UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
ENGENHARIA DE COMPUTAÇÃO

Jonathan Gouvea da Silva

# Investigação de abordagens híbridas para sistemas de recomendação implícitos

São Carlos - SP

2020

Jonathan Gouvea da Silva

# Investigação de abordagens híbridas para sistemas de recomendação implícitos

Trabalho de conclusão de curso apresentado para obtenção do título de Engenheiro de Computação

Orientação Prof. Dr. Jander Moreira

São Carlos - SP

2020

*Agradeço à minha família por seu apoio em toda a minha trajetória acadêmica. Esse trabalho é dedicado a eles.*

# Agradecimentos

Primeiramente agradeço a Deus que permitiu que tudo isso acontecesse. Agradeço à minha mãe, avós maternos e tios, por sempre me incentivarem e por me fazerem ter confiança em minhas decisões.

Agradeço ao meu professor orientador Dr. Jander Moreira, pelas valiosas contribuições dadas nesse trabalho e em trabalhos passados.

Agradeço também à toda a Universidade de São Carlos e o seu corpo docente, que proporcionaram um ambiente onde pude crescer intelectualmente e profissionalmente.

Sou grato a todos os amigos, com quem convivi intensamente nos últimos 5 anos, que me permitiram crescer não apenas como profissional, mas também como pessoa, e agradeço pelo genuíno apoio que demonstraram ao longo de tantos anos.

*"What I cannot create, I do not understand.*
(*Richard Feynman*)

# Resumo

Esse projeto trata da investigação de abordagens de sistemas de recomendação implícitos, em relação à acurácia e versatilidade dos sistemas, além da hibridização como ferramenta para melhora dessas métricas de desempenho. Foram utilizadas bases de dados de hábitos musicais de usuários.

**Palavras-chave**: Sistemas de recomendação, dados implícitos, recomendação de músicas.

# Abstract

This project deals with the investigation of implicit recommender systems, in relation to the accuracy and versatility of the systems, and of the hybridization as a tool to improve these performance metrics. Databases of users's musical habits where used.

**Keywords**: Recommender systems, implicit data, song recomendation.

# Lista de ilustrações

# Lista de tabelas

# Sumário

# 1 Introdução

Uma das mudanças da era da informação é de permitir que, a partir de coletas de dados, negócios ofereçam experiências aos seus clientes de forma personalizada e sem perda de escala. Essas experiências vão de recomendações de diferentes produtos ou serviços que combinam com o gosto desse cliente até a personalização completa dos ambientes virtuais que ele frequenta.

Essa recomendação pode ser construída a partir de diferentes técnicas, utilizando dados destes produtos que podem ser sugeridos ou a partir de padrões de interações entre os usuários e os itens com que têm contato. As diferentes técnicas, somadas à grande quantidade de variáveis que podem estar disponíveis, geram uma enorme gama de possibilidades. Entretanto, como usar dados brutos pode levar a resultados muito pobres (LIAN et al., 2018), se faz necessário o uso de diferentes técnicas para transformar os dados e a utilização de algoritmos adequados para cada conjunto de dados.

Os sistemas de recomendação são utilizados por muitas empresas, de diferentes domínios, e cada domínio traz diferentes desafios. As áreas de recomendação de vídeos, músicas, notícias, para *e-commerces* e redes sociais são extremamente importantes nesses domínios (AMATRIAIN; BASILICO, 2016). A necessidade crescente de personalização, explicabilidade de recomendações e escalabilidade fazem com que essa área esteja sempre evoluindo.

Este estudo busca avaliar o desempenho de sistemas de recomendação que utilizam fatores implícitos, como dos cliques em uma notícia, e fatores próprios aos dados, como as característica de um produto que está sendo vendido, e analisar possibilidades de unir ambos os sistemas.

Neste contexto, o trabalho se divide em quatro partes, embasamento teórico, metodologia, descrição dos experimentos e apresentação de resultados.

A primeira parte apresenta os Sistemas de Recomendação, as principais classificações e quais são os tipos de parâmetros que são analisados em cada sistema. Além disso, ele apresenta conceitos de hibridização e traça paralelos com trabalhos relacionados.

Em seguida, são apresentadas as bases de dados e *softwares* que foram utilizados. Na terceira parte os experimentos são descritos em quatro cenário.

Por fim, há a apresentação dos resultados encontrados e comparações entre os diferentes cenários.

Após o percurso acima, finaliza-se com conclusões referentes aos resultados encontrados.

# 2 Embasamento teórico

## 2.1 Sistemas de Recomendação

Sistemas de Recomendação (SRs) são ferramentas que geram sugestões de itens para um usuário (RICCI et al., 2011). Um item é tudo aquilo que pode ser recomendado, desde produtos, notícias, músicas etc. Em um e-commerce que oferece milhares de itens, um Sistema de Recomendação pode auxiliar sugerindo a um usuário um item que será útil para ele, de forma personalizada onde diferentes usuários recebem diferentes recomendações.

De forma geral, os algoritmos tentam calcular esta "utilidade" dos itens disponíveis para um usuário, ou um grupo de usuários em alguns contextos, e, dessa forma, é possível apontar os itens "mais úteis" com base nessa métrica calculada e, então, fazer a sugestão para o usuário.

Esse aspecto das sugestões serem individuais e personalizadas é o que difere um SR de um sistema de busca comum (BURKE, 2002). É possível presumir que os algoritmos que alcançam os melhores resultados são aqueles que oferecem a melhor experiência ao usuário. Porém, essa experiência também é composta por fatores externos, situacionais ou pessoais (KNIJNENBURG et al., 2012).

Um dos aspectos pessoais que podem afetar a experiência é o grau de domínio do contexto por parte do usuário. Em um sistema de recomendação de filmes, por exemplo, um sistema voltado para experts da indústria pode acabar fazendo sugestões com base nos diretores e roteiristas de um filme, mas uma recomendação nesse sistema parecerá não ter sentido quando vista por uma pessoa leiga que não conheça muitos detalhes sobre as equipes criativas. Da mesma forma, um sistema que use apenas dos gêneros de um filme e da data de lançamento dos mesmos pode ser ótimo para o público geral, mas não seria tão relevante para experts.

Da mesma forma, se um sistema de recomendação de restaurantes passa a fazer recomendações com base nos locais que o usuário passou enquanto caminhava, por exemplo, ele pode levantar preocupações quanto à segurança e à sua privacidade. Nesse caso, o usuário pode descartar as recomendações, por "melhores" que sejam, por conta desse fator de privacidade.

Entre os aspectos situacionais que podem influenciar estão a forma pela qual a recomendação é apresentada. Uma recomendação de um restaurante por meio de um aplicativo, por exemplo, pode ser que seja mais bem recebida em determinado horário ou até mesmo pela forma na qual essa recomendação foi feita, o teor do texto usado,

por exemplo.

Entre os fatores externos, pode se citar a presença do contato social do usuário com outros indivíduos, com os quais ele pode passar a fazer escolhas, de qual filme assistir por exemplo, muito diferentes das suas escolhas usuais, e isso pode fugir ao contexto que o sistema de recomendação esperava. Nesse caso, uma recomendação do sistema pode ser preterida por uma escolha que não tenha o perfil desse usuário, por exemplo.

### 2.1.1 Classificações de SRs

Para realizar as previsões, o domínio dos dados, quais premissas foram assumidas e o tipo de algoritmo podem ser os mais variados. Há seis classificações clássicas para esses algoritmos, como aponta Burke (2007a). Estas classificações são as seguintes.

#### Content-based

Um sistema *content-based* sugere itens parecidos com aqueles que o usuário "gostou". Esta similaridade é calculada com base em variáveis próprias ao item, como o gênero de um filme, o ritmo de uma música ou a cor de uma peça de roupa.

Logo, o objetivo é de criar um classificador que possa abstrair o método de avaliação de itens por um usuário específico a partir das características e avaliações de itens que o usuário interagiu.

Um exemplo seria um sistema de recomendação de filmes que se baseia em recomendar um filme dos gêneros que o usuário melhor avaliou. Nesse caso, pode se calcular quais filmes têm gêneros mais próximos àqueles que o usuário viu, e a sugestão é feita para os filmes mais similares.

#### Collaborative Filtering

O sistema *collaborative filtering* tenta recomendar, para um determinado usuário, itens que usuários com perfil semelhante ao seu avaliaram positivamente, como sugerir um filme porque "pessoas como você viram e gostaram desse filme". Esta é a abordagem mais popular dentre todas (RICCI et al., 2011).

Portanto, este algoritmo deve identificar usuários com gostos parecidos e conseguir extrapolar as suas avaliações para os outros usuários que não tiveram contato com um determinado item.

Nesse caso, um exemplo para um sistema de recomendações de filmes seria aquele que sugere um filme para o usuário porque 'quem gostou do filme X costuma gostar do filme Y'. Nesse caso, o sistema leva em conta as avaliações de outros usuários

e, assim, descobre a qual grupo de usuários esse mais se encaixa, e recomenda o que esse grupo costuma gostar.

### Demographic

Sistemas classificados como *demographic* realizam recomendações com base no perfil demográfico do usuário, como sua língua ou o seu país ou estado de origem. Nestes, o objetivo do sistema é de identificar os grupos demográficos dentre os usuários e de realizar sugestões com base nas experiências desse grupo.

Em um sistema *demographic* para sugestão de filmes, fatores como idade, gênero e nacionalidade, por exemplo, podem ser determinantes para a recomendação de qual filme o usuário pode gostar.

### Knowledge-based

Sistemas *knowledge-based* utilizam informações específicas do domínio para construir uma função que tenta correlacionar as necessidades de um usuário com as propriedades dos itens em seu catálogo. Nesse caso, os especialistas do domínio que definem as variáveis que são relevantes para a resolução do problema de recomendação.

No exemplo de sistemas para filmes, para construir um sistema desse tipo podem ser elencados grandes especialistas da indústria, que juntos podem definir que a idade de um usuário tenha um peso X, enquanto a profissão dele tem outro peso Y, em que cada profissão tenha uma classe de filmes que podem ser recomendados.

### Community-based

No caso do *community-based*, o sistema utiliza as preferências dos amigos do usuário para inferir sugestões. Essa é uma classificação relativamente nova dentro da área.

Nesse caso, uma rede social que queira construir um recomendador de filmes, nesse tipo de sistema, pode fazer a recomendação puramente baseada no que os amigos ou amigos de amigos dessa pessoa assistiu, ou a partir dos grupos que ele frequenta.

### Hybrid

Por fim, diversos sistemas são compostos pela combinação das técnicas acima, conseguindo explorar os dados de forma mais profunda e trazendo uma solução para os problemas que algumas técnicas apresentam quando aplicadas sozinhas. Estes são classificados como *hybrid*.

Ainda no exemplo de recomendadores para filmes, a famosa competição da Netflix premiou um sistema híbrido, composto de 107 algoritmos diferentes (AMATRIAIN;

BASILICO, 2016), em seu primeiro ano. Um algoritmo híbrido pode unir, por exemplo, todos tipos que foram citados anteriormente em apenas um.

### 2.1.2 Interações com os SRs

Um dos problemas na construção de sistemas é conseguir quantificar preferências qualitativas. Por não haver uma forma direta de traduzir os desejos e objetivos de um usuário através de números, diferentes métodos surgem para tentar modelar essas preferências.

Esses métodos variam dos implícitos aos explícitos. Fatores implícitos são aqueles em que o usuário não está ativamente envolvido na avaliação, nos quais o sistema se baseia em informações que o usuário pode não estar consciente. Exemplos de *feedbacks* implícitos variam dos sentimentos em um comentário do usuário, a posição do mouse, para quais locais o usuário está olhando etc. (POMMERANZ et al., 2012).

Os *feedbacks* explícitos são aqueles em que o usuário está totalmente consciente de sua avaliação, onde ele pode dar notas, avaliar como positivo ou negativo, ou mesmo avaliar múltiplos fatores. As interações variam de implícitas à explícitas, não precisando ficar, necessariamente, restritas a um lado ou outro, com uma grande variedade de graus de envolvimento do usuário.

Uma das diferenças entre os fatores explícitos e implícitos é que nos implícitos não há um *feedback* negativo. Caso o objetivo de um SR seja de recomendação de músicas, enquanto aquele que use fatores explícitos utiliza as notas de um usuário que indicarão aquilo que ele não gostou e aquilo que ele gostou, em um cenário implícito poderia ser avaliado apenas o fato do usuário já ter escutado determinada música ou não. Nesse caso, a não interação do usuário com este item pode significar desconhecimento, em que o usuário nunca teve contato com esse item ou que não teve interesse em conhecê-lo, ou que o usuário já despreze esse item como algo que ele não gostaria.

Além disso, o fato de um usuário ter escutado a uma música, ou feito a compra de um item, não significa, em um cenário implícito, que ele tenha gostado desse item e que desejaria receber recomendações de outros semelhantes.

Como apontado por Hu, Koren e Volinsky (2008), enquanto fatores explícitos indicam uma preferência, os implícitos apenas indicam a confiança, já que um item que reaparece frequentemente tem maior chance de refletir os gostos e desejos dos usuários.

Esses fatores implícitos podem ser transformados em fatores explícitos, através de considerações. O tempo que um usuário fica em uma página é um fator implícito, mas pode ser catalogado e transformado em explícito se considerado que, por exemplo, se o usuário ficar 1 minuto ele "não gostou" e se ele ficar 10 minutos ele "amou" essa página. Sendo assim, as variáveis podem ser consideradas mais ou menos implícitas,

dependendo do entendimento de preferência que elas podem trazer.

## 2.2 Técnicas de PLN aplicadas em SRs

Em sistemas de recomendação *content-based*, muitas vezes os metadados disponíveis não estão presentes como categorizações ou como dados numéricos, e sim em outros formatos multimídia, que exigem um esforço para serem transformados em *features* que sistemas podem utilizar. No caso de dados em formato textual, existem diversas técnicas que podem ajudar a entender as relações entre diversos itens (WILSON; WIEBE; HWA, 2006).

No caso de informações como gêneros musicais, nomes de artistas envolvidos ou de fabricantes de produtos, podem ser representados como um modelo saco de palavras (*bag of words*), em que o texto é representado como um *multiset* de palavras, em que é avaliada apenas a recorrência da palavra, desprezando a ordem das palavras. Um modelo mais robusto é o de representação de palavras como vetores multidimensionais, que podem permitir que as informações semânticas se mantenham. (BERBATOVA, 2019).

Além disso, a análise do valor de *tf-idf*, *term frequency–inverse document frequency*, também é bastante utilizada para a análise de textos. Nela, é possível identificar os termos mais importantes para um determinado texto, quando comparado com outros textos no mesmo contexto.

Outro lado bastante utilizado do processamento de língua natural em SRs é a análise de sentimentos (LEUNG; CHAN; CHUNG, 2006). Com ela, pode-se analisar textos e extrair os sentimentos descritos nela, como positivos a negativos ou de formas mais complexas. Com isso, esse valor pode ser usado para interferir em *features* explícitas de modelos de *collaborative filtering*.

## 2.3 Sistemas de Recomendação Híbridos

Ao combinar duas ou mais técnicas de recomendação, os sistemas de recomendação híbridos ganham performance ao passo que diminuem as desvantagens dos sistemas de recomendação isolados. Existem diversas técnicas para tornar esses sistemas híbridos, como apresentadas por Burke (2007b).

### Weighted

Nos recomendadores híbridos *weighted*, ou ponderados, cada recomendador associado tem um peso. Nesse caso, para uma recomendação para um usuário, podem ser analisados os *scores* de cada item em cada recomendador, e eles se unem como uma

média ponderada a partir dos pesos do algoritmo e assim são decididas recomendações. Outra abordagem não utiliza os *scores* e sim a saída pura de cada algoritmo, e é feito um mecanismo de voto, levando em conta os pesos de cada algoritmo, para selecionar as possíveis recomendações.

Nesses sistemas, é assumido que cada recomendador pode contribuir de forma uniforme e da mesma forma para todos os itens e usuários, o que nem sempre é verdade. As vantagens desse sistema estão na facilidade de implementação e uso, além de manter a explicabilidade. Para a escolha dos pesos, podem ser utilizadas diferentes técnicas através de testes para encontrar a melhor combinação.

A explicabilidade é uma qualidade que pode ser percebida em alguns sistemas de recomendação, e é um dos elementos presentes no *framework* de avaliação centrada no usuário para SRs apresentada por Pu, Chen e Hu (2011). Essa qualidade gera valor a um sistema quando, ao apresentar a explicação por trás da recomendação de um item para o usuário, gera confiança e satisfação e ajuda a aumentar o envolvimento do usuário com o sistema.

#### Switching

Um classificador híbrido *switching* seleciona um recomendador dentre os seus membros. Dependendo do perfil do usuário, um sistema de recomendação diferente pode ser utilizado, e, assim, o sistema híbrido consegue contornar as dificuldades que determinado recomendador pode ter.

Algoritmos desse tipo requerem que o critério para a troca do recomendador seja confiável, tal como uma métrica do valor de confiança. Nesse caso, um exemplo seria um ambiente que use o sistema de recomendação A por padrão e, quando é gerada uma previsão com uma confiança muito baixa, o sistema B é utilizado e as recomendações desse passam a ser utilizadas. O significado dessa confiança "muito baixa" depende do contexto e testes experimentais.

Nesses sistemas, esse racional para realizar a troca pode elevar substancialmente a complexidade do sistema, sendo um novo parâmetro que pode ser ajustado.

#### Mixed

Nos híbridos do tipo *mixed* as recomendações dos diferentes algoritmos são apresentadas ao mesmo tempo para o usuário. Uma dificuldade que surge é a forma de combinação das listas, em qual ordem devem aparecer. Isso porque a ordem de uma lista de recomendações pode gerar diferentes resultados, já que as primeiras posições são consideradas mais relevantes para o usuário.

Além disso, caso sejam recomendados muitos itens ao mesmo tempo, um pro-

blema pode surgir. Muitas sugestões podem fazer com que o usuário perca a atenção para as recomendações e as ignore.

#### Feature Combination

Na classe de algoritmos híbridos do tipo *feature combination*, as variáveis dos diferentes recomendadores são unidas em um único algoritmo de recomendação. Por exemplo, em uma abordagem que mescla *content-based* e *collaborative filtering*, um possível híbrido seria colocar os dados colaborativos como parte do modelo de dados, e utilizar técnicas baseadas em conteúdo por cima desse novo dataset.

#### Cascade

Em técnicas de hibridização *cascade*, é gerado um híbrido hierárquico, dependente da ordenação dos seus componentes. Em um sistema de recomendação desse tipo, um primeiro SR (forte) gera um conjunto de sugestões e, em caso de empates, um segundo SR (fraco) faz as sugestões de forma a desempatar. A definição de empate para esse contexto não é, geralmente, de *scores* matematicamente iguais, e sim de valores próximos, dentro de um intervalo definido por testes e dependente do contexto.

#### Feature Augmentation

Na técnica de *feature augmentation*, um recomendador gera uma classificação ou sugestão de nota para cada item, e o recomendador seguinte utiliza essa nova feature, além de outras já presentes no item, para realizar uma recomendação.

## 2.4 Descrição de Arquiteturas e Algoritmos

Nesta seção, serão apresentados os dois algoritmos que serão utilizados nos experimentos, junto às arquiteturas em que cada um está envolvido. De início, será apresentada uma arquitetura de sistema de recomendação do tipo *content-based* e, em seguida, de um *collaborative filtering*, com a descrição do algoritmo *Alternating Least Squares*.

### 2.4.1 Content-Based: Nearest-Neighbours

O foco do desenvolvimento de algoritmos *content-based* está em desenvolver um modelo que se baseia nas informações presentes nos itens de interesse de cada usuário. Estes são compostos de três partes, sendo elas o analisador de conteúdo, que irá fazer a extração das informações relevantes de um dado e que está muito relacionado à área de Recuperação da Informação; uma segunda parte que trata do aprendizado de perfis

do usuário, que tem o objetivo de construir um perfil consolidado para cada usuário, baseado nas variáveis extraídas no passo anterior; e uma terceira etapa de filtragem, que tratará os feedbacks do usuário e os contextos em que ele está inserido, para oferecer as melhores recomendações.

Para a segunda etapa, de construção de perfil, deve ser levado em conta o processo para construir um perfil de usuário e o processo de comparação desse perfil com os outros dados, para extrair recomendações.

Em um modelo que utilize dados explícitos, o objeto do construtor de perfil será gerar um perfil de acordo com os "gostos" do usuário, priorizando aqueles itens mais bem avaliados. Entretanto, em um cenário implícito isso não é possível e será necessário construir alguma lógica de cálculo de relevância para os itens que o usuário teve contato.

No cenário tratado, e em diversos contextos distintos, é o número de interações entre o usuário e item que definirá a sua relevância. No caso das músicas, uma música ouvida três vezes terá o triplo de relevância de uma música ouvida uma vez. Para a construção do perfil de usuário, será feita uma ponderação entre os vetores de *features* dos itens levando em conta o número de interações, formando um perfil médio.

Com o perfil médio, podem ser usados diversos algoritmos para a geração de recomendações. O algoritmo selecionado será o de *nearest-neighbours*, que considera que os itens mais próximos têm maior semelhança e, por tanto, mais compatibilidade com os gostos do usuários. Logo, o algoritmo para a recomendação para um usuário irá selecionar os itens, não vistos pelo usuário, mais próximos ao seu perfil.

Esse cálculo da distância pode ser feito de diversas formas, sendo a distância do cosseno amplamente adotada quando os itens estão mapeados no modelo de vetores no espaço (Vector Space Model), que é o caso da maioria das abordagens (SARWAR et al., 2001).

As principais vantagens dessa abordagem é a independência de usuários, já que os dados utilizados para a criação de uma recomendação são formados apenas pelos itens que o próprio usuário interagiu e os dados do universo de itens existentes. Além disso, há uma certa transparência e explicabilidade nos modelos, e o sistema suporta o surgimento de novos itens com facilidade, já que ele não depende de que algum usuário interaja com esse item para ele ser sugerido.

Entretanto, as desvantagens estão nas limitações dos dados e do analisador de conteúdo, já que esse pode ter ignorado informações relevantes ou utilizar apenas de informações inválidas, já que nestes modelos não são construídas relações entre grupos de dados, ou usuários, como em outros. Nesse a similaridade entre dois itens é explicada pela similaridade matemática entre um conjunto de variáveis, que pode não ser o suficiente para realmente traçar recomendações entre itens. Além disso, essa

arquitetura tem dificuldade em sugerir itens muito diferentes, se tornando viciada em sugerir mais do mesmo. Ela também não lida bem com novos usuários, já que eles não tem um perfil inicial bem estabelecido.

Por fim, o algoritmo de Nearest-neighbour também possui uma desvantagem em eficiência, já que ele não possui uma fase de treinamento e todo o processamento acaba sendo feito na hora de criação de recomendações. Logo, o custo ao realizar uma recomendação é alto, e quanto mais recomendações, para mais usuários, o custo continua a aumentar e se torna maior que o tempo de processamento de muitos outros algoritmos.

### 2.4.2 Collaborative Filtering - Alternating Least Squares

O algoritmo de Mínimos Quadrados Alternados (*alternating least squares* – ALS) foi criado inicialmente para bases de dados com variáveis explícitas, em que os dados desconhecidos são tratados como faltantes e que a função objetivo é prever qual a avaliação desses dados incompletos. Dado um conjunto de dados, em um cenário baseado em vizinhança, o valor previsto para um determinado dado é a média ponderada entre os valores de seus usuários vizinhos, utilizando uma métrica de distância ou de correlação. Uma alternativa análoga é comparar item a item para gerar uma predição baseado nos itens mais semelhantes. A estratégia por trás do ALS explícito consiste em estimar o efeito dos usuários e itens individualmente e em sequência, onde o resultado de uma passo serve como entrada para o passo seguinte. (BELL; KOREN, 2007)

Em modelos de fatorização de matrizes, a matriz esparsa de avaliações pode ser decomposta em duas matrizes, $U$ e $V$, sendo que cada usuário pode ser descrito em $d$ variáveis e cada item também é descrito em $d$ variáveis. Com o aprendizado dessas duas matrizes, basta multiplicar as duas e a matriz resultado será uma aproximação da avaliação de todos os usuários para todos os itens. Nesse caso, cada usuário $u$ é associado com os fatores de usuários $\mathbf{x}$ e cada item $i$ é associado aos fatores de itens $\mathbf{y}$.

Logo, a previsão da avaliação $\hat{r}_{ui}$ de um determinado usuário $u$ para um item $i$ pode ser dada como

$$\hat{r}_{ui} = \mathbf{x}_u \mathbf{y}_i^T = \sum_d \mathbf{x}_{ud} \mathbf{y}_{di},$$

e o erro $l_{ui}$ pode ser mostrado como

$$l_{ui} = r_{ui} - \mathbf{x}_u \mathbf{y}_i^T.$$

Logo, pode ser feita a minimação do erro quadrático, para o erro $L$ do modelo, através da fórmula

$$L = \sum_{u,i} \left(r_{ui} - \mathbf{x}_u \mathbf{y}_i^T\right)^2.$$

Para evitar o sobreajuste (*overfitting*) são adicionados dois termos de regularização $\mathbf{L}_2$, e a função objetivo se torna

$$L = \sum_{u,i} \left(r_{ui} - \mathbf{x}_u \mathbf{y}_i^T\right)^2 + \lambda \left( \sum_u \|\mathbf{x}_u\|^2 + \sum_i \|\mathbf{y}_i\|^2 \right),$$

onde $\lambda$ é um parâmetro que determina a importância do fator de regularização.

No caso do modelo com feedback implícito, como não há a ideia de avaliação, a variável $r_{ui}$ é binarizada ao ser transformada para a variável $p_{ui}$, que determina a preferência do usuário, onde

$$p_{ui} = \begin{cases} 1, & r_{ui} > 0 \\ 0, & r_{ui} = 0. \end{cases}$$

Por isso, em um cenário implícito não é possível ter a certeza sobre o gosto de um usuário, já que um valor $p_{ui} = 0$ pode indicar o desconhecimento desse item pelo usuário, ou mesmo indisponibilidade, e $p_{ui} > 0$ não necessariamente indica que esse usuário gosta e quer recomendações semelhantes desse item, já que o contexto dessa ação não foi levada em conta.

Para diferenciar os diferentes níveis de confiança, é adicionada uma variável $c_{ui}$, formada a partir de $r_{ui}$, onde um $r_{ui}$ maior indicará maior indicação de gosto. A variável da confiança sobre a recomendação do item $i$ para um usuário $u$, $c_{ui}$, pode ser descrita por

$$c_{ui} = 1 + \alpha r_{ui},$$

onde a constante $\alpha$ define a taxa de aumento da confiança e é similar à taxa de aprendizado presente em outros algoritmos de *machine learning*.

Por fim, a função perda a ser minimizada para o ALS Implícito é

$$L = \sum_{u,i} c_{ui} \left(p_{ui} - \mathbf{x}_u \mathbf{y}_i^T\right)^2 + \lambda \left( \sum_u \|\mathbf{x}_u\|^2 + \sum_i \|\mathbf{y}_i\|^2 \right)$$

Ao ser minimizada para computar os fatores de usuários, a partir da diferenciação para minimizar a função $L$, a expressão se torna

$$x_u = (Y^T C^u Y + \lambda I)^{-1} Y^T C^u p(u),$$

e para os fatores de itens

$$y_i = (X^T C^i X + \lambda I)^{-1} X^T C^i p(i),$$

como mostra Hu, Koren e Volinsky (2008).

O algoritmo para o treinamento é bem simples, primeiro é montada a matriz $C$, formada pelo parâmetro $\alpha$ multiplicado pelo conjunto de treinamento. Em seguida, são iniciados os fatores de usuário e itens aleatoriamente. Então, para cada interação, é calculado $X^TX$ e $Y^TY$ para diminuir o processamento. Para cada usuário é resolvida a função para $X$ com $Y$ fixado e, depois, para cada item é resolvida a função para $Y$ com $X$ fixado.

Uma vantagem desse algoritmo é o fato dele usar tanto os dados entre usuários quanto entre itens, ele tem uma alta explicabilidade, e isso permite a criação de *insights* e de explicações para o usuário final. Ele é bastante popular por sua velocidade e escalabilidade.

As principais desvantagens estão ao inserir novos itens e usuários no sistema, já que não há nada no algoritmo que ajude nessas previsões iniciais, além da necessidade do sistema ser rodado novamente em outras interações com novos itens.

## 2.5 Descrição de Métricas

Nesta seção são apresentadas as principais métricas dentro do contexto de sistemas de recomendação, com a motivação e a forma de cálculo de cada uma. Inicialmente são apresentadas aquelas que se referem à predição de uso, ou seja, o "acerto" do SR em comparação à base original, e em seguida àquelas de cobertura, que apresentam dados sobre a variedade de sugestões.

### 2.5.1 Predição de uso

Em muitos sistemas de recomendação o objetivo é de recomendar itens que serão interagidos pelos usuários. Em um cenário explícito, recomendar itens que o usuário 'goste' em detrimento de itens que o usuário 'não gosta' é o objetivo, são geradas métricas relacionadas ao erro médio ou acurácia, porém isso não é possível em cenários implícitos (RICCI et al., 2011).

Como é apenas possível verificar se esse item foi utilizado ou não, durante a fase de testes os itens são divididos entre recomendados e não-recomendados, e em usados e não usados. Com isso, pode se definir que uma recomendação pode ser um verdadeiro positivo (vp) quando ele foi recomendado e usado, falso positivo (fp) quando recomendado mas não usado, falso negativo (fn) quando não recomendado e usado e verdadeiro negativo (vn) quando não recomendado e nem usado.

Com isso, surgem as principais métricas para medir a acurácia de sistemas implícitos, a precisão e recall. A precisão é dada por: $\frac{tp}{tp+fp}$ e o recall por $\frac{tp}{tp+fn}$. Enquanto a

recomendação de menos itens pode diminuir o recall, elas podem aumentar a precisão e vice-versa.

Outras métricas que surgem para unir essas duas são a F-Measure, que é dada por

$$\frac{2 * recall * precision}{recall + precision}$$

e a AUC (*Area Under the ROC Curve*) que é a área formada em um gráfico construído a partir do recall e precisão. Durante o trabalho será levada em conta a métrica de *Recall@50*, que é o *recall* sobre 50 recomendações.

### 2.5.2 Cobertura

Um dos objetivos de sistemas de recomendação é que eles façam recomendações de itens que combinem com o usuário de forma mais única possível, para que usuários um pouco diferentes tenham recomendações um pouco diferentes e usuários muito diferentes tenham recomendações muito diferentes. Com isso, é interessante que o sistema consiga recomendar o máximo de itens diferentes possíveis.

Uma das métricas avaliadas é a de cobertura do catálogo, calculada pelo número de itens distintos sugeridos dividido pelo total de itens existentes. Para medir a variabilidade também será utilizado a Entropia de Shannon, que é dada por

$$-\sum_{i=1}^{n} p(i) \log p(i),$$

onde $p(i)$ determina a confiança, ou *score*, para um item $i$.

## 2.6 Trabalhos relacionados

Nesta seção são apresentados os principais métodos e artigos que utilizaram diferentes técnicas de sistemas de recomendação sobre a base de dados *Million Song Dataset* (MSD), que foi objeto de estudo para este trabalho.

### 2.6.1 Collaborative Metric Learning

Em 2017, foi publicado o *paper Collaborative Metric Learning* (HSIEH et al., 2017) que propôe uma nova abordagem para algoritmos do tipo *collaborative filtering*, onde o algoritmo desenvolvido, CML, calcula métricas de distâncias que buscam capturar os relacionamentos entre dados. No trabalho, foi estudada a conexão entre o aprendizado de métricas e a filtragem colaborativa.

O CML aprende a codificar as preferências dos usuários, além das similaridades entre itens e usuários, permitindo que seja utilizado um algoritmo do tipo *nearest-neighbor* para realizar as classificações.

Além disso, esse algoritmo também alcançou bons resultados em bases explícitas, onde ele não estima notas ou *ratings* e sim modelos entre os pares de usuários e itens. Essa abordagem ajuda na interpretabilidade, já que é possível identificar *clusters* de itens e usuários.

Para o MSD, esse algoritmo foi avaliado com as métricas de *Recall@50*, que avalia o *recall* de 50 recomendações por usuário, e *Recall@100*, que avalia para 100 recomendações. O *Recall@50* foi de $0.2460$ e o *Recall@100* foi de $0.3022$.

Neste trabalho, a base foi tratada de forma a manter apenas os usuários que escutaram apenas a 5 músicas, e foi levada em conta apenas a base que contém apenas as informações de código de usuário, código de item e quantidade de vezes que essa música foi ouvida por esse usuário. Não foram utilizadas outras *features*.

### 2.6.2 Multi-Variational Autoencoders

Em 2018, foi publicado o *paper Variational Autoencoders for Collaborative Filtering* (LIANG et al., 2018), que cria um modelo probabilístico não-linear a partir da extensão dos *Variational Autoencoders* (VAEs). Os *Autoencoders* "aprendem" uma representação dos dados, como uma redução de dimensionalidade, sendo uma rede neural que aprende a copiar a sua entrada para a saída, mapeando através de um *encoder*.

Os VAEs, no entando, são modelos geradores, que tentam simular como os dados são gerados e aprendem os relacionamentos entre os dados. No artigo, a abordagem parte da construção de um modelo gerador em redes neurais, como é usado em algumas abordagens linguísticas.

Além disso, os autores discorrem que, apesar de recomendações serem considerados um problema de *big data*, pelo volume de dados, eles a consideram um problema de *small data* já que os usuários só interagem com poucos dados.

Neste trabalho, o *Recall@50* foi de 0.363, avançando bastante o estado da arte naquele ano. A base foi tratada de forma a manter apenas usuários que escutaram pelo menos 20 músicas diferentes e as músicas escutadas por pelo menos 200 usuários.

### 2.6.3 Embarrassingly Shallow AutoEncoder

Foi publicado o artigo *Embarrassingly Shallow Autoencoders for Sparse Data* (STECK, 2019) em 2019, e ele também utiliza os VAEs nessa abordagem. No artigo, foi descoberto que redes neurais de poucas camadas podem atingir uma melhor acurácia. Nesse modelo,

o modelo da VAE não possui camadas intermediárias e a auto-similaridade entre a camada de entrada e saída é levada a zero, fazendo com que o modelo tenha que aprender a gerar uma saída de um item a partir dos outros itens.

Nesse caso, foi encontrado um *Recall@50* de $0.428$, o estado da arte para essa base de dados. O tratamento dos dados da base foi feito da mesma forma que para o *Multi-Variational Autoencoders*.

### 2.6.4 Outras abordagens

Existem diversas abordagens para essa base de dados. A abordagem mais básica é a recomendação por popularidade, onde, para todos os usuários, são recomendados os itens mais populares. Nesse caso, o *Recall@50* foi de $0.068$, como apontado em Steck (2019).

Uma abordagem clássica é a *Bayesian Personalized Ranking*, BPR, que é uma otimização para a geração de recomendações com ranking, da mais "recomendada" para a menos, diferente dos métodos para fatoração ou vizinhança. Esta técnica deriva de uma análise *Bayesiana* do problema. Nesse caso, o *Recall@50* atingido foi de $0.1246$ (HSIEH et al., 2017).

# 3 Metodologia

## 3.1 Materiais e métodos

Nesta seção são apresentados as principais fontes de dados utilizadas no desenvolvimento da pesquisa e as bibliotecas utilizadas.

### 3.1.1 Dados

Para os testes foi utilizada a base de dados Million Song Dataset (MSD), compilada por Bertin-Mahieux et al. (2011), que contém dados de identificadores de usuários, músicas, número de vezes que a música foi escutada e as letras das músicas, em formato de *bag-of-words*.

Essa base é composta por dados da empresa *Echo Nest*, uma plataforma de dados para música. Ela suge, inicialmente, com os dados de músicas e usuários, e apresenta algumas bases adicionais para expansão, como a base de letras de músicas que foi utilizada.

A base crua da MSD contém $1\,450\,932$ dados diferentes, contendo $110\,000$ usuários e $163\,206$ músicas, que leva a uma esparsidade de 99,991%. Por isso, apesar de ser uma base grande, ela contém uma quantidade muito maior de possíveis combinações do que em combinações existentes. Na seção 5 são apresentados os tratamentos dos dados realizados e uma análise exploratória dos mesmos.

### 3.1.2 Software

A linguagem escolhida para desenvolvimento foi Python devido à sua popularidade, à vasta disponibilidade de bibliotecas de inteligência artificial e por haver uma grande comunidade dessa área em atividade.

A biblioteca *scikit-learn* (PEDREGOSA et al., 2011) foi utilizada para o tratamento dos dados. Ela provê um vasto número de implementações de muitos algoritmos conhecidos de aprendizado de máquina, de forma eficiente. Também foi utilizada a biblioteca *pandas* (TEAM, 2020) para o carregamento e tratamento de dados.

Para o modelo de recomendação ALS (Alternating Least Squares), foi utilizada uma implementação feita pela biblioteca Implicit, que traz esse e outros algoritmos com bastante escalabilidade. (FREDERICKSON, 2020)

# 4 Descrição dos Experimentos

Os testes foram organizados por meio de cenários, onde os primeiros mostram os resultados isolados de cada algoritmo e abordagem utilizada, e, por fim, o terceiro e quarto utilizam técnicas de hibridização.

Todos os testes são realizados com diferentes parametrizações. Para cada teste, há a verificação das métricas de *Recall@50*, cobertura e entropia já apresentadas.

## Cenário 1: Apenas recomendação implícita

O primeiro cenário de testes utiliza apenas o algoritmo *Alternating Least Squares*, se utilizando apenas da base de dados esparsa entre os usuários e as músicas ouvidas.

## Cenário 2: Baseada em letras de músicas

No segundo cenário de testes, a base de dados utilizada é baseada em vizinhanças e construída a partir da base de letras de músicas disponível. Com isso, para cada usuário se construiu o "perfil médio de música", a partir do espaço de vetores das principais palavras nas letras.

## Cenário 3: Híbrido Mixed

No terceiro cenário há a hibridização entre o cenário implícito, que está apresentado no cenário 1, e o cenário baseado em conteúdos, apresentado no cenário 2. Nesse caso, cada um dos algoritmos faz a sugestão de alguns itens para cada usuário, e o híbrido utiliza os $K_1$ primeiros de um e $K_2$ primeiros do outro sistema, e remove possíveis duplicatas.

## Cenário 4: Híbrido Weighted

No quarto cenário, com hibridização Weighted, os dois sistemas de recomendação base são os apresentados no cenário 1 e 2, e os *scores* de cada um deles sofrem o processo de ponderação, para a sugestão de itens. São explorados diferentes parâmetros para os pesos nesse algoritmo.

# 5 Resultados e Discussão

## 5.1 Preparação dos dados

A primeira base de dados que foi carregada foi a de triplas entre as informações de usuário (`user_id`), itens (`item_id`) e a quantidade de vezes que esse usuário ouviu essa música (`listen_count`). Além disso, foi carregada a base de dados das músicas, que contém o identificador (`item_id`), nome do artista (`artist_name`) e nome da música (`song_name`). Juntando as duas bases, há a presença de $48\,373\,586$ triplas, mas são filtradas para manter apenas as músicas que foram ouvidas por mais de cinco vezes por um mesmo usuário e assim são mantidas $5\,010\,444$ triplas, ou seja, $10.35\%$, e estão presentes $227\,747$ músicas diferentes e $680\,078$ usuários diferentes.

Na base de letras de músicas, foram escolhidas as primeiras cem mil músicas. Para cada música, está presente um identificador da palavra (`word_id`) e o número de vezes que ela aparece na música. Também é carregada uma base de palavras, que contém as $5000$ palavras que mais aparecem, e as palavras que não estão nessa lista não estão presentes na base de letras de músicas.

A base conjunta de triplas é filtrada para conter apenas as músicas que estão nessa lista de músicas com letras. Então, sobram $1\,048\,346$ triplas. Por fim, são mantidos apenas os usuários que escutaram mais de cinco músicas distintas, mantendo $336\,148$ dados.

## 5.2 Análise exploratória

Com a base filtrada dessa forma, o número de usuários mantidos é de $37\,568$ e o número de músicas é de $30\,382$. A esparsidade dessa base filtrada fica em $99.90\%$, que é um valor melhor que a base original mas reforça o ponto citado por Liang et al. (2018), de que as bases se mostram como um problema de *small data*, onde os usuários não estão interagindo com tantos itens. Na imagem 1 está apresentado o formato dos dados das primeiras linhas da base conjunta de triplas e dados das músicas.

A média de vezes que uma música é ouvida por um usuário é de treze vezes, e o máximo de vezes é de $1862$ vezes. A imagem 2 apresenta um histograma que contém essa variação, limitada em cem vezes. A maioria dos usuários ouve entre dez e vinte músicas distintas.

Na distribuição de letras das músicas e palavras, as palavras mais comuns são os artigos e conjunções da língua inglesa, como apresentado na figura 3. Nela, a coluna

| user_id | song_id | listen_count | track_id | artist_name | song_name |
|---|---|---|---|---|---|
| 441c38e7acf1de6dc27ecdf47837b1b25cfeee83 | SOMZWUW12A8C1400BC | 6 | TRIDGUW128F9359DBC | Bobby Freeman | Do You Wanna Dance |
| 441c38e7acf1de6dc27ecdf47837b1b25cfeee83 | SOFHEYT12A8C14393D | 6 | TRGJUKS128F92D2B6C | The Orlons | South Street |
| 441c38e7acf1de6dc27ecdf47837b1b25cfeee83 | SOHWFLV12A8C140550 | 7 | TRFNVHG128F92D2F81 | The Toys | A Lover's Concerto |
| 441c38e7acf1de6dc27ecdf47837b1b25cfeee83 | SOKVKON12A6D4F6DC8 | 44 | TRIVYUZ128F1497CA1 | King's X | Dogman (LP Version) |
| 441c38e7acf1de6dc27ecdf47837b1b25cfeee83 | SOSVAPZ12A58A7E7A2 | 8 | TRAITRS128F9308142 | Marcie Blane | Bobby's Girl (Previously U |
| 441c38e7acf1de6dc27ecdf47837b1b25cfeee83 | SOTRXGD12A8C141BB1 | 6 | TRFVKHH128F92F13C8 | Jan & Dean | Dead Man's Curve |
| 9907ab4c1bed1f21325ea42cb4034c814fe16f6c | SOMZWUW12A8C1400BC | 15 | TRIDGUW128F9359DBC | Bobby Freeman | Do You Wanna Dance |
| 9907ab4c1bed1f21325ea42cb4034c814fe16f6c | SOTKYBW12A8C13C3EA | 11 | TRXUWEC128F426BE3F | Bon Iver | Skinny Love |
| 9907ab4c1bed1f21325ea42cb4034c814fe16f6c | SOAOAHZ12A8C13AAF1 | 16 | TRDKDVV128F42733CD | Enigma | Knocking On Forbidden D |
| 9907ab4c1bed1f21325ea42cb4034c814fe16f6c | SOMCWAZ12A67ADBCE3 | 10 | TRPKQEL128E0795771 | Zero 7 | In The Waiting Line |

Figura 1 – Primeiras linhas da base conjunta de triplas e informações das músicas

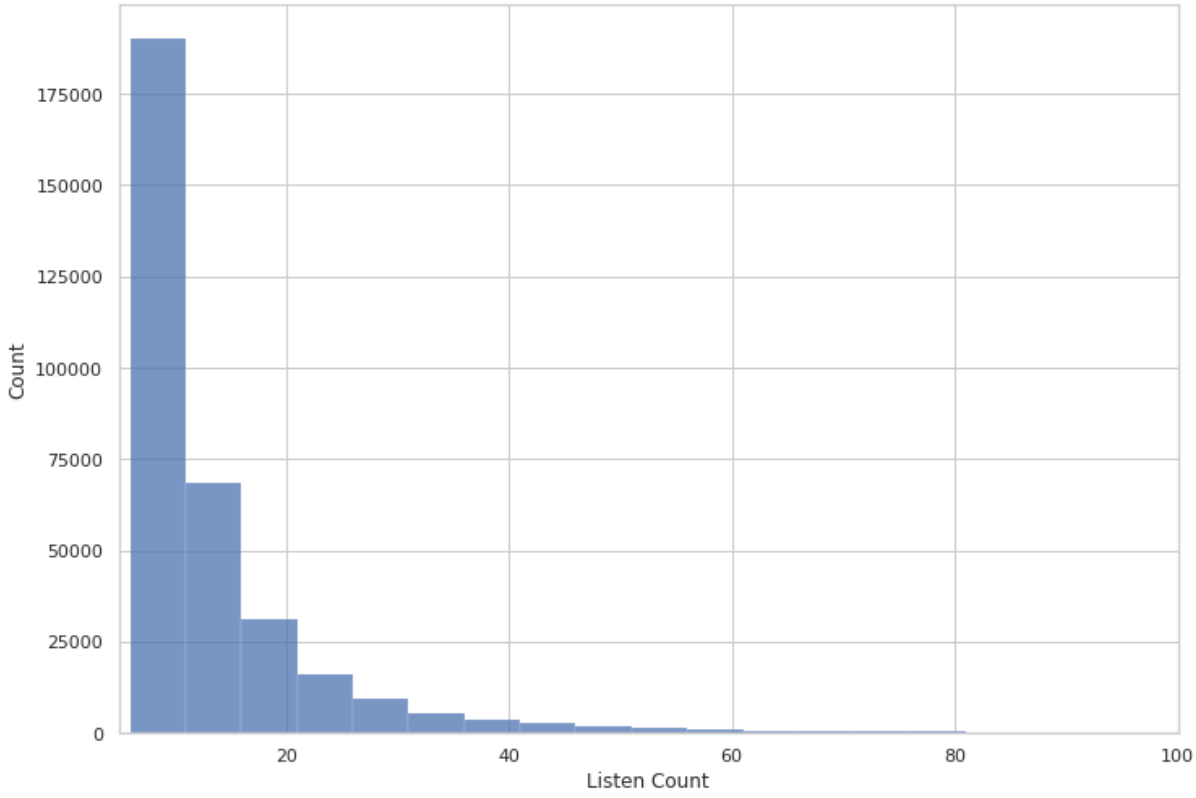


Figura 2 – Distribuição do número de vezes que um usuário ouve uma música

| | word_id | word_text | count_words | |
|---|---|---|---|---|
| | | | count | sum |
| **0** | 1 | i | 73307 | 872783 |
| **1** | 2 | the | 79543 | 790671 |
| **2** | 3 | you | 70226 | 730500 |
| **3** | 4 | to | 75692 | 449994 |
| **4** | 5 | and | 74177 | 442751 |
| **5** | 6 | a | 77072 | 405276 |
| **6** | 7 | me | 63446 | 322446 |
| **7** | 8 | it | 62997 | 346126 |
| **8** | 9 | not | 61228 | 309059 |
| **9** | 10 | in | 67690 | 265551 |

Figura 3 – Primeiras linhas da base de letras

| word_id | count_words | word_text |
|---|---|---|
| 3 | 27 | you |
| 9 | 15 | not |
| 1 | 13 | i |
| 2125 | 13 | handl |
| 70 | 13 | ca |
| 1117 | 10 | dare |
| 27 | 7 | love |
| 7 | 7 | me |
| 4 | 6 | to |
| 49 | 4 | they |
| 14 | 3 | your |
| 17 | 3 | on |
| 2770 | 3 | palm |
| 213 | 3 | show |
| 28 | 3 | so |
| 67 | 3 | take |
| 15 | 3 | that |
| 1290 | 2 | bore |
| 34 | 2 | just |
| 141 | 2 | then |

Figura 4 – Palavras mais recorrentes da música “5 Years”

*count* mostra o número de músicas em que essa palavra apareceu, enquanto a *sum* mostra o número total de vezes que essa palavra apareceu. Apesar das primeiras palavras não conterem muito significado quando olhadas sozinhas, após as 50 primeiras palavras, as seguintes são mais significativas e específicas a contextos, como verbos e substantivos.

Todas as palavras sofreram um processo de *tokeinização*, onde as palavras são tratadas e separadas, e são removidos plurais ou flexões verbais, e apenas a sua parte raiz é mantida. Um exemplo está na figura 4, que apresenta as vinte palavras mais recorrentes da música “5 Years”, da cantora islandesa Björk.

Nesse exemplo, pode se perceber que, apesar de haver palavras comuns da língua inglesa, também estão presentes outras palavras com várias repetições, como “handl(e)” e “dare”. Durante os experimentos, foram testadas versões que as cinquenta palavras mais comuns são ignoradas, as cem palavras mais comuns são ignoradas e testes em que nada é ignorado. Por fim, para mostrar a variedade de artistas presentes na base

```
artist_name
Dwight Yoakam            92097
Justin Bieber            75202
Björk                    72971
The Black Keys           58323
Five Iron Frenzy         52796
Bon Iver                 49375
Coldplay                 47614
Tub Ring                 38354
Metric                   33517
Jack Johnson             29794
Cage The Elephant        26320
Adam Lambert             22920
Kid Cudi                 21500
Florence + The Machine   21202
Orishas                  21134
Escape The Fate          20423
Brand New                20409
Rise Against             18933
Miley Cyrus              18083
Mike And The Mechanics   17953
Name: listen_count, dtype: int64
```

Figura 5 – Lista de artistas mais ouvidos

de dados estudada, são apresentados na figura 5 a lista de artistas mais ouvidos, com a contagem de vezes que esse artista foi ouvido no total.

## 5.3 Resultados dos experimentos

Nesta seção são apresentados e discutidos os resultados dos quatro cenários citados anteriormente. Para todos os cenários houve a separação da base de dados, após o tratamento descrito na seção anterior, em treino e teste, obedecendo a proporção de 75% dos dados para treino e o restante para teste. Essa separação foi feita entre as músicas que cada usuário ouviu, iterando por cada usuário e analisando quantas músicas ele ouviu. Então, 75% dessas músicas foram salvas na base de treino e o restante como teste; a seleção de quais músicas vão para cada base foi aleatória. Também foi feita a randomização da ordenação da base de dados, com as ordens dos usuários e das músicas sendo alterada.

### 5.3.1 Cenário 1: Apenas recomendação implícita

Para o cenário de recomendação implícita há uma fase bastante demarcada de aprendizado e outra de recomendação. Na fase de treinamento, após o tratamento da base e separação em base de treino e teste, os dados são colocados em formato de uma matriz esparsa $M$, em que os índices dos usuários ficam como colunas e os dos itens como linhas, ou vice-versa, e essa matriz guarda o número de iterações nas suas células. Esse formato é interessante pois ele é otimizado para os casos em que existem muitas células vazias, com valor zero.

Então, para o modelo ALS estão disponíveis as alterações dos parâmetros de números de iterações, o valor de $\lambda$ para as regularizações $L2$, o número de fatores $d$ para os vetores de usuário e itens após a fatoração da matriz. Também foi explorado o parâmetro $\alpha$, que altera a confiança por trás de cada item. Após o treinamento desse modelo ele está pronto para fazer as recomendações, basta passar a matriz esparsa sobre

| Número de Fatores | Valor de $\alpha$ | Recall@50 | Cobertura | Entropia |
|---|---|---|---|---|
| **32 fatores** | **1** | 0.2438 | 4.69% | 6.5913 |
| | **10** | 0.2989 | 10.63% | 7.0510 |
| | **40** | 0.3027 | 15.78% | 7.2752 |
| | **100** | 0.2950 | 19.57% | 7.4021 |
| | **200** | 0.2915 | 22.60% | 7.4939 |
| | **1000** | 0.2733 | 28.97% | 7.6851 |
| **64 fatores** | **1** | 0.2713 | 8.39% | 7.0281 |
| | **10** | 0.3334 | 17.93% | 7.4537 |
| | **40** | 0.3392 | 26.74% | 7.6598 |
| | **100** | 0.3406 | 34.47% | 7.7906 |
| | **200** | 0.3335 | 39.99% | 7.8910 |
| | **1000** | 0.3160 | 45.55% | 8.0519 |
| **128 fatores** | **1** | 0.2979 | 14.33% | 7.5031 |
| | **10** | 0.3437 | 31.09% | 7.9572 |
| | **40** | 0.3473 | 48.13% | 8.2267 |
| | **100** | 0.3428 | 56.06% | 8.3703 |
| | **200** | 0.3378 | 58.55% | 8.4516 |
| | **1000** | 0.3192 | 60.25% | 8.5535 |

Tabela 1 – Resultados para o cenário 1

as quais serão feitas as recomendações. Foram feitas as análises do *Recall@50*, cobertura e entropia com diversos cenários com alterações dos parâmetros.

A alteração no fator de regularização $\lambda$ não causou nenhuma mudança significativa em qualquer uma das métricas, por isso ele não foi apresentado nos resultados, assim como o número de iterações que não contribuiu para uma mudança de desempenho, e também não foram encontrados no modelo um número de iterações que o leva a um *overfitting*. Para o número de iterações, foram feitos testes com a variação entre 50 iterações à 2000 iterações, enquanto para o fator de regularização foram feitas variações entre $0.05$ e $0.5$. A tabela 1 apresenta um compilado dos resultados a partir da variação dos outros parâmetros, mantendo o número de iteração em $500$ e o valor de $\lambda$ em $0.02$.

É possível observar que os resultados deste cenário foram bastante positivos, ultrapassando com facilidade o *Recall@50* do recomendador dos itens mais populares e até mesmo CLB, mostrando que o algoritmo conseguiu entender os perfis de usuário e fazer recomendações relevantes. A sua cobertura chegou a $48.13\%$ no cenário com o melhor *Recall@50*, o que mostra que foram feitas recomendações muito diversas, que é um fator bastante positivo. Apesar disso, a entropia não variou muito, mostrando que, independente dos parâmetros, existem alguns itens que são muito recomendados em detrimento de outros, já que em um cenário de distribuição uniforme a entropia seria igual à zero.

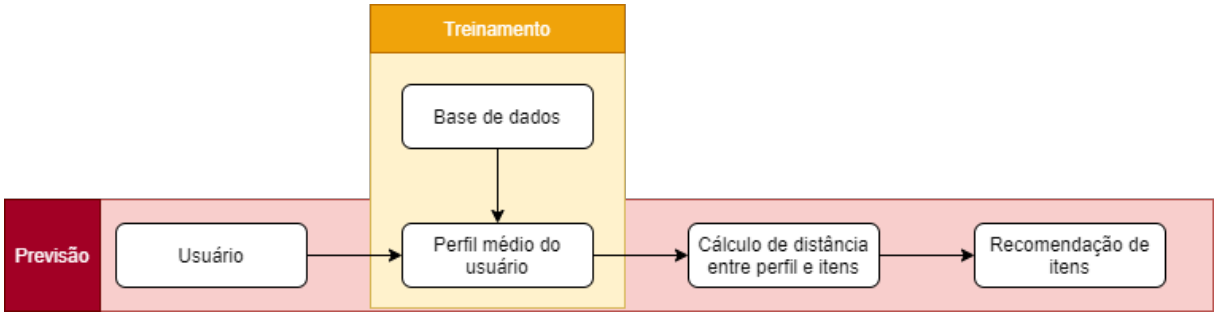


Figura 6 – Arquitetura do cenário 2

### 5.3.2 Cenário 2: Baseada em letras de músicas

Neste experimento, após o tratamento da base e separação entre treino e teste, é montado o perfil médio de cada usuário. Para isso, para cada usuário, são somadas as frequências das palavras presentes nas músicas que ele escutou e estão na base de treino, de forma proporcional à quantidade de vezes que ele escutou. Também são feitas normalizações em cada música, para que a soma das frequências de cada palavra na música seja igual à 1 e, ao final, também é feita uma normalização no vetor final de perfil médio do usuário.

Feita essa fase inicial, as recomendações são feitas para cada usuário segundo o seu perfil. Esse perfil é comparado com todas as músicas presentes na base de dados e as $k$ com maior similaridade são as recomendações, para fins de avaliação as recomendações de músicas presentes na base de treinamento desse usuário são ignoradas. Essa similaridade é o inverso de uma medida de distância, onde uma distância próxima à zero indica que o item analisado e o perfil do usuário estão muito próximos e essa seria uma boa recomendação. Nos testes, foram analisadas a performance das métricas de distância do cosseno, euclidiana, correlação e Hamming.

A figura 6 apresenta um resumo dessa arquitetura. Um ponto interessante dessa arquitetura *nearest neighbout* é que a fase de treinamento pode ocorrer junto à de teste, onde o perfil do usuário é calculado, ou atualizado, conforme a demanda de recomendações.

Para os testes, foram feitos testes com a não utilização de todas as palavras na base de letras, para manter apenas aquelas mais significativas e verificar se há uma alteração nos resultados. Para isso, foi feito um cenário sem remoção de palavras, um com a remoção das 50 mais comuns e outro com a remoção das 100 mais comuns. Os resultados para os testes estão apresentados na tabela 2

É possível verificar que, nos testes, a distância de Hamming obteve um *Recall@50* bastante superior aos outros casos. Essa distância, geralmente aplicada em contextos textuais, leva em conta a quantidade de modificações que um vetor tem que sofrer para ficar igual aquele com que ele é comparado. Essa métrica é bem interessante para o

| | | Recall@50 | Cobertura | Entropia |
|---|---|---|---|---|
| **Com todas as palavras** | **Cosseno** | 0.0016 | 36.92% | 6.6576 |
| | **Euclidiana** | 0.0000 | 0.97% | 3.9560 |
| | **Correlação** | 0.0016 | 37.28% | 6.6728 |
| | **Hamming** | 0.0233 | 29.24% | 5.7732 |
| **Sem as 50 palavras mais comuns** | **Cosseno** | 0.0066 | 63.80% | 8.2754 |
| | **Euclidiana** | 0.0000 | 1.18% | 4.0266 |
| | **Correlação** | 0.0066 | 66.12% | 8.3393 |
| | **Hamming** | 0.0233 | 28.41% | 5.7158 |
| **Sem as 100 palavras mais comuns** | **Cosseno** | 0.0100 | 76.33% | 8.5331 |
| | **Euclidiana** | 0.0000 | 1.03% | 4.0755 |
| | **Correlação** | 0.0100 | 77.42% | 8.5561 |
| | **Hamming** | 0.0216 | 28.26% | 5.6781 |

Tabela 2 – Resultados para o cenário 2

problema já que ela não leva em conta as diferenças nas frequências de cada palavra, e sim se as mesmas palavras aparecem ao se comparar duas músicas, por exemplo.

As métricas de cosseno e correlação trouxeram resultados muito parecidos, porém abaixo da distância de Hamming. Isso pode ser explicado pelas letras de duas músicas, mesmo que falando de um mesmo tema, tem frequências distintas de aparecimento de cada palavra, e isso impacta nessas métricas. Esse mesmo ponto aparece ao se observar a euclidiana, que não parece adequada para esse problema, já que mudanças grandes de frequência causam muito impacto.

Apesar do *Recall@50* desse cenário não trazer valores tão interessantes, já que está abaixo do caso em que são feitas recomendações apenas dos itens mais populares, ela traz valores interessantes de cobertura, que mostra que grande parte da base de dados está sendo ativamente recomendada, especialmente quando são removidas algumas das palavras comuns. A entropia também diminui ao serem consideradas menos palavras, o que significa que as recomendações estão menos focadas em 1 ou outro exemplo e sim com uma distribuição mais equilibrada, no caso de uso da distância de *Hamming*.

Por tanto, a inclusão do cenário com as letras das músicas foi considerado positivo, já que não era esperado um valor de *Recall@50* alto, pelo fato dos usuários não ouvirem, necessariamente, músicas dos mesmos temas e com as mesmas letras, mas o algoritmo atingiu o objetivo de fazer recomendações variadas de músicas com temas semelhantes àquelas presentes no perfil do usuário, permitindo que sejam feitas recomendações diversas e propiciando a descoberta de novos artistas e sonoridades por parte do usuário, mantendo os seus temas de interesse.

| Recomendações Implícitas | Recomendações por letras | Recall@50 | Cobertura | Entropia |
|---|---|---|---|---|
| **10** | **40** | 0.2009 | 29.90% | 6.7076 |
| **25** | **25** | 0.2930 | 28.71% | 7.7188 |
| **40** | **10** | 0.3341 | 26.81% | 7.9601 |

Tabela 3 – Resultados para o cenário 3

### 5.3.3 Cenário 3: Híbrido Mixed

Para o cenário híbrido do tipo *mixed*, as arquiteturas dos dois cenários citados anteriormente são unidas, de forma que cada um dos algoritmos faz $k$ recomendações e elas são apresentadas ao usuário em conjunto. Para os testes, foram utilizados os testes com melhor *Recall@50*, no caso do cenário 1 é aquele com 128 fatores e $\alpha$ igual a 40 e para o cenário 2 é aquele que ignora as 50 palavras mais comuns e utiliza a distância de Hamming.

Nesse cenário os parâmetros analisados foram em questão do número de recomendações que cada cenário realiza. Na tabela 3 estão apresentados os resultados dos experimentos.

É possível notar que, apesar do *Recall@50* não chegar ao valor do cenário 1 original, os resultados se mantiveram satisfatórios, mostrando que o implícito conseguiu manter bons resultados mesmo fazendo menos recomendações. Além disso, a cobertura e entropia melhoraram em comparação ao cenário 1, mostrando que o cenário 2 pode funcionar bem para trazer sugestões secundárias de novas músicas que o usuário pode se interessar, mas que não necessariamente são o que usuários parecidos com ele costumam ouvir.

### 5.3.4 Cenário 4: Híbrido Weighted

Para o último cenário foi analisado um caso híbrido em que cada algoritmo do cenário 1 e 2 fazem recomendações e, então, ponderando as confianças dos dois recomendadores, a partir de pesos pré-definidos, são feitas as recomendações finais para o usuário. Assim como no cenário 3, para esse foram utilizados os experimentos com o maior *Recall@50*. Essa arquitetura está apresentada na figura 7.

Nesse caso foram analisados três variações de pesos entre o cenário implícito e aquele baseado em letras. Como o cenário baseado em letras tem como métrica de similaridade a distância $dist$ entre o perfil médio e os itens, para o experimento foi considerado o *score* como $1 - dist$, já que no cenário implícito o score que tem valores próximos à 0 é considerado pouco relevante. Um problema que foi verificado durante a análise dos *scores* é de que o cenário implícito gerava resultados com confiança muito maior, onde as 10 primeiras recomendações giravam em torno de $0.9$, já aquelas do

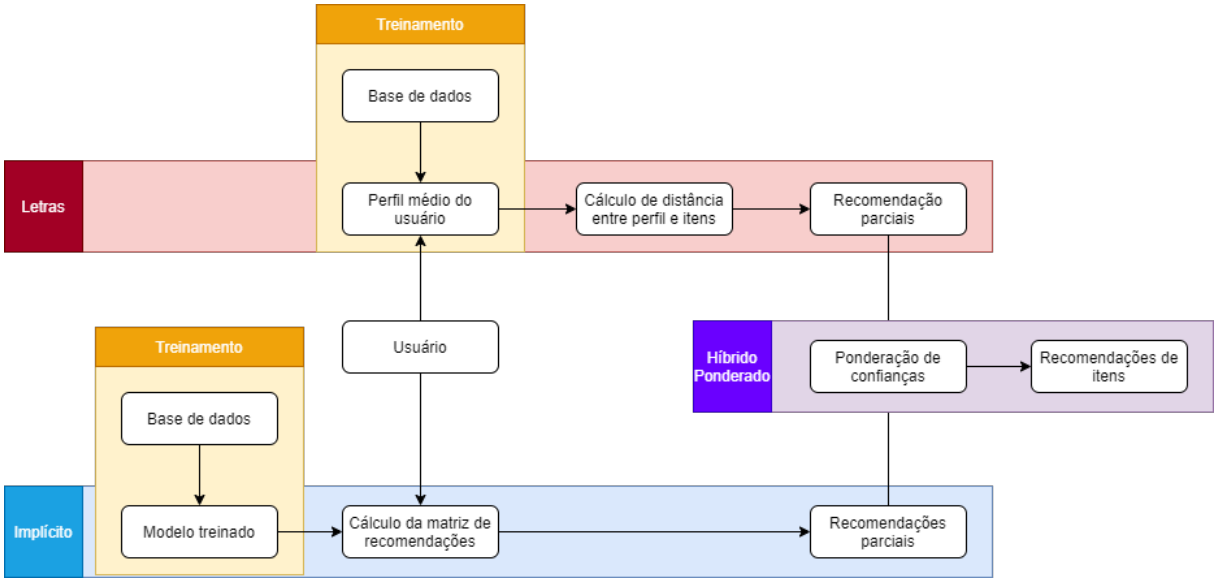


Figura 7 – Arquitetura do modelo híbrido *weighted*

| Peso de recomendações Implícitas | Peso de recomendações por letras | Recall@50 | Cobertura | Entropia |
|---|---|---|---|---|
| **0.1** | **0.9** | 0.0652 | 27.41% | 5.9214 |
| **0.5** | **0.5** | 0.1540 | 23.41% | 6.8869 |
| **0.9** | **0.1** | 0.3098 | 16.51% | 7.8123 |

Tabela 4 – Resultados para o cenário 4

cenário baseado em letras gira em torno de $0.55$ e isso afetou os resultados para esse cenário 4.

Na tabela 4 estão apresentados os resultados para esse experimento. Nos casos extremos, onde o peso de um dos cenários base é de $0.1$ o resultado fica bastante próximo ao do cenário original, sem um ganho claro de acurácia ou variedade. No caso intermediário, onde os dois possuem pesos iguais, o resultado parece estar intermediário mas, ainda assim, em questão de *Recall@50*, inferior à qualquer experimento feito para o cenário 1 e sem ganho em cobertura ou entropia.

Essa técnica de hibridização, por tanto, não trouxe resultados tão satisfatórios quanto a técnica anterior. Por conta da forma que os dois modelos tratam a confiança, o modelo do cenário 2 não consegue levar recomendações interessantes e acaba por apenas reduzir a performance que o cenário 1 teria.

# 6 Conclusões

O trabalho atual permitiu analisar as principais técnicas para construção de sistemas de recomendações em contexto implícito, apurando principalmente a hibridização como ferramenta de mitigação de pontos negativos que uma ou outra abordagem pode trazer.

Ao realizar experimentos com o uso de uma base de dados de hábitos musicais e de letras de músicas, foi possível analisar os resultados que os diferentes sistemas obtiveram, tanto em questão de acurácia quanto em versatilidade. Com os sistemas híbridos construídos foi possível notar que este conseguiu manter a acurácia de um sistema implícito, ao mesmo tempo que a versatilidade foi aumentada pelo sistema baseado em *features* não tradicionais.

Levando em conta as duas técnicas de hibridização que foram utilizadas nos testes, a *mixed* obteve resultados mais interessantes, já que os dois algoritmos base conseguiram realizar recomendações dentro do esperado. Entretanto, esse resultado não foi alcançado no cenário *weighted*, já que nele a disparidade da confiança dos seus algoritmos componentes ficou clara, de forma que a ponderação dos *scores* apagou os pontos positivos daquele com menor peso e não trouxe os ganhos esperados.

Pela abordagem feita nesse trabalho, de que as características de interesse de um sistema de recomendação vão além da acurácia, um sistema de recomendação híbrido *mixed* é uma opção válida para sistema em contextos implícitos, que é voltado para facilitar o descobrimento de novos itens por parte do usuário.

Também é possível observar que os resultados encontrados podem ser aplicados em outros meios, além das recomendações musicais que foi um suporte para a aplicação das técnicas desenvolvidas. Um próximo passo seria trazer outras classes de algoritmos para o desenvolvimento de híbridos mais robustos, explorando outros conjuntos de dados.

Outra abordagem interessante é de trazer mais métricas para a análise, e construir uma avaliação centrada no usuário para os sistemas híbridos, avaliando a contribuição de cada algoritmo base em cada métrica.

# Referências

AMATRIAIN, X.; BASILICO, J. Past, present, and future of recommender systems: An industry perspective. In: Proceedings of the 10th ACM Conference on Recommender Systems. New York, NY, USA: Association for Computing Machinery, 2016. (RecSys '16), p. 211–214. ISBN 9781450340359. Disponível em: <https://doi.org/10.1145/2959100.2959144>. Citado 2 vezes nas páginas 1 e 5.

BELL, R. M.; KOREN, Y. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In: IEEE. 7th IEEE International Conference on Data Mining (ICDM 2007). [S.l.], 2007. p. 43–52. Citado na página 10.

BERBATOVA, M. Overview on NLP techniques for content-based recommender systems for books. In: Proceedings of the Student Research Workshop Associated with RANLP 2019. Incoma Ltd., 2019. Disponível em: <https://doi.org/10.26615/issn.2603-2821.2019_009>. Citado na página 6.

BERTIN-MAHIEUX, T. et al. The million song dataset. In: Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011). [S.l.: s.n.], 2011. Citado na página 16.

BURKE, R. Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction, p. 2, 2002. Citado na página 2.

BURKE, R. Hybrid web recommender systems. The Adaptive Web, p. 377–408, 2007. Citado na página 3.

BURKE, R. Hybrid web recommender systems. In: The adaptive web. [S.l.]: Springer, 2007. p. 377–408. Citado na página 6.

FREDERICKSON, B. Fast python collaborative ltering for implicit datasets. 2020. Disponível em: <https://github.com/benfred/implicit>. Citado na página 16.

HSIEH, C.-K. et al. Collaborative metric learning. In: Proceedings of the 26th international conference on world wide web. [S.l.: s.n.], 2017. p. 193–201. Citado 2 vezes nas páginas 13 e 15.

HU, Y.; KOREN, Y.; VOLINSKY, C. Collaborative filtering for implicit feedback datasets. In: IEEE. 2008 Eighth IEEE International Conference on Data Mining. [S.l.], 2008. p. 263–272. Citado 2 vezes nas páginas 5 e 12.

KNIJNENBURG, B. P. et al. Explaining the user experience of recommender systems. User Model User-Adap Inter, 2012. Citado na página 2.

LEUNG, C. W.; CHAN, S. C.; CHUNG, F.-l. Integrating collaborative filtering and sentiment analysis: A rating inference approach. In: Proceedings of the ECAI 2006 workshop on recommender systems. [S.l.: s.n.], 2006. p. 62–66. Citado na página 6.

LIAN, J. et al. Xdeepfm: Combining explicit and implicit feature interactions for recommender systems. In: Proceedings of the 24th ACM SIGKDD International

Conference on Knowledge Discovery & Data Mining. New York, NY, USA: Association for Computing Machinery, 2018. (KDD '18), p. 1754–1763. ISBN 9781450355520. Disponível em: <https://doi.org/10.1145/3219819.3220023>. Citado na página 1.

LIANG, D. et al. Variational Autoencoders for Collaborative Filtering. 2018. Citado 2 vezes nas páginas 14 e 18.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, v. 12, p. 2825–2830, 2011. Citado na página 16.

POMMERANZ, A. et al. Designing interfaces for explicit preference elicitation: a user-centered investigation of preference representation and elicitation process. User Modeling and User-Adapted Interaction, Springer Science and Business Media LLC, v. 22, n. 4-5, p. 357–397, mar. 2012. Disponível em: <https://doi.org/10.1007/s11257-011-9116-6>. Citado na página 5.

PU, P.; CHEN, L.; HU, R. A user-centric evaluation framework for recommender systems. In: Proceedings of the fifth ACM conference on Recommender systems. [S.l.: s.n.], 2011. p. 157–164. Citado na página 7.

RICCI, F. et al. Recommender Systems Handbook. [S.l.]: Springer, 2011. Citado 3 vezes nas páginas 2, 3 e 12.

SARWAR, B. et al. Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web. [S.l.: s.n.], 2001. p. 285–295. Citado na página 9.

STECK, H. Embarrassingly shallow autoencoders for sparse data. The World Wide Web Conference on - WWW '19, ACM Press, 2019. Disponível em: <http://dx.doi.org/10.1145/3308558.3313710>. Citado 2 vezes nas páginas 14 e 15.

TEAM, T. pandas development. pandas-dev/pandas: Pandas. Zenodo, 2020. Disponível em: <https://doi.org/10.5281/zenodo.3509134>. Citado na página 16.

WILSON, T.; WIEBE, J.; HWA, R. Recognizing strong and weak opinion clauses. Computational Intelligence, v. 22, 2006. Citado na página 6.