrever sistemas de recomendação.

Antes de nos aprofundarmos no tema, é importante entendermos a diferença entre recomendação e coisas que são parecidas com recomendação. Veja os exemplos e tente defini-los:

- Numa loja de música, pego uma pilha de CDs e, usando-os como contexto, inicio uma conversa com a pessoa atendente, que me indica outros discos que podem me agradar;
- Geralmente, recebo e-mails de supermercados com ofertas de produtos que eles dizem ser uma boa promoção;
- Na televisão, muito se pensa ao colocar comerciais com o conteúdo certo, com o objetivo de atingir públicos específicos;
- Uma vez por semana, recebo um newsletter com os filmes mais assistidos no cinema.

Para começar, por que a recomendação é importante? Para entendermos essa ideia, usaremos o conceito de long tail, cunhado por Chris Anderson em 2004. No

artigo, originalmente publicado na revista Wired, o autor identificou um novo modelo de negócios que é frequentemente visto na internet.

Em uma loja virtual, é possível armazenar um número infinito de produtos, visto que o aluguel de espaço é barato ou, se é vendido conteúdo digital, espaço não é um problema, o que leva seu custo a quase zero. O que é diferente de uma loja física, que é limitada geograficamente.

A ideia por trás da economia long tail é a possibilidade lucrar vendendo muitos produtos, mas poucos de cada, para muitas pessoas diferentes.

A grande questão que permanece é: como os usuários encontram aquilo que eles precisam?

Aqui entram os sistemas de recomendação, visto que eles ajudam pessoas a encontrar diversas coisas que, de outra maneira, talvez jamais imaginassem que existissem.

Depois dessas informações, podemos definir sistemas de recomendação da seguinte maneira: “Um sistema de recomendação calcula e provê conteúdo relevante ao usuário baseado no conhecimento do usuário, conteúdo e interações entre o usuário e o item.”

Tomando a Netflix como exemplo, definiremos alguns termos comuns ao tratarmos de sistemas de recomendação.

Termo	Definição	Exemplo Netflix
Predição	Uma estimativa de quanto um usuário irá gostar de um item.	A Netflix adivinha como você avaliará um item.
Relevância	Itens ordenados de acordo com o que é mais relevante para o usuário nesse momento. Relevância é uma função de contexto, demografia e ratings.	Ordena as linhas dos títulos de acordo com a aplicabilidade.

Recomendação	Os top N itens mais relevantes.	Top itens para você.
Personalização	Integra a relevância na apresentação.	As linhas dedicadas a você.

Entretanto, veremos que construir um sistema de recomendação que forneça recomendações adequadas não é simples. Observe a figura 1.

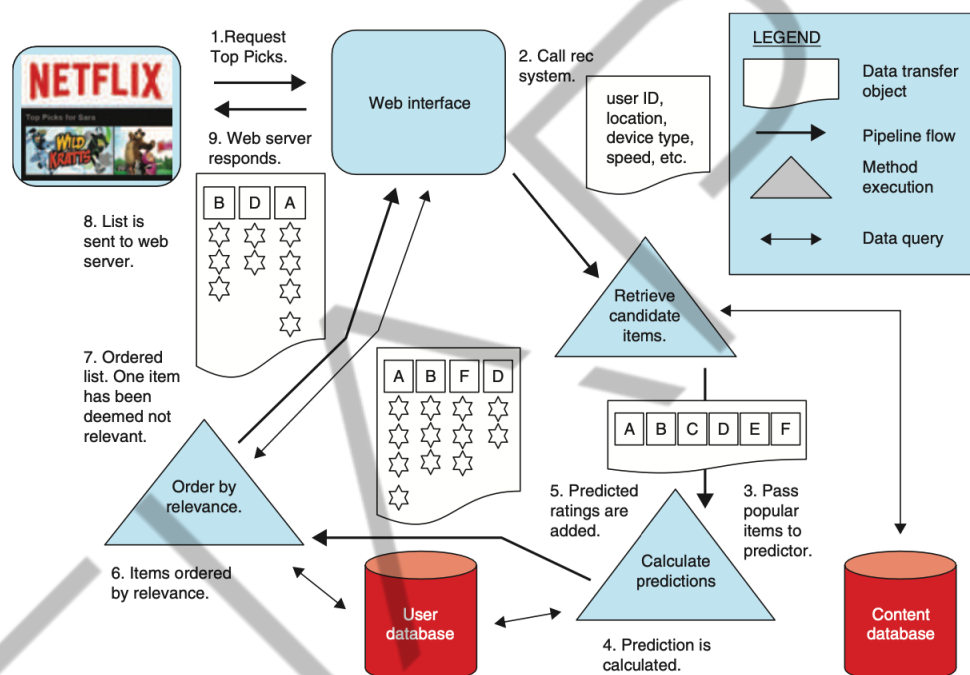


Figura 1 - Como as principais escolhas da Netflix podem ser selecionadas

Fonte: Practical Recommender Systems (2019)

Vamos entendê-la passo a passo:

1. Uma requisição para uma lista das principais escolhas é recebida.
2. O sistema de recomendação é chamado, retornando itens da base de dados que são mais similares ao gosto do usuário atual.
3. Os tops 5 itens (pode ser muito mais, na prática) são postos para o próximo passo, que é fazer a predição.
4. A predição é feita usando as preferências do usuário obtidas da base de dados de usuários. É provável que alguns itens fiquem de fora devido à baixa correlação com o usuário.

5. Os itens significantes são resultado da predição calculada, agora com o rating predito adicionado a eles.
6. Os itens significantes são ordenados de acordo com a preferência do usuário, contexto e dados demográficos.
7. Os itens são ordenados por relevância.
8. O pipeline retorna os itens.
9. O servidor apresenta os resultados.

A imagem anterior evidencia que existem muitos aspectos a serem considerados quando lidamos com sistemas de recomendação. Além disso, esse pipeline não apresenta partes como coleta de dados e a construção de modelos.

Em nossa aula, o foco é na construção e entendimentos dos modelos de recomendação.

De maneira geral, os algoritmos de recomendação pertencem a dois grupos distintos. Os algoritmos que empregam o uso de dados são chamados “filtros colaborativos”. Algoritmos que usam conteúdo de metadados e perfis de usuários para calcular recomendações são chamados “filtros baseados em conteúdos”. Ainda existe um terceiro grupo, chamado de sistemas híbridos, que é uma combinação dos dois anteriores.

Na prática, precisamos ter uma ideia do que queremos fazer com um sistema de recomendação. Podemos ter o seguinte ciclo de modelagem até ter um sistema em produção:

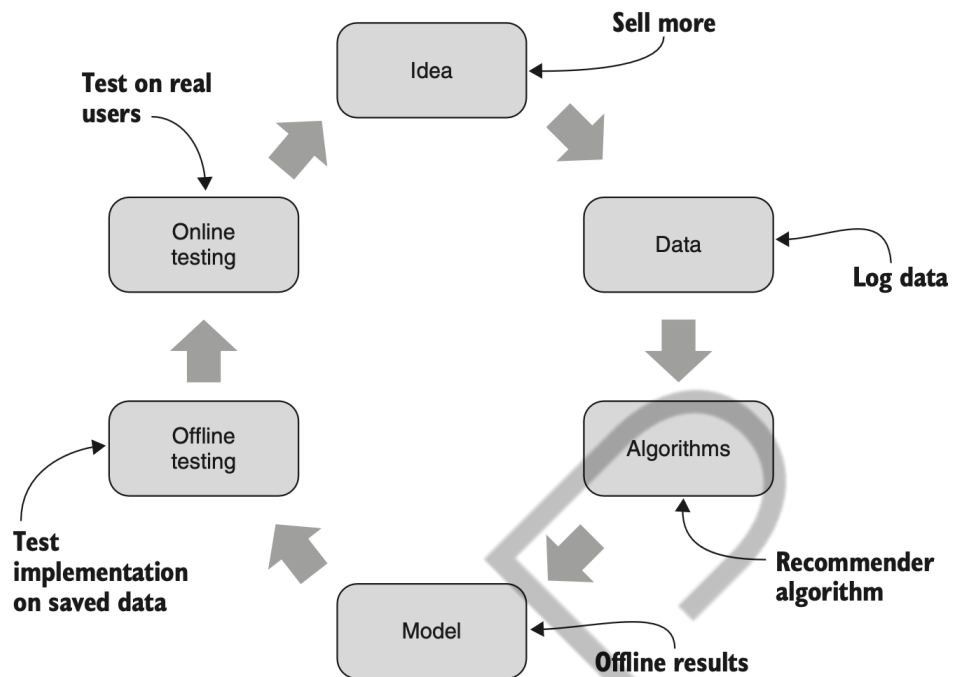


Figura 2 - Ciclo de modelagem de sistemas de recomendação

Fonte: Practical Recommender Systems (2019)

Novos usuários e Produtos

Percebemos que, para fazer recomendações, é preciso ter muitas informações sobre os gostos do usuário e também sobre os produtos que vamos recomendar, afinal, é preciso saber quais produtos são similares ao seu gosto. Entretanto, mesmo tendo muitos dados, não é possível apenas com isso resolver o problema de como introduzir sugestões, sejam elas usuários ou produtos. Esse problema é denominado cold start.

Para novos produtos, a solução é relativamente simples. Geralmente, a introdução de novos itens é acompanhada da promoção deles, por exemplo, enviando e-mails aos usuários ou criando uma área especial onde novos itens podem ser consultados.

Para novos usuários, a solução é mais complicada. Quando devemos começar a oferecer recomendações personalizadas? Qual a quantidade mínima de dados suficiente para determinar as preferências de uma pessoa? Quando um novo usuário chega, não sabemos nada. Sem dados, sem personalização.

Desta forma, precisamos de maneiras para lidar com esse problema. Para itens novos, usar filtros baseado em conteúdo pode ser uma solução. Entretanto, para

usuários novos, temos outras abordagens. A solução é encontrar informações mesmo em um conjunto de dados esparsos. Podemos usar a informação que já temos e relacionar isso com o novo usuário.

Podemos, então, usar regras de associação, bem como criar segmentos para usuários existentes e considerar o quão rápido podemos adequar um novo usuário a um segmento.

Regras de associação dizem que, se um usuário colocou pão no seu carrinho de compras, oferecer manteiga a ele pode ser uma boa recomendação. Para um usuário novo, podemos usar regras de associação como, por exemplo, por onde o usuário navegou. Podemos implementar isso da seguinte forma:

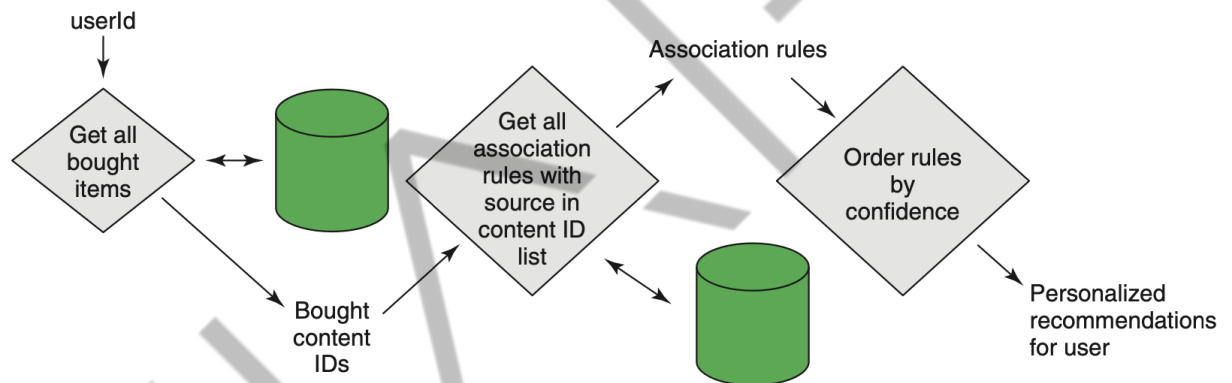


Figura 3 - Regras de associação para o problema do Cold Start

Fonte: Practical Recommender Systems (2019)

Outra forma de recomendação para lidar com o cold start é usar regras de negócio e conhecimento de domínio. Por exemplo, se um usuário navega pela seção de desenhos animados, apenas recomende desenhos animados e filmes “família”.

De outra maneira, podemos especificar coisas que nunca devem aparecer juntas. Exemplo: nunca recomende um filme de terror para alguém que está assistindo desenho animado.

Podemos implementar isso da seguinte maneira:

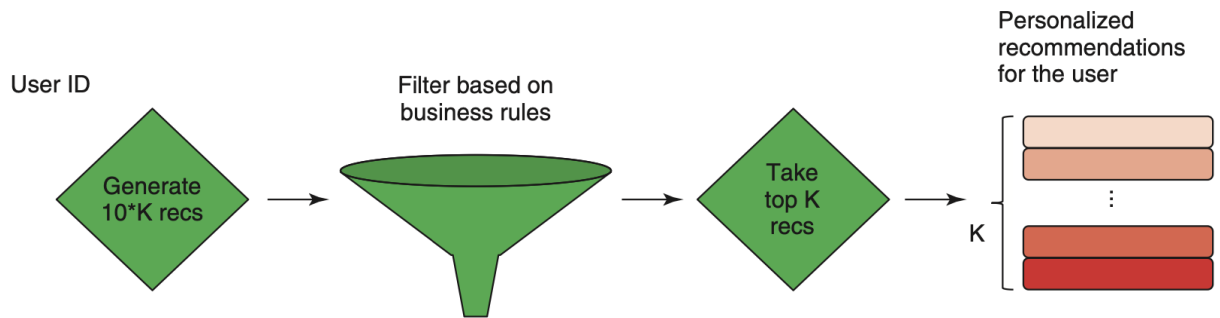


Figura 4 - Regras de negócios para o problema do Cold Start

Fonte: Practical Recommender Systems (2019)

Uma terceira maneira de lidar com o problema de cold start é usar segmentação. A ideia é agrupar usuários com gostos similares, de modo que você consiga descobrir que tipo de pessoa gosta de que tipo de conteúdo. Quando um novo usuário chega, podemos recomendar conteúdo popular com aquele segmento. Isso é conhecido como recomendações demográficas.

De maneira simples, você filtra o ip do usuário, determina sua localização e, então, oferece produtos vendidos nessa localização. Se você sabe o gênero, você pode apresentar produtos mais vendidos para cada um deles. O mesmo serve para idade. Esses são os segmentos mais óbvios de serem encontrados.

Por outro lado, podemos usar algoritmos não supervisionados (clustering) para agrupar nossos usuários conhecidos. Depois, podemos tentar encaixar um usuário novo em um desses grupos e realizar as recomendações.

Embora esse seja um problema para sistemas de recomendação, resolvê-lo não será o nosso foco aqui. Nossa ideia é usar as técnicas que vimos até o momento para construir um modelo de recomendação. Mas como eu determino se um produto é similar a outro ou um usuário é similar a outro?

Como calcular similaridade

Precisamos falar sobre similaridade porque queremos encontrar itens semelhantes àqueles que gostamos ou porque queremos encontrar usuários que gostem das mesmas coisas que gostamos.

Entretanto, como definimos similaridade? Por exemplo, numa escala entre -1 a 1, quão similar duas pessoas são? Se uma pessoa gosta de Star Trek e outra gosta de Star Wars, elas são similares?

Assim, precisamos de uma maneira de mensurar similaridade, que pode ser obtida através de uma função de similaridade, definida da seguinte maneira:

- Dados dois itens, i_1 e i_2 , a similaridade entre eles é dada por uma função $\text{sim}(i_1, i_2)$.

Geralmente tomamos o domínio $[0,1]$ para determinar a similaridade entre dois itens, com 0 sendo totalmente dissimilares e 1 sendo totalmente similares (mesmo item).

A figura 5 ilustra dois exemplos de funções de similaridade. É importante pensar sobre qual deve ser escolhida, visto que determinar qual é melhor depende do domínio (contexto) e dos dados:

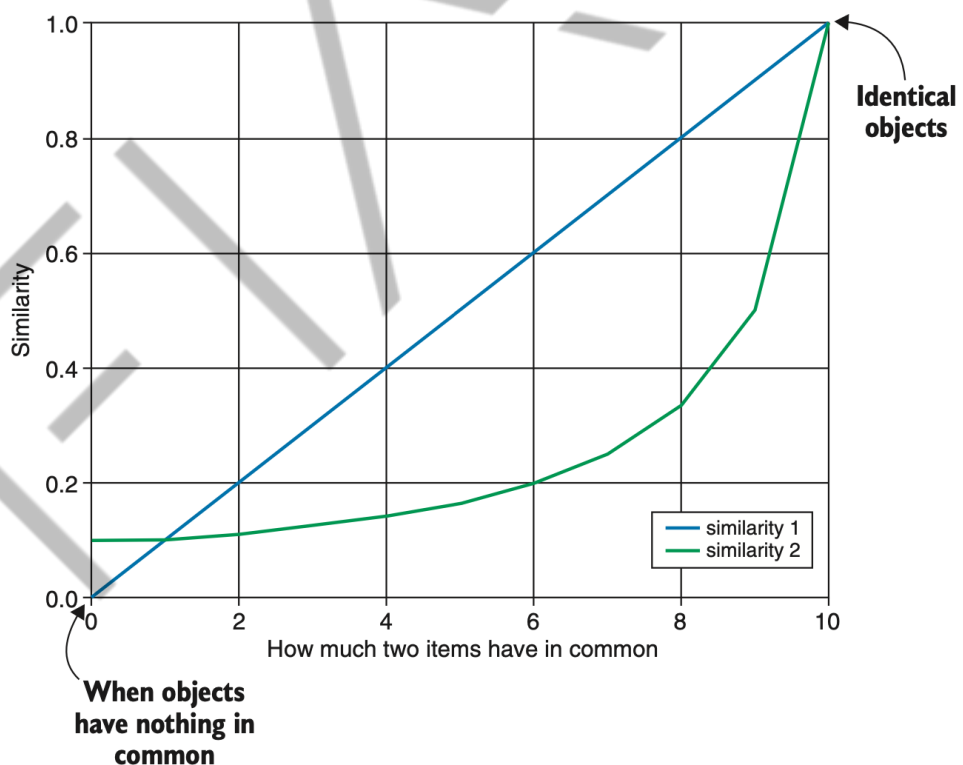


Figura 5 - Exemplo de medidas de similaridade

Fonte: Practical Recommender Systems (2019)

A medida de similaridade é intimamente relacionada com o cálculo da distância entre itens. Geralmente, podemos dizer que a relação entre similaridade e distância é a seguinte:

- Quando a distância aumenta, a similaridade tende a zero.
- Quando a distância tende a zero, a similaridade tende a um.

Como mencionado, não existe método certo ou método errado de similaridade. Diferentes métodos funcionarão melhor em diferentes datasets. Entretanto, existem alguns pontos de partidas e, aqui, vamos discutir algumas medidas de distância:

- Jaccard;
- Euclidiana;
- Cosseno;
- Pearson.

A distância de Jaccard é utilizada para indicar quão próximos dois conjuntos são. A explicação é simples: podemos dizer que cada filme é uma bag que contém todos os usuários que compraram esse filme. Assim, temos conjuntos de usuários, um para cada filme. Dessa maneira, podemos comparar dois filmes olhando para dois conjuntos de usuários.

Datasets como o mencionado anteriormente podem ser produzidos a partir das transações dos usuários, que podem ser transformadas numa lista onde cada linha indica se um usuário comprou um produto ou não (1 = comprou, 0 = não comprou).

Considere a seguinte imagem:







						
	Comedy	Action	Comedy	Action	Drama	Drama
Sara	1	1	0	0	0	0
Jesper	1	1	1	0	0	0
Therese	1	0	0	0	0	0
Helle	0	1	0	1	0	0
Pietro	0	0	0	0	1	1
Ekaterina	0	0	0	0	1	1

Figura 6 - Dados para calcular similaridade

Fonte: Practical Recommender Systems (2019)

Para calcular a similaridade entre dois itens, calculamos quantos usuários compraram (ou não) ambos os itens e, então, dividimos por quantos usuários compraram um ou outro. Formalmente, temos:

$$jaccard_{(i,j)} = \frac{\# \text{ users compraram (ou nao) ambos os itens}}{\# \text{ users que compraram ou i ou j}}$$

Observe o exemplo:

			
Sara	1	1	1
Jesper	1	1	1
Therese	1	0	0
Helle	0	1	0
Pietro	0	0	1
Ekaterina	0	0	1
			Sum = 4

Figura 7 - Calculando similaridade de Jaccard

Fonte: Practical Recommender Systems (2019)

A similaridade de Jaccard para esse caso é a seguinte:

$$jaccard_{(i,j)} = \frac{4}{6} = 0.67$$

Quando meu conjunto de dados é composto, por exemplo, de ratings de filmes, uma métrica mais apropriada é a distância Euclidiana, derivada do Teorema de Pitágoras. Ela pode ser entendida como o tamanho do segmento de linha que junta dois pontos de dados plotados num plano cartesiano n-dimensional.

Veja a figura 8 como exemplo:

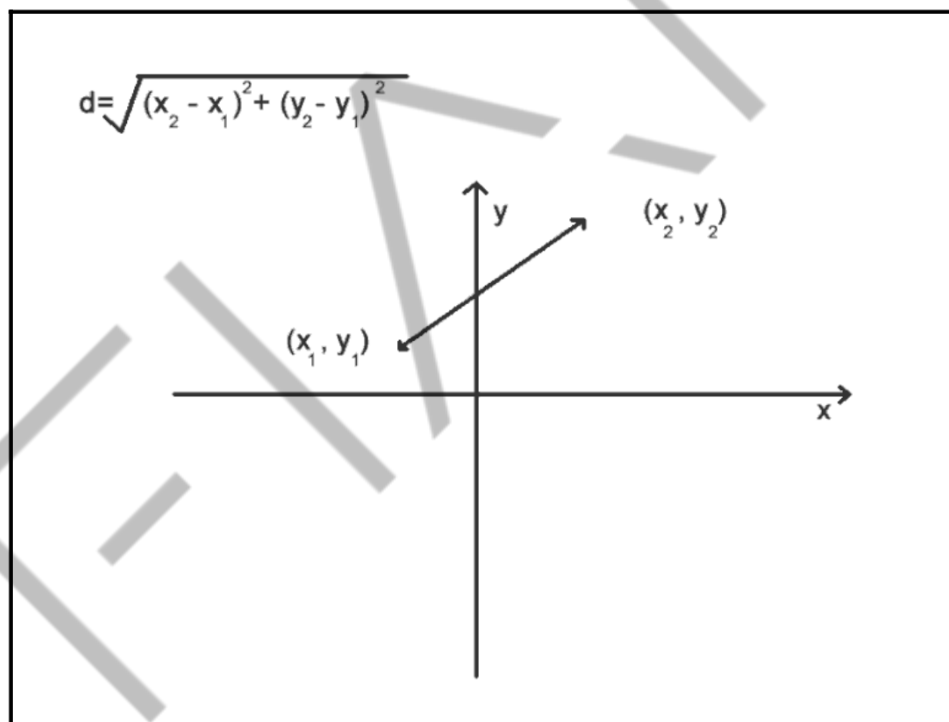


Figura 8 – Distância Euclidiana

Fonte: Practical Recommender Systems (2019)

Os pontos no espaço n-dimensional são dados por vetores, definidos da seguinte maneira:

- $v_1: (q_1, q_2, \dots, q_n)$
- $v_2: (r_1, r_2, \dots, r_n)$

Assim, podemos definir matematicamente a distância euclidiana da seguinte maneira:

$$d(v_1, v_2) = \sqrt{\sum_{i=1}^n (q_i - r_i)^2}$$

A distância pode variar entre 0 e infinito. Quanto menor seu valor, mais similares dois vetores são um do outro.

Uma outra maneira de olhar para conteúdos é enxergar as linhas de uma matriz de rating como vetores no espaço e, então, determinar o ângulo entre eles.

O cálculo é dado pela seguinte fórmula:

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Quando o valor da similaridade é 1 (ângulo = 0), os vetores são exatamente similares. Por outro lado, um valor de cosseno igual a -1 (ângulo = 180) significa que os vetores são totalmente diferentes.

Considerando ainda o exemplo de um conjunto de ratings de filmes para dois (ou mais) usuários diferentes, podemos usar a correlação de Pearson para determinar a similaridades entre dois usuários.

A correlação de Pearson mensura o quão diferente cada ponto é na média. O range varia entre [-1,1]. Com esse coeficiente, os ratings são normalizados, subtraindo o rating médio do item para cada rating, como mostrado na equação a seguir.

$$\text{pearson} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

Tendo visto as principais medidas de distância usadas na construção de sistemas de recomendação, estamos prontos e prontas para efetivamente começar a construir um sistema de recomendação. Vamos lá? Lembrando que, nas aulas em vídeo, exploraremos um pouco mais cada recommender, onde vocês também poderão ver os códigos.

Simple Recommender

Para continuarmos a entender os sistemas de recomendação, usaremos como base de comparação a lista de 250 filmes do IMDB, calculados segundo uma métrica específica. Todos os filmes na lista são não-documentário, lançamentos de cinema, com pelo menos 45 minutos e têm mais de 250.000 avaliações.

Como vimos, isso é um tipo de recomendação. Usaremos o conjunto de dados “The Movies Dataset” para criar uma lista parecida, na intenção de oferecer uma recomendação inicial.

Esse não é um tipo de recomendação personalizada. Ele apenas leva em conta as maiores notas que cada filme presente na base recebeu. A premissa é que, se muitas pessoas assistiram e gostaram, talvez o usuário também deva assistir.

Veremos, agora, outro tipo de sistema de recomendação, que já leva em consideração as preferências do usuário.

Knowledge-based Recommender

Vamos criar um sistema um pouco mais complexo para fazer recomendações. Realizaremos essas tarefas:

- Perguntar ao usuário seus gêneros preferidos;
- Perguntar ao usuário a duração do filme;
- Perguntar ao usuário a linha do tempo de filmes recomendados.

Usando essas informações coletadas, recomende filmes ao usuário que possuam um alto score e que satisfaça as condições anteriores.

É um tipo de recomendação que leva em consideração os gostos do usuário, mas não considera o seu histórico e é custosa para ele, visto que precisaria ficar inserindo essas informações toda vez que fosse procurar um filme. Além disso, ela é estática. Só seria alterada caso novos filmes fossem inseridos no conjunto de dados.

Filtros baseados em conteúdo

Os recommenders que construímos anteriormente são extremamente primitivos. O simple recommender não leva em consideração as preferências individuais e, enquanto o knowledge recommender leva em consideração a

preferência do usuário por gêneros, época de lançamento e duração, o modelo continua ainda muito genérico.

Por exemplo, se uma pessoa gosta dos filmes The Dark Knight, Homem de Ferro e Superman, claramente ela tem uma preferência por filmes de super-heróis. Entretanto, nosso modelo não será capaz de capturar tais detalhes. O máximo que conseguirá fazer é fornecer recomendações de filmes de ação, que é um gênero maior que envolve os filmes de super-heróis.

Poderíamos solicitar ao usuário mais informações sobre seus gostos e criar, por exemplo, uma categoria “subgênero”. No entanto, não temos informação pra isso e capturar todas as informações necessárias via usuário seria maçante e muito custoso.

Ao invés disso, o que os usuários costumam fazer é classificar seus filmes favoritos e o sistema apresenta a eles os mais similares.

Aqui, vamos construir dois tipos de filtros baseados em conteúdo:

Recommender baseado na descrição dos filmes, que vai comparar a descrição de diferentes filmes e prover recomendações de filmes que possuam descrição similar, e recommender baseado em metadados, que usa como features gêneros, keywords, cast, entre outros e faz recomendações baseadas nessas características.

Recommender baseado na descrição dos filmes

Essencialmente, os modelos que estamos construindo calculam a similaridade entre os textos das descrições, mas como conseguimos quantificar a similaridade entre dois textos?

Bem, usaremos as técnicas que vimos rapidamente durante a aula: vetorização. Mais precisamente, trabalharemos com o TF-IDF.

Basicamente, nosso modelo receberá o título do filme como argumento e retornará uma lista de filmes similares baseada na descrição. Estes são os passos que executaremos para construir o nosso modelo:

- Obter os dados requeridos para construir o modelo;
- Criar vetorização usando TF-IDF para cada filme;

- Calcular a similaridade do cosseno par-a-par para cada filme;
- Escrever uma função que recebe um título de filme como entrada e retorna uma lista de filmes similares.

Percebemos que nosso recommender funciona relativamente bem. A maioria das recomendações fazem sentido já que, de uma maneira ou outra, trata de filmes com leões em suas descrições. Entretanto, quem assistiu o Rei Leão provavelmente gostaria de ver recomendação de filmes das Disney, mas nosso recommender não consegue capturar essas nuances.

Felizmente, temos uma maneira de resolver esse problema. É o que vamos fazer com o recommender baseado em metadados.

Recommender baseado em metadados

Basicamente, seguiremos os mesmos passos dados anteriormente para criar nosso recommender. Entretanto, os dados a serem utilizados serão diferentes. Vamos usar os seguintes metadados:

- Gênero do filme;
- O diretor do filme;
- Três maiores estrelas do filme;
- Subgêneros ou Keywords.

As recomendações, nesse caso, foram muito diferentes das anteriores. Vimos que nosso recommender foi capaz de capturar mais informações que apenas 'leões'. Muitos dos filmes na lista são animações e dizem respeito a personagens antropomórficos.

O QUE VOCÊ VIU NESTA AULA?

Nesta aula, apresentamos o problema de recomendações. Definimos o problema e listamos diversas abordagens de como podemos fazer recomendações. Para isso, apresentamos o conceito de similaridade e as principais técnicas de recomendação, que são os filtros colaborativos.

O que achou do conteúdo? Conte-nos no Discord! Estamos disponíveis na comunidade para fazer networking, tirar dúvidas, enviar avisos e muito mais. Participe!

REFERÊNCIAS

ANDERSON, C. **The Long Tail: Why the Future of Business is Selling Less of More**. [s.l.]: Hyperion, 2011.

FALK, K. **Practical Recommender Systems**. Nova York: Manning Publications, 2019.

IMDB. **Os 250 melhores filmes do IMDB**. Disponível em:
<<https://www.imdb.com/chart/top/>>. Acesso em: 04 out. 2023.

Kaggle. **The Movies Dataset**. Disponível em:
<<https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset>>. Acesso em: 04 out. 2023.

PALAVRAS-CHAVE

Palavras-chave: Sistemas de Recomendação, Similaridade, Filtros colaborativos.

EMSE

The background is a dark blue gradient with abstract, wavy lines in shades of teal, yellow, and red. Scattered throughout are small, light blue dots and various geometric shapes, including a circle with the number 7, a circle with an 'x', a circle with a '0', and a hexagon.

POSTECH