UNIVERSIDADE FEDERAL DE SANTA CATARINA

Aplicação de PLN na recomendação de conteúdo educacional

Marcelo Muller Vieira Filho

Florianópolis
2024

Marcelo Muler Vieira Filho

Aplicação de PLN na recomendação de conteúdo educacional

Trabalho de Conclusão de Curso submetido como parte dos requisitos para obtenção do Título de Bacharel do Curso de Sistemas de Informação, da Universidade Federal de Santa Catarina.

Orientador: Elder Rizzon Santos.

Florianópolis, Junho de 2024

Marcelo Muller Vieira Filho

Aplicação de PLN na recomendação de conteúdo educacional

Trabalho de Conclusão de Curso apresentado como parte dos requisitos para a obtenção do Título de "Bacharel em Sistemas de Informação", e aprovado em sua forma final pelo Curso de Sistemas de Informação da Universidade Federal de Santa Catarina.

Florianópolis, Junho de 2024

Prof. Álvaro Junio Pereira Franco
Coordenador do Curso

Banca examinadora:

Prof. Elder Rizzon Santos
Orientador

Prof. Roberto Willrich
Membro da Banca

Prof. Thiago Ângelo Gelaim
Membro da Banca

Dedico este trabalho a todos que sempre me apoiaram e me deram forças.

"All we have to decide is what to do with the time that is given us."

(J. R. R. Tolkien)

Resumo

Muitos serviços contam com seu próprio sistema de recomendação, que essencialmente apresenta às pessoas produtos ou serviços que elas possam querer consumir, em uma lista determinada utilizando diferentes técnicas, que tem como base dados prévios de consumo. É possível observar isso ao receber sugestões de novos filmes em um serviço de streaming ou sugestões de novos produtos em uma loja online. O PLN, ou processamento de linguagem natural, uma vertente da Inteligência Artificial que atua na assistência da compreensão de dados gerados por humanos pelas máquinas, é uma ferramenta útil e poderosa para captar os gostos e preferências do usuário. No mundo acadêmico, os alunos se encontram muitas vezes perdidos com tanto conteúdo educacional disponível e não conseguem traçar uma rota de estudos clara e bem definida para atingir seus objetivos, como estudante ou profissional, em meio a essa infinidade de materiais. Utilizando técnicas de PLN, o trabalho se propõe a delinear um modelo de SR alimentado por uma base de dados de usuários que recomende materiais aos alunos de acordo com o perfil do usuário.

Palavras-chave: Processamento de linguagem natural, Sistema de recomendação, Conteúdo educacional.

Abstract

Several services have their own recommender system, which essentially presents people products or services that they might want to consume, in a determined list utilizing different techniques, which is based on previous consumption data. This can be seen by receiving suggestions for new movies on a streaming service or suggestions for new products on an online store. NLP, or natural language processing, a branch of Artificial Intelligence that assists in machine understanding of human-generated data, is a useful and powerful tool for capturing user's tastes and preferences. In the academic world, students often find themselves lost with so much educational content available and are unable to chart a clear and well-defined study path to achieve their goals, as a student or professional, amidst this plethora of materials. Using NLP techniques, this work proposes to outline a RS model fed by a user database that recommends items according to the user's profile.

Keywords: Natural language processing, Recommender system, Educational content.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

SR	Sistema de Recomendação
PLN	Processamento de Linguagem Natural
ML	*Machine Learning*
HTTP	*Hypertext Transfer Protocol*
JSON	*JavaScript Object Notation*
REST	*Representational State Transfer*
TCC	Trabalho de Conclusão de Curso
UFSC	Universidade Federal de Santa Catarina

Sumário

1 Introdução

É inegável a evolução avassaladora da tecnologia nos últimos anos. O mundo de vinte, trinta anos atrás é praticamente outro. E o impacto desse crescimento tecnológico atinge todos os setores, visto que é algo sentido em praticamente todos os âmbitos da sociedade. Dentre as maiores mudanças observáveis, pode ser apontada uma, do ponto de vista socioeconômico, como a responsável por guiar a maioria das pesquisas que visam o desenvolvimento de novas tecnologias para o futuro: a ascensão do objeto de maior valor atualmente, os dados. A grande questão é o que fazer com esses dados e como tratá-los.

Um dos desafios da computação moderna é fazer com que máquinas compreendam a linguagem humana. As aplicações são inúmeras quando o computador consegue entender, interpretar e manipular a linguagem natural do homem. O PLN, processamento de linguagem natural, existe justamente para atuar como elo entre a comunicação humana e a inteligência artificial. Tradução, resultados de buscas, predição de texto, filtros de e-mail e assistentes virtuais são apenas alguns exemplos que vemos diariamente da aplicação desse conceito. Segundo [Torfi et al. 2020] a importância do PLN como uma ferramenta de assistência na compreensão de dados gerados por humanos é uma lógica consequência da dependência contextual de dados. Os dados tornam-se mais significativos à medida que o contexto é explorado, o que facilita a análise textual e mineração. O PLN permite isso com as estruturas de comunicação e padrões humanos.

O marketing, unido com a exploração inteligente dos dados, é a mina de ouro da atualidade. Todo serviço que precisa manter as pessoas em constante estado de consumo conta com um sistema de recomendação, entre outras palavras, “se você gostou deste, tente este outro”. Streaming e redes sociais são os exemplos mais conhecidos e mais fáceis de citar, como o sistema de recomendação de novos filmes da Netflix e o do TikTok - um dos aplicativos mais utilizados no mundo - mas é possível perceber que praticamente tudo que é vendido na Internet conta com um sistema desse tipo. A Amazon e outras grandes lojas já possuem uma funcionalidade mostrando o que pessoas com compras e visitas parecidas no site estão comprando no momento.

Em frente a uma grande quantidade de conteúdo disponível, as pessoas muitas vezes se encontram perdidas e empregam muito tempo escolhendo um conteúdo dentre o leque infinito de possibilidades, como explicam [Aamir e Bhusry 2015]. Por essa e várias outras razões os sistemas de recomendação são uma necessidade no contexto atual do mundo. Um sistema de recomendação utiliza da combinação de técnicas de Machine learning (aprendizagem de máquina) e uma base de dados fortemente alimentada, porque é necessário que o algoritmo aprenda e se desenvolva a partir dos gostos, interesses e

características de cada usuário e faça suposições sobre isso. De acordo com [Lu et al. 2015], os SR são programas que tentam recomendar os itens (serviços ou produtos) mais relevantes para determinados usuários (empresas ou indivíduos), utilizando da predição do interesse do usuário em um item, baseado em informações relacionadas aos itens, usuários e interações entre itens e usuários. O objetivo é direcionar o conteúdo ao usuário com base no que mais condiz com seu perfil.

Machine learning, de acordo com [Brown 2021], é uma subárea da inteligência artificial, amplamente definida como a capacidade de uma máquina imitar o comportamento da inteligência humana. Sistemas de inteligência artificial são usados para realizar complexas tarefas de uma maneira similar à maneira que humanos resolvem problemas.

Hoje em dia existem diversas soluções que utilizam PLN em suas funcionalidades, como o IBM Watson, Amazon Comprehend, MonkeyLearn e Aylien [Wolff 2020]. O PLN pode ser inserido em vários contextos para resolver problemas como análise de sentimento em textos online, transformação de texto em fala e detecção de urgência em textos.

Em meio a tantos materiais disponíveis gratuitamente por toda a Internet, o acadêmico, hoje em dia, por muitas vezes se encontra perdido e sem uma rota de estudos traçada para atingir seus objetivos. Isso é ainda mais forte na área de computação, em que o profissional deve manter-se sempre atualizado com o que está mais forte no mercado e procurar se especializar em determinados assuntos e tecnologias, ao invés de tentar entender de tudo. Qual linguagem de programação, framework, ferramenta ou tecnologia aprender e direcionar seu foco e esforços é uma incógnita para muitos.

O objetivo do trabalho é propor e treinar um modelo de recomendação utilizando técnicas de PLN e uma base de dados para fazer sugestões e recomendar material didático disponível online.

1.1 Objetivos

1.1.1 Objetivo Geral

O objetivo deste TCC é aplicar técnicas de PLN para treinar um modelo de sistema de recomendação de conteúdos educacionais *online*, com possibilidade de aplicação em livros ou cursos *e-learning*, para alavancar o aprendizado de usuários que procuram orientação numa infinidade de materiais disponíveis na Internet.

1.1.2 Objetivos Específicos

Os objetivos específicos são os seguintes:

- Analisar estado da arte de SRs que utilizam PLN;;
- Definir os requisitos para uma solução desse contexto;
- Obter uma fonte de dados sobre cursos e material didático;
- Extrair e tratar os dados obtidos com técnicas de PLN;
- Desenvolver um sistema de recomendação com base nesses dados;
- Delinear um experimento para avaliar o protótipo do SR;
- Analisar e avaliar o SR com base no experimento.

1.2 Metodologia

A metodologia de pesquisa e desenvolvimento deste trabalho dividiu-se nas etapas:

- **Etapa 1:** Pesquisa bibliográfica e levantamento dos assuntos necessários para desenvolver a aplicação, investigando o estado da arte em sistemas de recomendação, em sua maioria os que utilizam PLN. A pesquisa foi realizada em artigos, documentação das ferramentas, *surveys* e outras publicações relacionadas ao tema;
- **Etapa 2:** Análise e definição dos requisitos necessários para a construção de uma solução dessa natureza. Também foram consideradas pesquisas acerca das tecnologias a serem utilizadas e diferentes técnicas para atuarem em todas as etapas da recomendação;
- **Etapa 3:** Pesquisa sobre bases de dados robustas e relevantes ao tema, para que possa alimentar o SR e gerar recomendações precisas.
- **Etapa 4:** Extração e tratamento dos dados obtidos pelos *datasets* com técnicas de PLN;
- **Etapa 5:** Desenvolvimento de um protótipo do SR com base nos dados;
- **Etapa 6:** Análise e avaliação do SR a partir de um experimento.
- **Etapa 6:** Documentar e relatar todo o procedimento do trabalho para as disciplinas de Introdução a Projetos, Projetos 1 e Projetos 2.

1.3 Estrutura do Trabalho

O presente trabalho se divide em 7 capítulos. Este primeiro apresenta a introdução do trabalho, informações sobre o problema, os objetivos e a metodologia a ser seguida.

No capítulo 2 são apresentados os conceitos básicos necessários para o entendimento e desenvolvimento da solução proposta.

No capítulo 3 é realizada uma análise dos trabalhos relacionados e uma discussão ao final, sobre as ferramentas e decisões de projeto mais relevantes a este trabalho.

O capítulo 4 apresenta o SR proposto do trabalho, com seu tipo de filtragem, funcionamento do sistema detalhado e todos os principais processos, fornecendo uma base para o entendimento da ideia do sistema ideal escolhido para ser construído.

Os capítulos 5 e 6 trazem o processo do desenvolvimento de cada parte do SR proposto, começando pelo colaborativo e seguindo para o baseado em conteúdo, com todo o funcionamento detalhado e ilustrado, apresentando cada parte do processo, e, ao final de

cada capítulo, a parte de experimentos onde se realizam testes para verificar o desempenho de cada um dos SR desenvolvidos para compor a solução propostaAo final, o capítulo 7 conclui o trabalho, apresentando considerações finais e as linhas de raciocínio para explicar o desempenho do sistema construído, apontando as falhas e possíveis causas e sugerindo melhorias para possíveis trabalhos futuros.

2 Fundamentação Teórica

Este capítulo lista os conceitos necessários para o entendimento e o desenvolvimento do trabalho. Serão apresentados alguns assuntos relevantes a sistemas de recomendação e processamento de linguagem natural, as bases do presente trabalho.

2.1 Sistemas de Recomendação

Algoritmos com o objetivo de predizer a relevância de um item e, com base nisso, realizar sugestões ao usuário são os chamados sistemas de recomendação. Essas sugestões podem partir de características de itens similares ou de avaliações feitas por usuários com perfis parecidos com o do usuário em particular. A maioria dos serviços prestados na Internet, hoje em dia, conta com seu próprio sistema de recomendação. *Streaming* (Netflix, HBO Max e Disney+), plataformas de vídeos (YouTube), *e-commerce* (Amazon) e redes sociais (Instagram).

Esses sistemas funcionam baseados em algoritmos de *machine learning* que fazem a extração de uma base de dados, tratam os dados e definem padrões de comportamento de usuários. Podem ser classificados quanto ao tipo de filtragem utilizado, seja baseada em conteúdo, colaborativa ou híbrida.

2.1.1 Filtragem baseada em conteúdo

A filtragem baseada em conteúdo utiliza a similaridade de vetores para recomendar itens a partir de features, ou seja, características extraídas da base de dados, para uma determinada entrada desejada.

Há um processo de examinação do perfil de um novo usuário quanto a seus interesses, baseado nas funções disponíveis nos objetos que o usuário avaliou. É um sistema de recomendação para palavras-chave específicas. Esse tipo de filtragem utiliza algoritmos que recomendam itens semelhantes aos que o usuário gosta [Balush, Vysotska e Albota 2021].

Filtragem baseada em conteúdo é um método de obtenção de informação que usa características de itens para selecionar e retornar itens relevantes a uma consulta do usuário. Este método muitas vezes toma características de outros itens em que o usuário expressou interesse [Murel e Kavlakoglu 2024].

2.1.2 Filtragem colaborativa

No caso da filtragem colaborativa, a similaridade é calculada baseada em preferências de usuários com perfis e gostos parecidos para, após reunir várias informações, realizar recomendações de determinados itens sem que o usuário tenha interagido com qualquer um daqueles itens.

Sistemas de recomendação que sumarizam avaliações de itens ou recomendações, reconhece semelhanças entre usuários, baseado em suas avaliações, e gera novas recomendações baseadas em comparações entre usuários. Funciona bem no caso de objetos complexos, onde variações de gosto são responsáveis pela maior parte das mudanças de preferências. Essa filtragem é baseada na situação de assumir que pessoas que concordaram no passado, decidirão no futuro [Balush, Vysotska e Albota 2021].

Esta abordagem aproxima grupos de usuários a grupos distintos baseando-se em seus comportamentos, o que algoritmos como o KNN (*K-nearest Neighbors*) fazem, retornando os itens mais próximos com base em um número determinado.. Usando características gerais dos grupos, ela retorna itens específicos para um grupo inteiro pelo princípio de que usuários similares se interessam por itens similares [Murel e Kavlakoglu 2024]

2.1.3 Filtragem híbrida

Os sistemas híbridos unem a exploração de dados de interação entre usuários e itens (e metadados de usuários e itens). Assim, o sistema híbrido é o melhor considerado a lidar com diversos tipos de problemas com os quais os outros tipos de filtragem têm dificuldades, já que pode contar com as informações de metadados quando não há dados de interações disponíveis [Vita e Cioffi 2022].

Combinando ambos os tipos de sistema, essa filtragem fornece recomendações comparando os hábitos de consumo e pesquisa de usuários semelhantes, assim como gera recomendações de itens com características semelhantes a itens bem avaliados pelos usuários [Balush, Vysotska e Albota 2021].

A seguir há uma tabela comparativa dos três tipos de filtragem, com vantagens e desvantagens:

Nome	Vantagens	Desvantagens	Aplicação
Filtragem colaborativa	Avaliações de usuários são levadas em conta	Performance baixa para primeiros usuários	Portais de informação
	Não é presa à área do serviço em questão	Necessita de muita informação de avaliações de usuários	Pequenas lojas *online*
Filtragem baseada em conteúdo	Instantânea, mesmo para primeiros usuários	Ligada ao conteúdo do serviço em questão	Blogs
Filtragem híbrida	Funciona bem, mesmo com poucos dados	Não se baseia em desejos do usuário	Plataformas de música e filmes
	Alta produtividade	Difícil de manter	Grandes lojas *online*
	Sem os problemas das demais filtragens	Processo de desenvolvimento complexo	Sistemas complexos com muitos usuários

Tabela 1 – Diferenças entre os tipos de filtragens [Balush, Vysotska e Albota 2021]

2.2 Processamento de Linguagem Natural

O processamento de linguagem natural é uma área dentro da Inteligência Artificial que atua como uma ponte entre a linguagem natural, isto é, a linguagem que os humanos entendem, para a linguagem entendida pelo computador. O PLN é responsável por extrair informações a partir do contato com usuários e permitir que a máquina compreenda o que foi escrito pelos seres humanos.

Entre os usos do PLN é possível citar as aplicações que dependem que o contexto seja explorado e entendido perfeitamente, como interpretação de textos, análise de sentimentos, tradução de acordo com significado semântico, etc. Plataformas de busca, como o Google, utilizam diversas técnicas de PLN para entender rapidamente o que o usuário está querendo dizer ao buscar um termo ou uma frase. As assistentes virtuais, muito fortes hoje em dia, e cada vez mais se tornando parte da vida das pessoas, realizam tarefas e atendem a comandos diariamente aplicando PLN em seu funcionamento [Education 2020].

2.3 *Machine learning*

Machine learning é uma subárea da Inteligência Artificial, amplamente definida como a capacidade de uma máquina imitar comportamentos inteligentes do ser humano. Sistemas de IA são usados para realizar tarefas complexas de uma maneira similar à que humanos resolvem problemas.

No *machine learning* os dados são coletados e preparados para serem usados como dados de treinamento ou as informações nas quais o modelo de aprendizado de máquina será treinado. Quanto mais dados, melhor o programa [Brown 2021].

Suas aplicações são diversas, como os próprios sistemas de recomendação, carros com piloto automático (*Tesla*), algoritmos de análise de imagens. Muito forte e presente na tomada de decisões, é uma grande aliada de diversas empresas atualmente, principalmente durante as decisões mais arriscadas, no direcionamento de operações de negócios e nos possíveis caminhos que a empresa pode tomar [Kanade 2022].

2.4 K-nearest Neighbors

É um algoritmo de classificação de aprendizado supervisionado muito utilizado na área de *machine learning* e *data mining* para resolver problemas de classificação e regressão. Ele utiliza métricas de distância entre vetores para classificar grupos de dados.

A classificação por *nearest neighbor*, também conhecida por KNN, é baseada na ideia de que os padrões mais próximos de um padrão X alvo, o qual procura-se classificar, entrega informações úteis de classificação. O KNN atribui o rótulo da classe à maioria dos K mais próximos padrões em dados dimensionais [Kramer 2013].

2.5 Matrizes item-item e usuário-item

São matrizes cujas dimensões podem representar usuários e itens ou itens e itens. Em cada registro entre um usuário ou item x com um item y há uma pontuação em relação à interação entre eles, seja uma avaliação ou uma revisão. Exemplo:

	item 1	item 2	item 3
usuário 1	4		1
usuário 2		2	3
usuário 3	3	4	3

Tabela 2 – Representação de uma matriz usuário-item

2.6 Fatoração de matriz

Fatoração de matriz é um modelo matemático que ajuda o sistema a dividir uma entidade em múltiplas entradas menores, por meio de um conjunto retangular de números ou funções ordenadas, a fim de descobrir *features* ou informações ocultas nas interações entre usuários e itens [Raman 2024].

É uma técnica extensivamente utilizada em SRs de filtragem colaborativa. Seu objetivo é fatorar uma matriz usuário-item em duas matrizes de ranque menor, a matriz de usuário-fator e a matriz de item-fator, para que possam predizer novos itens nos quais os usuários possam se interessar [Kumar 2021].

3 Trabalhos Correlatos

Com o objetivo de realizar uma busca por trabalhos que tratassem de sistemas de recomendação utilizando PLN e relacionados, foi utilizada a ferramenta Google Scholar, um indexador de artigos científicos. Para levantar o estado-da-arte, assim como outras soluções relevantes ao tema, foram utilizados termos de busca como *"recommender systems nlp"*, *"recommendation systems nlp"*, *"recommendation machine learning nlp"* e *"recommendation systems"*. Dentre os artigos encontrados, foram avaliados e selecionados os três melhores e mais relacionados com o presente trabalho para a primeira seção, e outros três servindo como mais exemplos de trabalhos com relação ao tema.

3.1 Hybrid Movie Recommender System

Em sua tese, [Vita e Cioffi 2022] decidem resolver o problema da quantidade massiva de informação disponível *online* com um sistema de recomendação híbrido, defendendo que a forma como eles funcionam pode melhorar muito a qualidade das recomendações com relações usuário-item muito mais complexas e melhorias no problema de partida a frio dessas relações, comparando com os métodos baseados em conteúdo e filtragem colaborativa. Para isso, utilizando *"The Movies Dataset"* disponível no *Kaggle*[1] e após analisar o estado-da-arte dos algoritmos de recomendação, treinaram, avaliaram e compararam diversos modelos de recomendação, um deles utilizando a abordagem baseada em conteúdo, um modelo de filtragem colaborativa construído por fatoração matricial e matriz de interações e um modelo híbrido explorando ambos pontuação e metadados usuário-item. Também conduziram experimentos utilizando LightFM[2], uma biblioteca que utiliza um método de recomendação híbrida, uma tipologia especial de recomendação que usa ambas as filtragens, colaborativa e baseada em conteúdo, para gerar recomendações. O autor investiga como a informação em linguagem natural pode contribuir para melhorar a recomendação de itens de acordo com as preferências do usuário, fazendo um estudo sobre o estado-da-arte de técnicas de PLN para a engenharia textual de recursos, a partir de modelos livres de contexto, como o TF-IDF[3] e o Word2Vec[4], até modelos baseados em transformadores, como o BERT[5] e GPT-2[6]. O trabalho teve uma investigação e implementação de técnicas de determinação de similaridade de documentos e modelagem de tópicos para treinar um modelo híbrido de recomendação, capaz de sugerir com precisão filmes baseados em interações passadas

[1] https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset
[2] https://github.com/lyst/lightfm
[3] *Term Frequency Inverse Document Frequency*
[4] https://www.tensorflow.org/tutorials/text/word2vec
[5] *Bidirectional Encoder Representations from Transformers*
[6] *Generative Pre-trained Transformer 2*

do usuário, mas também utilizando metadados de itens em sua vantagem. As sinopses dos filmes são utilizadas como pontos em um espaço multidimensional usando o BERT e categorizadas em tópicos com o HDBScan, agrupando para que seja computado pelo motor de recomendação. Além disso, são realizados experimentos de avaliação do modelo híbrido sobre o problema de partida a frio de itens e usuários, configurações específicas de recomendação para usuários novos ou itens sem interações passadas.

Sobre as vantagens do sistema de filtragem colaborativa, os autores apontam a forma como a comparação da similaridade com a matriz resultante é muito mais escalável, especialmente em datasets maiores e mais esparsos. Além disso, não é necessário conhecimento sobre o domínio porque as incorporações no sistema são aprendidas automaticamente apenas pela matriz de interação, entre outros motivos.

O problema de partida a frio, apontado pelos autores, consiste na situação em que há novos itens ou novos usuários com poucas ou nenhuma interação dentro do modelo com filtragem colaborativa, e que por esse motivo a performance é fraca.

Segundo [Vita e Cioffi 2022], os sistemas híbridos de recomendação combinam os modelos baseados em conteúdo e de filtragem colaborativa em maneiras que suas vantagens se complementem. O *Netflix* é apontado como um ótimo exemplo. O *streaming* gera recomendações comparando os comportamentos de busca e visualização dos usuários (a parte da filtragem colaborativa), e também sugere filmes que compartilham das qualidades de um filme bem avaliado pelo usuário (a parte baseada em conteúdo). Os sistemas híbridos unem a exploração de dados de interação entre usuários e itens e metadados de usuários e itens. Assim, o sistema híbrido é o melhor considerado a lidar com o problema da partida a frio, já que pode contar com as informações de metadados quando não há dados de interações disponíveis.

Também utilizam técnicas de PLN para *embedding* de sentenças. A ideia é vetorizar os enredos dos filmes e torná-los comparáveis e processáveis pelos sistemas de recomendação. Foi utilizado TF-IDF no contexto da modelagem de tópicos do BERT para extrair significados de vários *clusters*. Depois da extração dos *embeddings* do BERT nas sentenças, foram concatenados os documentos pertencentes ao mesmo *cluster* para entender seu significado semântico e avaliar a qualidade do agrupamento. O TF-IDF foi aplicado em cada um dos documentos, extraindo as palavras mais relevantes. O Word2Vec é uma rede neural que pode determinar se palavras são similares, opostas ou se um par de palavras A possui a mesma relação entre si de um par de palavras B. O BERT, que foi utilizado no trabalho, é um modelo que leva em consideração o contexto, podendo classificar vetores considerados de mesma representação por outros modelos como vetores diferentes. Os *embeddings* fornecidos pelo BERT são contextualizados para cada palavra diferente, de acordo com a sentença, sendo considerado bidirecional, ou, segundo o autor, a melhor palavra para descrevê-lo seria não-direcional. Essa característica permite que o modelo

aprenda o contexto da palavra baseado no que está ao seu redor, à direita e à esquerda.

Para explorar metadados de itens na recomendação híbrida, os metadados textuais foram representados e agrupados como pontos em um espaço *embedding* multidimensional, utilizando modelagem de tópicos com o BERT. O objetivo era que os enredos comparáveis dos filmes fossem agrupados juntos para que os tópicos correspondentes fossem encontrados.

O conjunto de dados contém metadados sobre 45466 filmes. Cada filme possui 24 atributos. Cada filme pode ser de 20 gêneros diferentes, e cada um pode estar listado entre mais de um gênero. Há também dados sobre a produtora, país de origem, idioma, data de lançamento, etc.

A limpeza dos dados, assim como o pré-processamento, é feita em grande parte por técnicas PLN. Muitos dos valores do conjunto de dados eram obtidos por objetos JSON. Eles são transformados em strings para uma mais fácil extração de palavras-chave e valores. Algumas características são removidas por não serem muito relevantes ou terem muitos *outliers*. O coeficiente de Pearson checa a correlação entre pares de atributos numéricos, já que eles poderiam afetar negativamente o processo de recomendação. As razões para a exclusão dos atributos são de pouca ou nenhuma informação intrínseca, muitos valores inválidos, muitos valores únicos, alta correlação com outros atributos e redundância. Para a engenharia de características, gênero, data de lançamento, popularidade, entre outros atributos, diversas técnicas foram utilizadas para selecionar e manipular os dados disponíveis nas características utilizadas na recomendação.

Foram utilizados quatro diferentes modelos de recomendação:

1. Modelo baseado em conteúdo puro, utilizando somente os metadados de itens de filmes com interações de usuários para gerar recomendações topK;

2. Modelo de filtragem colaborativa LightFM, um modelo de fatoração matricial simples para explorar interações usuário-item ou avaliações para gerar recomendações topK;

3. Modelo híbrido LightFM com metadados de item, um modelo colaborativo de fatoração matricial que explora interações usuário-item e características de filmes;

4. Modelo híbrido LightFM com metadados de item e usuário, um modelo colaborativo de fatoração matricial que explora interações usuário-item e características de filmes e usuários.

O modelo 1 possui três principais passos de funcionamento:

1. Geração do vetor do perfil de usuário com média ponderada pela avaliação dos vetores de filmes assistidos;

2. Comparação entre o vetor e de outros filmes;

3. Pontuação dos topK filmes recomendados.

O modelo 2 foi alimentado com registros de informação sobre interações entre usuário e item, como *id* de usuário, *id* de filme, etc. O modelo 3 utilizou várias características de filmes. O modelo 4 teve um misto de características de itens e usuários, sendo as de itens gêneros preferidos do usuário, atores mais assistidos, diretor mais assistido, etc.

O experimento é conduzido após o pré-processamento, separando o *dataset* final em treinamento e teste, numa proporção 7:3. Cada usuário, em média, teria 100 filmes no conjunto de treinamento e 49 no conjunto de teste. Para avaliar a qualidade da divisão, foi utilizada a distância média, com similaridade de cosseno, entre os vetores de usuários no treinamento e no teste, após a divisão. Após a análise, foi verificado que a divisão preservou bem as preferências de usuários, pois a média de similaridade sempre era abaixo de 0,6. As métricas da avaliação levaram em consideração a relevância do filme. Um filme é considerado relevante para o usuário se ele o tivesse visto e revisado com uma nota acima de um determinado número. O número é obtido a partir de um valor fixo arbitrário ou derivado dinamicamente dos dados. É importante notar que se um usuário avaliou o filme com um valor igual ou maior ao valor da média, o filme seria considerado relevante para ele. Ao realizar uma recomendação para um usuário, filmes que já foram vistos não são levados em conta, já que afetariam o resultado negativamente.

A métrica para precisão utilizada foi Precision@K, definindo a fração de itens relevantes em meio aos itens obtidos:

$$p = \frac{\#\text{ relevant recommendations}}{\#\text{ recommended items}}$$

Figura 1 – Definição da precisão [Vita e Cioffi 2022]

Onde o denominador é chamado K. A precisão diz o quanto o modelo consegue recomendar itens relevantes em meio às topK recomendações. O Precision@K foca somente no topo do *ranking* de recomendações, onde não importa o resto do *ranking*, contanto que os primeiro K itens sejam, em maioria, positivos. O MAP (*Mean Average Precision*) é a média da precisão entre os usuários para um dado valor K.

Para o *recall*, ou sensibilidade, é utilizado o Recall@K. O *recall*, que geralmente define a fração dos itens relevantes que são recomendados com sucesso:

Ou seja, é a probabilidade de um item relevante ser obtido na consulta. Acima, o denominador representa o número de itens relevantes dos dados de teste para um dado

$$r = \frac{\#\text{ relevant recommendation}}{\#\text{ relevant items}}$$

Figura 2 – Definição do recall [Vita e Cioffi 2022]

usuário. O MAR (*Mean Average Recall*) é a média do *recall* entre os usuários para um dado valor K.

O F1-Score é utilizado com a ideia de que *recall* e precisão devem ser avaliados em par, garantindo uma igual contribuição de ambos:

$$F1 = 2\,\frac{p\,r}{p+r}$$

Figura 3 – Definição do F1-Score [Vita e Cioffi 2022]

Como uma média harmônica para a precisão e o *recall*. O melhor valor para a métrica é 1 e o pior é 0.

A métrica AUC é a probabilidade de um filme positivo (assistido e avaliado pelo usuário) ter uma pontuação maior que um filme negativo. O valor máximo é 1, significando que não há filmes negativos mais altos no ranking do que um filme positivo. O AUC serve para avaliar a qualidade geral do *ranking*.

É realizada validação cruzada para escolher os melhores hiperparâmetros e avaliar a robustez da aleatoriedade das divisões, para garantir que a performance do conjunto de teste não seja afetada pelo tipo de divisão. Com os modelos LightFM, é aplicada a busca aleatória sobre os hiperparâmetros. Os melhores são escolhidos por melhor F1-Score, para melhor precisão e *recall*. Para os modelos híbridos, deve-se escolher os metadados de itens e usuários mais informativos. Uma busca exaustiva sobre as possíveis combinações é feita, escolhendo os com melhor F1-Score.

Os resultados são gerados com diversos valores de K, sendo K o número de recomendações por usuário, com cada modelo, com o conjunto de treinamento e teste, utilizando a mesma divisão. As métricas utilizadas são MAP e *recall*, F1-Score e AUC.

O modelo baseado em conteúdo apoia-se em metadados de itens e avaliações de usuários para os vetores de perfil de usuário. Para cada K e *id* de usuário, o vetor é computado com a média ponderada, com as avaliações como peso. O vetor é então comparado com todos os filmes disponíveis: o número total de filmes, menos os já assistidos pelo usuário. O modelo gera, então, a recomendação. Dadas essas recomendações, foram computados o MAP e o *recall*.

K	MAP	MAR	F1-Score
3	0.2960	0.0491	0.0362
5	0.2718	0.0734	0.0522
7	0.2522	0.0942	0.0585
9	0.2415	0.1116	0.0647
11	0.2327	0.1305	0.0671
13	0.2224	0.1442	0.074
15	0.2136	0.1566	0.0728
17	0.2076	0.1704	0.0740
19	0.2003	0.1828	0.0765
21	0.1946	0.1941	0.0774

Tabela 3 – MAP e MAR do modelo LightFM de filtragem baseada em conteúdo

Analisando os resultados, os autores percebem que somente os metadados de itens não são suficientes para prover uma recomendação confiável que seja precisa e sensível. Aproximadamente, um em dez filmes recomendados são realmente vistos pelos usuários e o modelo não consegue um *recall* maior que 7% dos filmes do conjunto de testes.

O modelo de filtragem colaborativa explora a fatoração matricial com os dados de interações, levando em consideração a magnitude das interações. Para cada divisão do *dataset*, uma busca aleatória é feita para obter os hiperparâmetros. Os melhores são mostrados na tabela:

K	MAP	MAR	F1-Score
3	0.2960	0.0491	0.0842
5	0.2718	0.0734	0.1155
7	0.2522	0.0942	0.1371
9	0.2415	0.1116	0.1526
11	0.2327	0.1305	0.1672
13	0.2224	0.1442	0.1749
15	0.2136	0.1566	0.1807
17	0.2076	0.1704	0.1871
19	0.2003	0.1828	0.1911
21	0.1946	0.1941	0.1943

Tabela 4 – MAP e MAR do modelo LightFM de filtragem colaborativa

O modelo alcançou 1 de 3 recomendações corretas em média (30% MAP) e 20% de *recall*. Após a análise, concluiu-se que o modelo perde a capacidade de generalização conforme o K aumenta.

Os modelos híbridos tinham em comum os metadados de itens levados em consideração. Algumas características de filmes foram obtidas a partir da seleção feita.

K	MAP	MAR
3	0.3341	0.0541
5	0.2925	0.0801
7	0.2827	0.0988
9	0.2653	0.1332
11	0.2501	0.1451
13	0.2483	0.1579
15	0.2344	0.1678
17	0.2289	0.1802
19	0.2209	0.2001
21	0.2175	0.2198

Tabela 5 – MAP e MAR do modelo LightFM de filtragem híbrida

O modelo híbrido teve uma performance muito superior ao colaborativo, especialmente em precisão. Ele se provou menos provável a ter *overfit* nos dados de treino. A única desvantagem encontrada em inserir metadados de itens no LightFM foi o aumento no tempo de treinamento e predição. Também foi inspecionado o quanto o tamanho do *dataset* influenciava na performance dos dados de teste. O Precision@K mostrou direta proporcionalidade com o crescimento dos dados de treino, ou seja, quanto mais filmes um usuário assistisse e avaliasse, melhor seria a acurácia das recomendações. Foi realizado então o teste com ambos metadados de itens e usuários, mas não se obteve uma mudança considerável nos resultados. No fim, o modelo com os dois tipos de metadados não superou o modelo que utilizava apenas os de itens.

Em conclusão, o autores reafirmam que o propósito da tese era investigar tipos diferentes de sistemas de recomendação e construir um sistema híbrido capaz de explorar os pontos mais positivos deles, tentando eliminar as desvantagens que eles trazem. Também foi um objetivo estudar como a linguagem natural poderia ser explorada para aumentar a qualidade das sugestões e melhorar diversos problemas.

3.2 *Recommendation System for Students' Course Selection*

O trabalho de [Naren, Banu e Lohavani 2020] tem como objetivo ajudar estudantes de graduação a escolherem novas disciplinas a cada semestre, levando em conta suas próprias capacidades, tendências do mercado e outros fatores, utilizando para isso técnicas de processamento de linguagem natural e mineração de dados, em conjunto com scripts de *frontend* e *backend* escritos em Python e hospedados na Internet. *Data mining*, também, segundo os autores, popularmente conhecida como *knowledge discovery from data* (KDD), foi utilizada para coletar e analisar dados de estudantes sobre cursos finalizados junto com suas notas. O PLN, ou processamento de linguagem natural, foi utilizado na mineração e tratamento textual para o entendimento da máquina, com técnicas como tokenização, remoção de *stop words*, stemização, lematização, entre outras.

O modelo proposto por [Naren, Banu e Lohavani 2020] conta com a construção de uma plataforma para os estudantes realizarem a escolha dos cursos, baseada em disciplinas anteriores e notas obtidas. As notas indicam o nível de aptidão do estudante para uma disciplina. É feito um cálculo da similaridade entre as disciplinas finalizadas e as que podem ser escolhidas. As cinco primeiras do ranking com a maior similaridade são tomadas e o valor cumulativo de nota delas é calculado. A partir disso, as duas disciplinas com o maior valor calculado são recomendadas ao estudante. Foi utilizado o *framework* de código aberto Flask, que é um serviço *web* para o Python, com requisições HTTP feitas por meio de API REST e *template* HTML tratado com Jinja 2.

O *dataset* utilizado foi fornecido pelo departamento de Ciência da Computação e Engenharia da *SASTRA University*, formado por 45 cursos. As disciplinas foram tratadas como dados a serem inseridos na parte do *frontend*, pelo usuário, sendo tratadas junto com sua descrição no *backend*. São dados de 9205 disciplinas com categoria, código, título, descrição, professor, etc. Porém, na análise da similaridade descrita acima são usados o título e a descrição da disciplina.

A tokenização é utilizada para quebrar os parágrafos das descrições de disciplinas em palavras. Então, a remoção de *stop words* é feita para retirar palavras que não serão usadas na filtragem do texto e isolar as palavras-chave, para simplificar e otimizar o processo. A lematização busca reduzir palavras à sua forma base, seu radical, para

encontrar resultados relacionados com essas palavras. A vetorização alterou dados textuais para a forma numérica, como vetores.

Após a formação do modelo vetorial, é calculada a similaridade entre disciplinas utilizando os coeficientes de Sørensen–Dice e cosseno:

$$\text{Similarity} = \cos(\emptyset) = \frac{\mathbf{A}.\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

Figura 4 – Similaridade de cosseno [Naren, Banu e Lohavani 2020]

A e B são vetores do documento. Aplicando-se o coeficiente de Dice:

$$s = \frac{2n_t}{n_x + n_y}$$

Figura 5 – Similaridade de Dice [Naren, Banu e Lohavani 2020]

Aqui, "s" é o coeficiente, nt o número de palavras bigramas do documento x, e ny é o número de palavras bigramas do documento y.

Na aplicação web, as informações de pré-requisitos de disciplinas e notas - convertidas para inteiros - são introduzidas pelo usuário para que se realize o processo de cálculo da similaridade e coeficientes. O resultado, então, é retornado e mostrado em tela. O programa é separado em três funções:

1. Primeira função: Pré-processamento das descrições das disciplinas. Remoção das *stop words* e tokenização. O texto é convertido para letras minúsculas e é realizada a stemização em cada palavra;

2. Segunda função: Realização do cálculo do coeficiente de Sørensen–Dice. O arquivo é dividido em listas bigramas ordenadas para encontrar palavras correspondentes em ambos os documentos para as duas disciplinas escolhidas, e então uma pontuação é calculada baseada nisso;

3. Terceira função: A função principal consiste em passar argumentos para as outras duas funções e a contagem das palavras é feita no dicionário. Com ele, os valores TF-IDF de vetorização são calculados, criando um modelo de estado vetorial para o documento. Dentro dessa lista de vetores, valores de similaridade obtidos pelo método do cosseno são calculados e retornados para a interface gráfica.

De acordo com os autores, os coeficientes cosseno e Sørensen–Dice obtiveram os

melhores resultados, dentre os demais, dado o contexto e o uso específico em documentos textuais.

Table 77.1 Cosine and Dice coefficients for the first set of subjects

Subjects	C	Signals and systems	Unix	Java	Matlab	Software engineering
Cosine	37.25	4.44	30.71	8.88	38.5	28.49
Dice	87.69	77.78	85	90.48	87.69	94.29

Table 77.2 Cosine and Dice coefficients for the second set of subjects

Subjects	Compiler	Structure of computing	Data science	Operating systems	Computational thinking
Cosine	37.25	4.44	30.71	8.88	38.5
Dice	87.69	77.78	85	90.48	87.69

Figura 6 – Resultados obtidos pelo modelo proposto por [Naren, Banu e Lohavani 2020]

A figura, que representa as tabelas 77.1 e 77.2, mostra os resultados de uma amostra teste da computação dos coeficientes de similaridade, de diferentes tipos, verificando a similaridade entre 11 disciplinas com a disciplina de Python. Comparando todas as formas de medida, como já previamente constatado, os coeficientes de Dice e cosseno obtiveram os melhores resultados, sendo Dice declarado o melhor. Outras formas de medida como distância Euclidiana, distância Manhattan e coeficiente de Pearson não se encaixaram no contexto do sistema e foram desconsideradas.

3.3 *Using Natural Language Processing Techniques to Provide Personalized Educational Materials for Chronic Disease Patients in China*

Utilizando materiais educacionais para instruir os pacientes chineses disponíveis na Internet, [Wang et al. 2020] apresenta um estudo com a finalidade de criar um sistema de recomendação na área da saúde para fornecer material educacional apropriado para pacientes com doenças crônicas na China, e avaliar os efeitos do sistema. O método utilizado foi por meio de uma ontologia e várias técnicas de processamento de linguagem natural.

Os autores utilizam diversas fontes para reunir os materiais educacionais para os pacientes e um *dataset* de pacientes coletado de um sistema de telessaúde. O objetivo é projetar e implementar um sistema capaz de descobrir as necessidades potenciais dos pacientes e lhes recomendar material de estudo relevante.

A parte principal do processo de recomendação é uma ontologia chamada *Chronic Disease Patient Education Ontology* (CDPEO), que descreve características de pacientes

para geração de recomendação. Os dados de pacientes e dos materiais educacionais são convertidos em vetores de mesmo tamanho que serão comparados e terão sua similaridade checada ao final. Os dados de pacientes são convertidos por meio de uma abordagem baseada em regras, já os dados de materiais são convertidos por meio de uma abordagem baseada em PLN.

O desenvolvimento do estudo é dividido em três etapas. Na primeira, a ontologia é construída para descrever as características dos pacientes contidas nos dados. Na segunda, é delineado e implementado um algoritmo para gerar recomendações baseadas na ontologia. Dados de pacientes e material educacional são mapeados para a ontologia e convertidos em vetores de mesmo tamanho, gerando recomendações de acordo com a similaridade entre os vetores. Já na etapa 3, a ontologia e o algoritmo são introduzidos em um sistema móvel de saúde para praticidade. Algoritmos de extração de palavras-chave e incorporações de palavras pré-treinadas são utilizados para o pré-processamento dos materiais educacionais.

A ontologia é construída da seguinte maneira:

- Definição do domínio e escopo por meio de perguntas às quais a ontologia deve ser capaz de responder;
- Coleta das terminologias que poderiam ser utilizadas pela ontologia;
- Seleção dos termos capazes de descrever características dos pacientes, materiais ou conceitos envolvidos no processo de recomendação;
- Revisão da lista de termos obtida pelos médicos;
- Definição das classes e hierarquia por uma abordagem *top-down*, começando pela definição dos conceitos mais gerais do domínio, seguida pela especialização desses conceitos;
- A CDPEO foi construída em dois principais níveis de abstração. O nível 1 inclui cinco termos que descrevem características de pacientes. O nível 2 inclui detalhes para cada classe do nível 1;
- Definição das propriedades de classes baseadas nos termos restantes para descrever a estrutura interna dos conceitos. As propriedades consistem em objetos e dados de propriedades;
- Definição das restrições de propriedades para completar a semântica precisa das classes;
- Criação de instâncias das classes na hierarquia. O CDPEO foi instanciado pelos dados de pacientes e foi definida uma classe de perfil de paciente no topo para ser o componente principal das instâncias;

- Com SWRL, regras foram codificadas para inferências mais complexas, como gerar uma propriedade nova de uma instância.

São utilizadas técnicas de PLN para mapear documentos para espaço vetorial. Primeiro há uma sumarização de tópicos para cada documento por palavras-chave. Dois algoritmos são utilizados, TF-IDF e TextRank, para extrair essas palavras-chave. Durante o processo, é dado um peso maior a palavras mais aptas a serem palavras-chave, como títulos e substantivos. Palavras com peso maior são responsáveis por aprimorar a extração das palavras-chave. Para os documentos em chinês, as sentenças são divididas em pedaços menores. Assim, observou-se que palavras compostas eram divididas também em palavras menores, porém era um problema, pois essas palavras continham, geralmente, mais informações do que a palavra quebrada. O problema foi resolvido aplicando uma série de condições para a filtragem dessas palavras e um dicionário para guardá-las antes de realizar a segmentação. Também são eliminados sinônimos na extração de palavras.

As palavras-chave extraídas são mapeadas para o espaço vetorial da ontologia para gerar o vetor de texto baseado na similaridade de cosseno entre vetores e palavras-chave. A similaridade é calculada baseada em um *word embedding* pré-treinado para cada palavra-chave e vetor. O modelo utilizado é o Word2Vec, junto com a arquitetura *bag of words* e uma janela de tamanho cinco, com o método de amostragem negativa para o treinamento.

Com os vetores de pacientes e de texto, é calculado o produto interno entre cada par para verificar a correlação entre dados de pacientes e materiais educacionais. O produto interno é uma similaridade de cosseno não-normalizada, considerando a similaridade de vetores em ambos direção e magnitude. Produtos internos maiores indicam maior correlação. As recomendações são geradas baseadas nesses produtos.

Como resultados, a ontologia contou com 40 classes, 31 propriedades de objetos, 67 propriedades de dados e 32 indivíduos. São definidas 80 regras SWRL para a semântica da lógica do mapeamento de dados de pacientes para o espaço vetorial da ontologia. O sistema de recomendação é implementado como uma aplicação móvel utilizada nos telefones dos pacientes. A precisão macro foi de 0,970 para a recomendação *top ranking* e uma pontuação MAP de 0,628, mostrando que a recomendação na área da saúde tem um grande potencial na automatização da educação dos pacientes com doenças crônicas. As técnicas PLN aliadas com estratégias de IA foram muito eficientes para melhorar a performance do sistema.

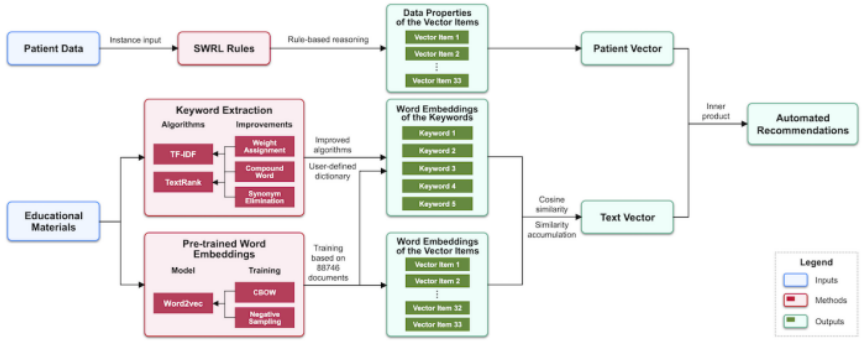

Figura 7 – As etapas de geração de recomendações [Wang et al. 2020]

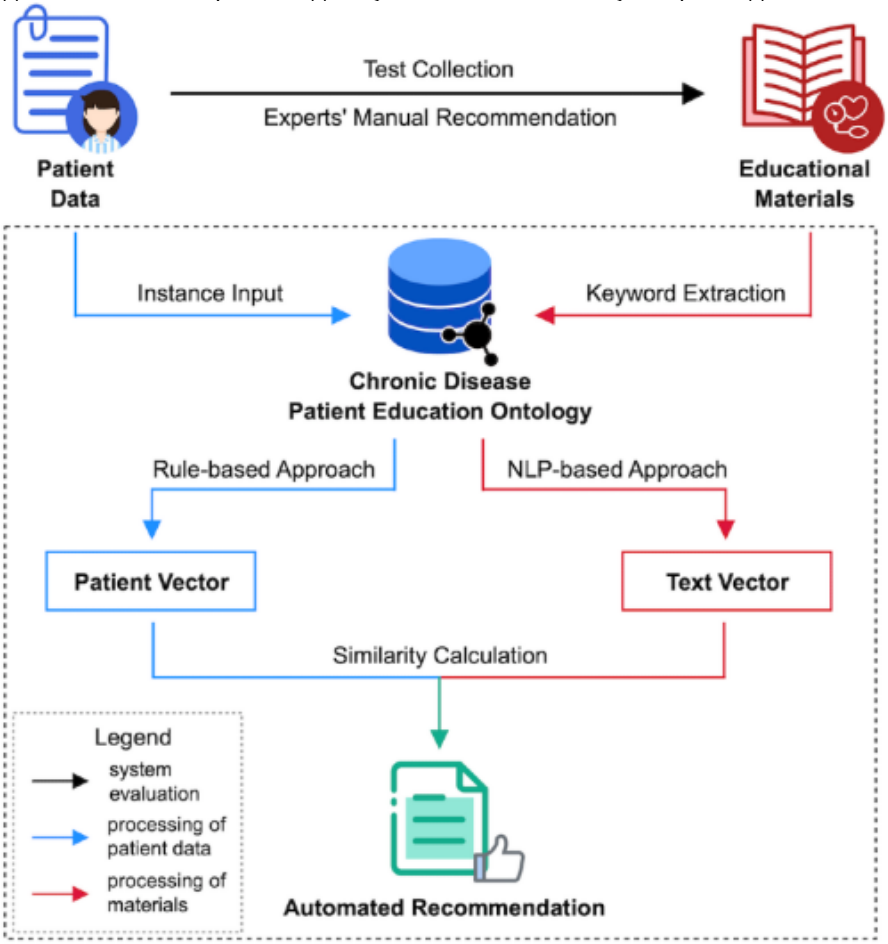

Figura 8 – O fluxo do sistema de recomendação proposto por [Wang et al. 2020]

3.4 Outros trabalhos

3.4.1 *Carreer Recommendation Systems using Content Based Filtering*

O que [Yadalam et al. 2020] busca com seu trabalho é resolver o problema da escolha de carreira dos jovens estudantes de engenharia com um sistema de recomendação de trabalhos baseado em conteúdo, mas que leva em consideração tanto os interesses do estudante, quanto seu perfil e habilidades.

A metodologia tem como objetivo resolver os problemas de partida a frio, confiança e privacidade gerados pela abordagem de filtragem colaborativa. Os autores utilizaram Python com bibliotecas para a construção do sistema, um *dataset* criado a partir do pré-processamento e combinação de múltiplas bases de dados, com 17 colunas e 20 mil registros sobre habilidade em diversas áreas de estudo, preferências e outras informações, e HTML e CSS para o *frontend*.

O sistema recomenda uma carreira a partir de um formulário. São geradas as três melhores opções de trabalho de acordo com a entrada. Foi utilizado Python e Flask no *backend* para gerar as recomendações. Para a comparação de vetores, similaridade de cosseno. As ferramentas pandas e numPy foram utilizadas na transformação de dados. *Label encoding* foi utilizada para traduzir os textos dos registros dos dados em números para o algoritmo das sugestões de trabalhos. O PLN é utilizado apenas no *feedback* dado pelos usuários numa seção de comentários sobre a aplicação, com objetivo de prover melhores resultados futuramente.

3.4.2 *An Efficient Approach of Product Recommendation System using NLP Technique*

O objetivo de [Sharma et al. 2021] é propor um sistema de recomendação de produtos eficiente utilizando a base de dados *Amazon Apparel*, que contém registros de 180 mil peças de vestuário, utilizando uma CNN (rede neural convolucional) e várias abordagens PLN como *bag of words*, Word2Vec, TF-IDF. Para a comparação de vetores, é utilizada distância euclidiana ponderada e as recomendações são do tipo baseado em conteúdo.

Como resultado, os autores apontam que o sistema conseguiu predizer produtos bastante similares aos selecionados. Em análise manual, concluem que o modelo teve sucesso com marcas, tipos e cores similares na recomendação das peças de roupas. Os algoritmos utilizados são eficientes e a inclusão das ferramentas e técnicas PLN reduz de maneira significativa o código e torna o sistema muito mais rápido.

3.4.3 *Movie Recommendation System Using NLP Tools*

Em seu artigo, [Kapoor, Vishal e Krishnaveni 2020] propõem uma abordagem diferente das soluções atuais para a recomendação de filmes. Utilizando análise de sentimento em avaliações de filmes feitas pelos próprios usuários, a solução apoia-se no uso de um aplicativo de celular que recebe essas informações de usuários logados, faz os procedimentos de tratamento de texto com PLN, então a comparação das palavras vetorizadas.

Os vetores gerados a partir das avaliações são comparados com vetores gerados a partir de dados de mais de 5000 filmes fornecidos pelo TMDB[7] (*"The Movie Database"*), disponível grátis na Internet. A detecção de sentimentos é realizada, então, por um modelo SVM (*Support Vector Machine*), responsável por categorizar palavras positivas e negativas. As recomendações são feitas a partir de um sistema de *ranking* dos filmes, comparando vetores de palavras de novas avaliações com palavras já armazenadas pelo sistema. Quando a palavra é totalmente nova, é feito o cálculo da similaridade com KNN, utilizando a distância euclidiana.

A avaliação é feita em duas partes, primeiro do modelo de detecção de sentimentos e depois da recomendação de filmes. Atentando-se apenas à recomendação, como o sistema fornecia um *ranking* de filmes de acordo com avaliações e gostos de gêneros do próprio usuário, a equipe realizou testes com usuários reais e entre 65-70% deles tiveram um *feedback* positivo com o sistema.

[7] https://www.themoviedb.org/

3.5 Discussão

Na tabela 5 são apresentadas as soluções de cada autor, levantadas durante a pesquisa de literatura, destacando os pontos mais importantes que compõem todo o processo dos sistemas de recomendação. O objetivo desta tabela é mostrar as principais diferenças e semelhanças entre as soluções da bibliografia e a solução proposta pelo presente trabalho.

Referência	Domínio	Dataset	Método de recomendação	Filtragem	PLN	Comparação
Wang	Material didático	Público 88 mil registros	Ontologia Algoritmo	Regras	TF-IDF TextRank Word2Vec	Cosseno
Cioffi	Filmes	Público 45 mil registros	Algoritmo	Híbrida	TF-IDF BERT Word2Vec	Cosseno
Naren	Disciplinas de universidade	Privado 9205 registros	Algoritmo	Conteúdo	Tokenização Lematização Remoção *stop words* TF-IDF	Cosseno Dice
Yadalam	Empregos	Privado e próprio 20 mil registros	Algoritmo	Conteúdo	Detecção sentimentos	Cosseno
Sharma	Vestuário	Público 180 mil registros	Rede neural CNN Algoritmo	Conteúdo	*Bag of words* Word2Vec TF-IDF	Distância euclidiana ponderada
Kapoor	Filmes	Público 5000 registros	Algoritmo	Conteúdo	NLTK Word2Vec TF-IDF	KNN distância euclidiana

Tabela 6 – Comparativo de trabalhos relacionados extraídos da literatura

Como é possível observar, já existem diversas soluções de diferentes formas para a questão da recomendação de itens utilizando técnicas de PLN. Como destaque, duas se sobressaem, tanto pela eficiência dos modelos propostos, quanto pela criatividade e boas escolhas nas abordagens utilizadas. O modelo de Wang et al, 2020 tratou de um tema bastante relevante e de grande utilidade, que é a automatização do processo de aprendizado de pacientes com doenças crônicas, fazendo uso de uma ontologia para gerar recomendações relevantes de material didático, por meio de um aplicativo. Naren et al, 2020 trouxe uma solução para um problema parecido com o levantado por este trabalho,

e apresentou resultados bastante satisfatórios, principalmente pelo uso do coeficiente de Dice, junto com o cosseno, garantindo recomendações precisas e relevantes.

Das técnicas PLN utilizadas, serão aproveitadas para a solução proposta a tokenização, a lematização (pela sua vantagem em relação à stemização), TF-IDF e Word2Vec. Para o cálculo da similaridade, a abordagem do cosseno foi a mais utilizada e, portanto, é a candidata mais forte para compor a etapa de comparação de vetores.

4 Proposta de sistema de recomendação híbrido

Ao analisar o estado-da-arte levantado nas pesquisas e os respectivos resultados de cada solução, a proposta para este trabalho é definida como um sistema de recomendação híbrido, em que há a ação de ambas filtragem colaborativa e por conteúdo. No funcionamento do sistema proposto, primeiramente há a filtragem colaborativa, contando com os dados de perfis de usuário, retornando recomendações de itens similares aos itens relacionados ao perfil de entrada, e em um segundo momento a filtragem baseada em conteúdo, fazendo um refinamento das recomendações e priorizando os itens de maior relevância para a entrada fornecida.

O sistema terá como entrada dados de interações de usuários com os itens que podem ser recomendados, disponíveis pela base de dados. Esses itens são todos os cursos disponíveis pelo *dataset* de uma plataforma de cursos *online*, e o sistema será baseado em uma aplicação real, utilizando dados reais, porém trazidos para um contexto de testes e experimentos, com usuários de teste, situações, variáveis e cenários definidos pelo autor do trabalho. Em seguida, será descrito o funcionamento do sistema e como são definidas as duas grandes etapas, a filtragem colaborativa e a baseada em conteúdo.

A seguir será descrito o passo-a-passo do desenvolvimento do modelo de recomendação e será apresentado um diagrama com os componentes e atividades mais importantes, descrevendo o funcionamento geral de como serão geradas as recomendações.

A figura 9 evidencia, em alto nível, as etapas do funcionamento do sistema. A etapa de número 1 compreende três tarefas que compõem uma parte comum às duas filtragens: preparação de dados. Dessa forma, as etapas de 1 a 4, na parte esquerda e central, realizam em conjunto a **filtragem colaborativa** no sistema. Após isso, as etapas 1 e 5, na parte à direita, realizam a **filtragem baseada em conteúdo**:

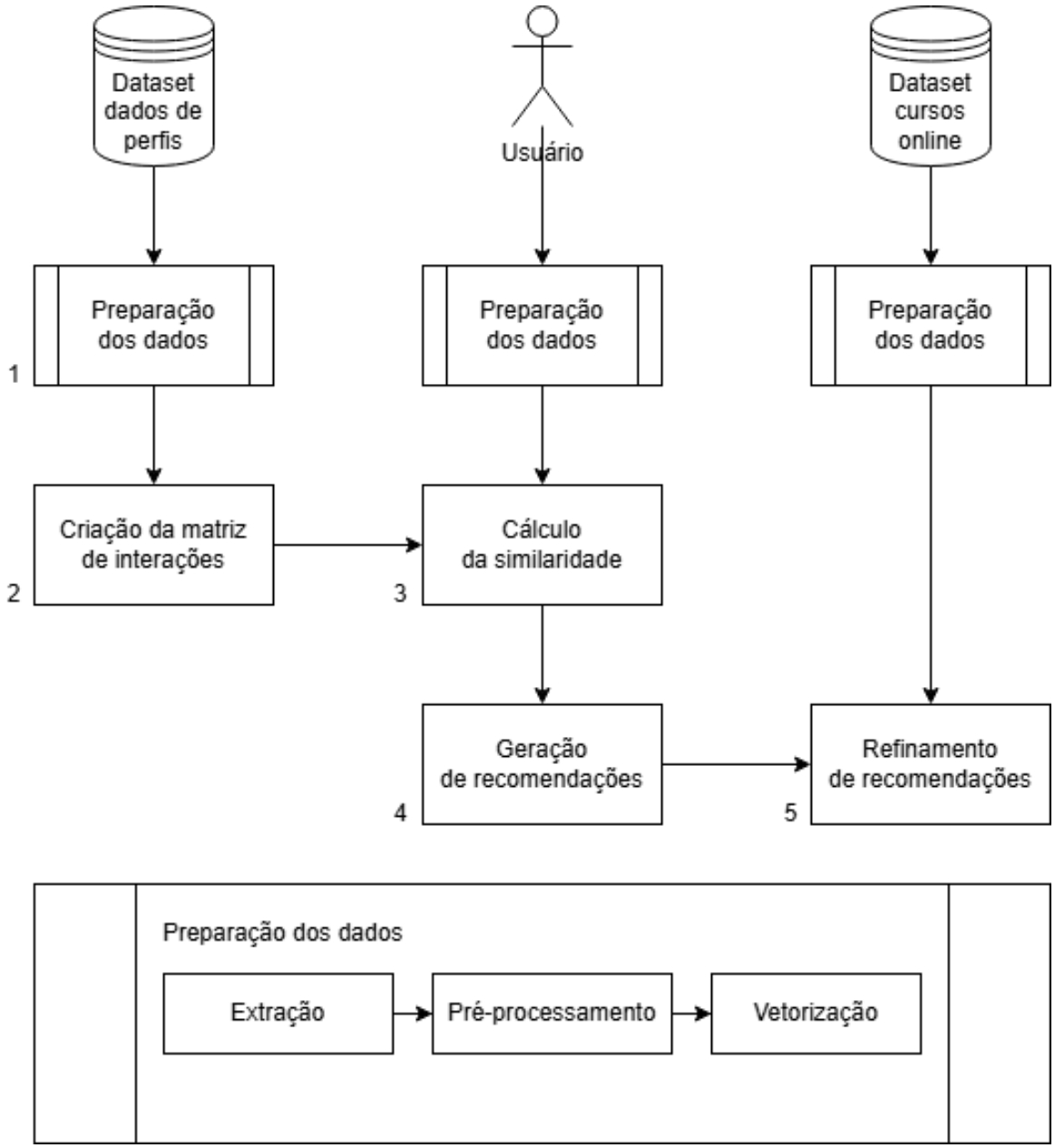

Figura 9 – O fluxo do sistema de recomendação proposto do presente trabalho

O início do sistema se dá pela etapa 1, **preparação dos dados**, na etapa da filtragem colaborativa, com a extração dos dados, que é feita em dois âmbitos: os disponíveis pelas bases de dados e os fornecidos pelos usuários dos experimentos. Com os dados extraídos, inicia-se o **pré-processamento de dados**. Nesta etapa, há a análise dos dados obtidos e o tratamento, para que se adequem às técnicas, algoritmos e ferramentas utilizadas no processo de geração das recomendações. Os dados mais relevantes aqui são os que dizem respeito às interações dos usuários com os itens disponíveis no *dataset*, ou seja, avaliações textuais, avaliações numéricas, tempo de permanência em um curso, se o usuário terminou o curso, se o usuário indica o curso, entre outras informações relacionadas. Os dados, ao final do processo, passam por uma outra transformação para que possam ser recebidos pelo algoritmo em seguida. Esse processo é chamado de vetorização.

Na **vetorização**, os dados já tratados e pré-processados são transformados em estruturas numéricas chamadas vetores, e são eles os elementos que participam do processo de comparação para que se encontre similaridades entre eles, e também compõem a estrutura da etapa 2, a **matriz de interações**. A matriz criada é uma estrutura bidimensional em que se encontram os vetores gerados pelos dados extraídos e tratados, e é percorrida pelo sistema ao realizar a comparação de vetores. Esse processo é denominado cálculo de similaridade, e é muito comum em sistemas de recomendação, pois o que se quer é encontrar sempre itens de maior similaridade entre si, que é o núcleo de uma recomendação.

Com todos os dados das bases disponíveis como vetores, pode-se também **extrair o perfil do usuário** que faz uso do sistema. Todos os processos descritos anteriormente valem para os dados de usuário. Então, eles são extraídos, pré-processados, transformados em vetores, e após essa preparação, os dados de perfis estão prontos para participar da etapa 3, a última antes da geração de recomendações, já citada anteriormente, o **cálculo da similaridade**.

A partir dos vetores disponíveis pela base de dados, o vetor que modela o perfil de usuário é selecionado e utilizado para a comparação. Nesse processo, o código compara todas as estruturas vetoriais disponíveis com o vetor do usuário e seleciona as que tiverem a maior **similaridade**, uma métrica entre 0 e 1 para determinar quão próximos são dois vetores. Quanto mais próximo de 1, maior é a similaridade. Com todos os itens comparados, os de maior similaridade são selecionados e retornados no sistema, compondo a etapa 4, com o conjunto inicial de **recomendações geradas**.

Após o primeiro levantamento de itens relevantes para o usuário, começa a segunda grande parte da recomendação, o refinamento com base nos dados textuais de conteúdo dos cursos, por meio da filtragem baseada em conteúdo. Do *dataset* de cursos, a extração dá ênfase ao título, descrição e quaisquer outros dados de texto que descrevem o conteúdo dos cursos. Esses dados são priorizados e extraídos para participarem da etapa seguinte.

Com os dados extraídos, inicia-se a preparação dos dados. Nesta etapa, há a análise dos dados obtidos e o tratamento, para que se adequem às técnicas, algoritmos e ferramentas utilizadas no processo de refinamento das recomendações. É também nesta etapa que entra a **PLN**, onde se aplicam diversas formas de transformação de dados textuais, remoção de palavras irrelevantes, redução de palavras para o radical de maior peso, remoção de pontuações, entre outras. Após esse processo, as palavras que restam são transformadas em ***tokens***, que por sua vez são a entrada para o processo seguinte, a vetorização.

O processo de vetorização segue igual ao descrito anteriormente, transformando os *tokens* extraídos em estruturas vetoriais que são a entrada do processo de comparação de vetores e cálculo de similaridade.

A última etapa, de número 5, é definida por um novo **cálculo de similaridade**. Para este cálculo, o primeiro conjunto retornado anteriormente é selecionado para a comparação com os dados textuais de cursos que foram transformados em vetores. Ao final do processo, os itens de maior similaridade são ranqueados (*top-n* itens) e retornados para o usuário, de acordo com o número de cursos escolhido para serem recomendados.

Os capítulos 5 e 6 apresentam e descrevem com detalhes cada filtragem, o primeiro com a filtragem colaborativa, e o segundo com a filtragem baseada em conteúdo. Cada uma das seções dará detalhes maiores sobre cada filtragem, explicando cada uma das etapas maiores e menores em cada um dos tipos de filtragem. A primeira, colaborativa, é desenvolvida para gerar as recomendações que são refinadas em seguida, e levantar o primeiro grupo de cursos recomendados, de onde saem os cursos que são recomendados ao final, para o usuário. A segunda, baseada em conteúdo, é desenvolvida para servir ao propósito do refinamento, definido na seção atual.

Nas últimas subseções de cada uma das próximas seções há a parte de testes e experimentos. Os testes deste trabalho dizem respeito a um contexto teórico e explicativo, em que os dados utilizados são reais, mas inseridos em um sistema simulado, com usuários e cenários definidos pelo autor. Cada indivíduo que participa dos experimentos tem seu perfil extraído e definido manualmente, algo que ocorre naturalmente em aplicações e plataformas *online* que possuem sistemas de recomendação, como Udemy, Udacity, Netflix e Amazon. Nesses serviços, o perfil de usuário está sempre disponível e os dados são mantidos para gerar e aprimorar as recomendações. Aqui, a forma como esses dados são obtidos é explicada em cada seção, e são utilizados por meio da entrada do usuário no sistema simulado.

O sistema será desenvolvido em código aberto em um Jupyter Notebook[1], disponível para executar em qualquer editor de texto com suporte e qualquer máquina com Python 3 e as devidas dependências instalados. Cada etapa terá pedaços de texto ou comentários explicando o que está sendo feito e pedaços de código construindo e executando o sistema proposto. O ChatGPT entra como um auxílio nas partes mais detalhistas do código e mais específicas, para otimizar o desenvolvimento. Como explicado anteriormente, os perfis de usuário serão a entrada de cada sistema, sendo introduzidos manualmente pelo usuário ativo nas funções que retornam as recomendações de cursos.

Como principais ferramentas no desenvolvimento do modelo colaborativo e do modelo baseado em conteúdo serão utilizadas as seguintes bibliotecas e recursos:

- NumPy[2]

[1] https://github.com/marcelomuller/sr-projeto
[2] https://numpy.org/

- pandas[3]
- SciPy[4]
- scikit-learn[5]
- re (Python)[6]
- NLTK[7]
- GenSim[8]

[3] https://pandas.pydata.org/
[4] https://scipy.org/
[5] https://scikit-learn.org/
[6] https://docs.python.org/3/library/re.html
[7] https://docs.python.org/3/library/re.html
[8] https://pypi.org/project/gensim/

5 Sistema de recomendação: filtragem colaborativa

Para o caso da filtragem colaborativa, um dataset de interações de usuário se fez necessário para compor a matriz que faria o papel do elemento principal para a geração de recomendações. Com base nisso, dados de avaliações de alunos dos cursos da Coursera foram levantados. O conjunto conta com dois *datasets*, um de avaliações com mais de 200 mil registros de avaliações de usuários, e um de cursos, com mais de 600 cursos, ambos coletados no ano de 2020.

O funcionamento geral do SR baseado em filtragem colaborativa desenvolvido por este trabalho pode ser resumido em 5 etapas: **reconhecimento dos dados**, onde um primeiro estudo é feito sobre as bases de dados obtidas, **preparação dos dados**, em que é feita uma adaptação para se enquadrar nos métodos que serão utilizados e nas técnicas para gerar as recomendações, **criação da matriz esparsa**, em que uma matriz de usuário-item é gerada para ser a peça central na geração de novos itens, **geração de curso recomendado**, uma função que utiliza o KNN para gerar novos cursos com base em um curso de entrada, e **fatoração de matriz**, uma etapa extra, não necessária, que ilustra um processo de compactação da matriz utilizada anteriormente que, em alguns casos, pode gerar melhores recomendações.

A figura a seguir ilustra o funcionamento, em alto nível, do sistema de recomendação proposto:

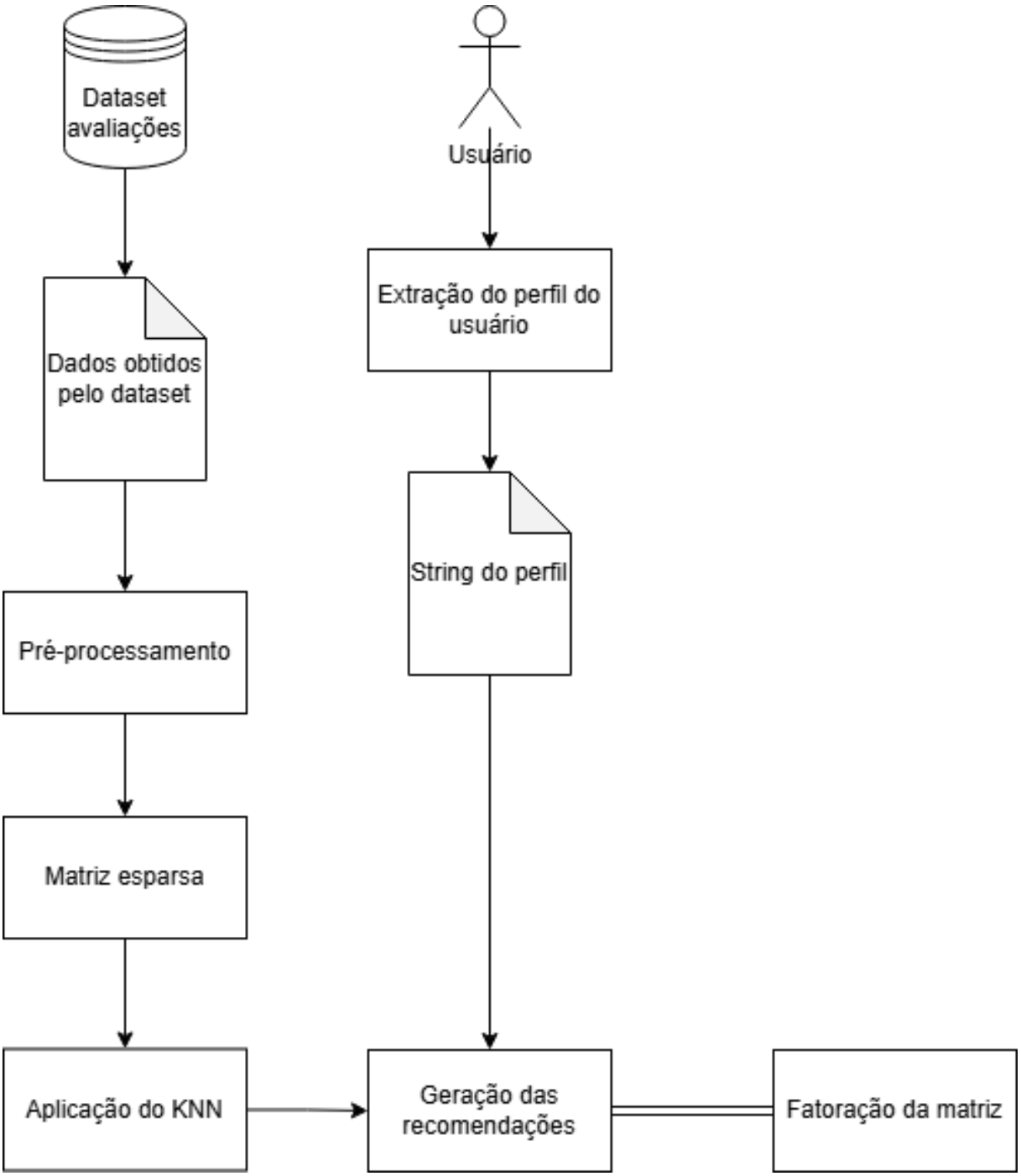

Figura 10 – O fluxo do sistema de recomendação colaborativo

O diagrama acima representa todas as etapas que participam do processo de geração das recomendações da solução. O início se dá pelo carregamento dos dados de ambos os *datasets*, avaliações e cursos. Os processos seguem na ordem em que aparecem. **Pré-processamento**, que compreende um tratamento inicial nas colunas disponíveis em cada base de dados, como remoção de colunas irrelevantes e adequação de identificadores. Em seguida, **matriz esparsa**, em que é criada uma matriz usuário-item, utilizando IDs de usuários e cursos, para servir como base do funcionamento do algoritmo de recomendação colaborativa. Após isso, a aplicação da técnica do **KNN (k-Nearest Neighbors)** para

gerar k cursos mais relevantes, de acordo com uma métrica estabelecida, um dado valor de k e um curso inicial para obter os próximos mais similares. Por fim, a **geração das recomendações**, fornecendo como entrada e como perfil de usuário um curso X para encontrar os K cursos mais próximos a ele. A última etapa, sendo opcional, **fatoração da matriz**, reduz a dimensionalidade da matriz inicial criando duas novas matrizes, podendo ou não obter resultados melhores do que a primeira matriz.

5.1 Obtenção e carregamento dos dados

O caso da filtragem colaborativa iniciou com um grande problema, encontrar uma base de dados rica em interações de usuários para os cursos trabalhados anteriormente na filtragem baseada em conteúdo.

A primeira proposta para tratar esse desafio foi criar um *dataset* do zero, com o mínimo possível de colunas (id de usuário, id do curso e avaliação, por exemplo) e utilizando apenas alguns assuntos de todos os disponíveis, e desses assuntos, uns cinco ou seis cursos apenas. A ideia era realizar uma espécie de experimento direcionado, contando com a participação de colegas e amigos de diversas áreas que pudessem contribuir com notas, porém, com base apenas no título, descrição e nota média de cada curso, já que eles não eram alunos de verdade.

Em meio à implementação dessa ideia, um conjunto de *datasets*[1] disponíveis no Kaggle foi encontrado. Um deles com cursos da Coursera, com título, o nome da instituição que o disponibilizou na plataforma, a URL e o id. E o outro com avaliações escritas e numéricas desses cursos. Levando em consideração que eram bases de dados muito mais completas do que a da proposta anterior, e isso geraria uma matriz menos esparsa e recomendações melhores, esta solução foi a escolhida para implementar o SR colaborativo.

O primeiro *dataset*, o de cursos, contém 604 cursos únicos e quase 1 milhão e meio de avaliações. Dessas avaliações, há mais de 287 mil usuários únicos, sendo que todos os cursos contam com pelo menos uma avaliação.

Com inicialmente cinco colunas, o *dataset* mostra as avaliações textuais na coluna *reviews*, os nomes de cada aluno que escreveu a avaliação na coluna *reviewers* (sendo um padrão o primeiro nome seguido pelas iniciais dos sobrenomes, precedido por um 'By'), a data da postagem da avaliação em *date_reviews*, a nota numérica em *rating* (a coluna mais importante nas recomendações), e *course_id* com os IDs de cada curso. Exemplo:

[1] https://www.kaggle.com/datasets/imuhammad/course-reviews-on-coursera

reviews	reviewers	date_reviews	rating	course_id
Pretty dry, but I was able to pass with just t...	By Robert S	Feb 12, 2020	4	google-cbrs-cpi-training
would be a better experience if the video and ...	By Gabriel E R	Sep 28, 2020	4	google-cbrs-cpi-training
Information was perfect! The program itself wa...	By Jacob D	Apr 08, 2020	4	google-cbrs-cpi-training
A few grammatical mistakes on test made me do ...	By Dale B	Feb 24, 2020	4	google-cbrs-cpi-training
Excellent course and the training provided was...	By Sean G	Jun 18, 2020	4	google-cbrs-cpi-training

Tabela 7 – Primeiras linhas do *dataset* Coursera_reviews

A matriz utilizada no processo de recomendação é criada utilizando um *dataframe* chamado *ratings*, o qual armazena as avaliações dos usuários, com id de usuário, id do curso e avaliação numérica:

userId	courseId	rating
0	machine-learning	4
1	python-data	3
2	technical-support-fundamentals	4
3	what-is-datascience	3
4	r-programming	2

Tabela 8 – Exemplo do *dataframe* ratings

5.2 Pré-processamento dos dados

O pré-processamento de dados nesse modelo é diferente da versão baseada em conteúdo, e precisa de menos ajustes. Não há colunas a serem usadas com arquivos de texto nesse caso, então as técnicas PLN utilizadas no pré-processamento do modelo baseado em conteúdo, como remoção de *stopwords*, stemização e aplicação de filtros por expressões regulares não são relevantes aqui. Isso será melhor comentado na seção de trabalhos futuros.

Duas alterações nos dados foram necessárias. Adicionar uma coluna de id de usuário (userId) no *dataframe* de avaliações, *ratings* (citado anteriormente), que inicialmente contava apenas com o nome do aluno que avaliou o curso, e como o nome dos alunos não é relevante para o modelo, a coluna inteira foi removida também. E, para evitar que

o mesmo usuário avalie duas vezes o mesmo curso, os dados duplicados com base em um par de *userId* X e *courseId* Y foram removidos, mas sem levar em conta a data das avaliações. Isso evita com que a matriz acabe unindo as avaliações duplas e seja preenchida por números maiores do que o intervalo possível de valores para o *rating*, que é de 0 a 5. O exemplo a seguir ilustra como ficou o *dataframe*:

n	userId	courseId	rating
0	0	google-cbrs-cpi-training	4
1	1	google-cbrs-cpi-training	3
2	2	google-cbrs-cpi-training	4
3	3	google-cbrs-cpi-training	3
4	4	google-cbrs-cpi-training	2

Tabela 9 – Coluna userId criada após pré-processamento

5.3 Criação da matriz esparsa usuário-item

Esta etapa é responsável por criar uma matriz esparsa que age como o principal componente na geração de recomendações com base nos dados apresentados. A entrada da função que cria a matriz é o *dataframe ratings*, e a partir dos campos de id *courseId* e *userId*, os mapeadores e mapeadores inversos são criados em conjunto, e a partir deles pode-se extrair as duas dimensões da matriz, usuário e item (curso). O processo gera uma matriz de *shape* 287808x604, sendo usuários e cursos, respectivamente. Após a criação, a matriz será uma estrutura grande e formada majoritariamente por zeros (por isso o nome, esparsa), com os registros não-zero preenchidos pelas avaliações do par *userId-courseId*. A tabela a seguir é uma representação de parte da matriz:

userId	courseId	rating
0	41	3
1	42	4
2	65	1
3	69	4

Tabela 10 – Tabela de representação da matriz esparsa

5.4 Obtenção do perfil

Diferente da filtragem baseada em conteúdo que será apresentada no próximo capítulo, a forma como os perfis são obtidos aqui é diferente. Nos *datasets*, não há possibilidade de classificar cursos quanto ao nível (iniciante, avançado etc.) como na outra filtragem, visto que aqui conta-se apenas com a nota atribuída por cada usuário, já que, como explicado anteriormente, as avaliações em texto não são consideradas. Dito isso, os usuários que fazem parte do experimento são usuários encontrados na base de dados. Fazendo dessa forma pode-se mitigar também o problema do *cold start*, ao menos para o objetivo do experimento feito neste trabalho, porque não há possibilidade de um usuário não ter ainda avaliado pelo menos um curso.

O perfil dos usuários é representado por uma *string*, com palavras separadas por hífens, que é ou o título do curso respectivo, ou parte do título. Essas *strings* são os próprios IDs dos cursos no *dataset* utilizado e são elas que servem de entrada para o modelo de recomendação, pois ele traz cursos similares a um curso de entrada.

Esse perfil é totalmente representado apenas pelo primeiro curso dentro do *dataset* que o indivíduo tenha feito. Essa é uma limitação da abordagem escolhida, também devido ao fato de não se ter um conjunto de testes para realizar os experimentos, como é comum em outras técnicas, abordagens e áreas do ML. No modelo proposto, não se tem informações de nível de usuário, área, gostos ou preferências. A opção escolhida que mais se encaixou com a forma como as bases de dados são e o funcionamento do motor de recomendação com o KNN foi representar o perfil sendo a *string* ID do curso inicial, com o qual ele deseja encontrar outros cursos parecidos. Exemplo:

String (ID)	Título do curso
'python-data'	Python Data Structures
'learning-how-to-learn'	Learning How to Learn: Powerful mental tools to help you master tough subjects
'data-scientists-tools'	The Data Scientist's Toolbox

Tabela 11 – Exemplo de string de perfil e o título de curso que representa

5.5 Aplicação do KNN e recomendações

O método escolhido para gerar as recomendações foi o KNN (k-Nearest Neighbors), que traz um número k de vizinhos para um dado item, baseado em alguma métrica. Na seção de experimentos algumas métricas são utilizadas e comparadas, como a distância de cosseno e distância *Manhattan*. A métrica padrão é o cosseno.

A função que realiza o cálculo do KNN recebe o id do curso do perfil de interesse, os mapeadores criados junto com a matriz esparsa, a própria matriz, o valor de k e a métrica escolhida. O mapeador aponta na matriz o vetor que deve ser utilizado, com base no ID do curso utilizado na entrada. Com o vetor levantado, os k vizinhos são encontrados (k + 1 para excluir o próprio curso da entrada) e os IDs dos cursos são retornados. Exemplo:

Na matriz, os cursos têm nomes mapeados para valores numéricos. Supondo que o curso principal da linguagem Python tenha ID 233, pode-se encontrar registros na matriz da seguinte forma:

userId	courseId
25	233
333	233

Tabela 12 – Representação de dois usuários que avaliaram o curso 233

Os mapeadores transformam os valores de *userId* e *courseId* em índices de 0 a n, e os inversos fazem o caminho contrário. Eles funcionam da seguinte forma:

0:	0
1:	1
2:	2
3:	3

Tabela 13 – Representação do mapeador do *userId*

'aboriginal-education'	0
'access-control-sscp'	1
'accounting-analytics'	2
'accounting-data-analytics-python'	3

Tabela 14 – Representação do mapeador do *courseId*

No KNN, esse ID por escrito servirá de entrada para retornar IDs de cursos vizinhos, para o determinado K, utilizando determinada métrica para o cálculo da distância (padrão é cosseno). Como é mostrado a seguir:

Curso de interesse: 'python'

K: 5

Cursos retornados:

'python-data'
'python-network-data'
'python-databases'
'machine-learning'

Tabela 15 – Retorno da função do KNN

Esse exemplo ilustra como as recomendações já funcionam, porém ainda mostrando apenas os IDs dos cursos. O fato de os IDs serem em texto contribuiu para que o usuário já tenha uma boa ideia de como está o funcionamento do modelo, mesmo sem visualizar o título do curso por completo.

Após esse processo, os nomes dos cursos são listados e levantados de acordo com os IDs obtidos a partir do KNN, e dessa forma as recomendações são retornadas com os nomes dos cursos:

Curso de interesse: 'python'

K: 5

Cursos retornados:

Python Data Structures
Using Python to Access Web Data
Using Databases with Python
Machine Learning

Tabela 16 – Recomendações baseadas no k fornecido e curso de entrada

5.6 Fatoração de matriz

Após o uso completo da matriz criada anteriormente, há também a possibilidade de realizar um processo a mais para que, em alguns casos, obtenha-se melhores recomendações. Esta última etapa é opcional e será comentada na seção de experimentos, para fins de comparações entre a matriz original e a matriz fatorada.

Quando se busca aprimorar recomendações visando *features* latentes, que não se consegue enxergar apenas observando a base de dados como é, a fatoração de matriz pode ser uma opção de técnica interessante para representar de forma mais compacta os gostos dos usuários e descrições de itens. Isso se torna ainda mais interessante quando os dados trabalhados são muito esparsos. O algoritmo fatora a matriz original usuário-item em duas matrizes: usuário-fator (n de usuários, k) e item-fator (k, n de itens).

A redução de dimensão da matriz busca evidenciar os gostos. O que cada *feature* k representa não é exato, então o valor real dessas features é interpretativo de acordo com o *dataset* e com os perfis dos usuários.

A técnica utilizada foi o TruncatedSVD[2] do sklearn, e ela permite que o usuário altere como quiser os valores de componentes presentes na matriz e também o número de iterações. Conforme esses valores mudam, melhores ou piores podem se tornar as recomendações retornadas. É preciso encontrar uma combinação "especial" para cada base de dados usada, e até para cada caso trabalhado. Exemplo de uso:

N componentes: 30

N iterações: 40

Shape: 604x30

Curso: 'python'

Foundations of Data Science: K-Means Clustering in Python
Corporate & Commercial Law I: Contracts & Employment Law
Introduction to Economic Theories
Programming Fundamentals

Tabela 17 – Saída da matriz redimensionada pela fatoração

Como a saída evidencia, não necessariamente se tem resultados melhores do que os apresentados pela matriz X anterior. Uma de várias explicações é o tamanho da base de dados, já que a fatoração se beneficia de bases de dados muito grandes e esparsas. E também, como já comentado, é preciso encontrar o *sweet spot* na configuração de todos os valores.

5.7 Experimentos

Esta seção trata da condução completa dos experimentos de testes realizados com o modelo construído e a base de dados utilizada.

[2] https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

Na primeira parte, uma breve descrição contextualizando os indivíduos, cenários e diferentes estilos para cada exemplo de experimento. Em seguida, a origem dos dados é detalhada, evidenciando a captação do perfil de cada usuário que participa do experimento e a forma didática como esses perfis fazem para gerar as recomendações.

Na próxima subseção, as métricas utilizadas são citadas, explicando motivos para o uso e decisões, conforme se deu a condução do experimento e a criação do modelo.

Por fim, a condução dos experimentos para cada caso, e ao final de cada um, uma discussão sobre os resultados obtidos e esperados, e algumas interpretações sobre as recomendações para cada cenário.

5.7.1 Contexto e cenários

Para este experimento, três cenários com três indivíduos de diferentes objetivos e contextos de vida foram reunidos para compor o elenco de sujeitos de teste. São situações realmente diferentes para reforçar que esses cursos servem para todo tipo de objetivo e assunto desejado. Cada um busca uma categoria específica de curso online para suprir seus interesses e agregar à sua vida acadêmica e profissional, para um hobby, ou até mesmo por pura curiosidade. Implementando o sistema proposto, foram definidos:

1. Um aspirante a cientista de dados que quer ingressar na carreira e já realizou o curso inicial de Python;

2. Um pai de primeira viagem em busca do máximo possível de materiais que o ajudem a levar a vida nova, e que já realizou o curso Everyday Parenting;

3. Um entusiasta da música eletrônica que deseja se aprofundar na parte de produção musical, e já cursou A arte da produção musical.

5.7.2 Origem dos dados

A avaliação consiste na captação do perfil do usuário, utilizando o id do curso inicial realizado, que servirá de base para o KNN retornar os k cursos mais próximos. O modelo utiliza como base de dados um *dataframe* com as avaliações dos outros usuários da plataforma. Uma matriz esparsa é criada com base nesse *dataframe*, e utilizando os mapeadores, o KNN percorre a matriz e traz os k cursos de maior relevância para a pesquisa, de acordo com outras avaliações de usuários que realizaram também aquele curso, ou seja, usuários de perfil similar.

O valor de k utilizado no experimento para o KNN será k - 1, ou seja, para um k = 11, são retornados 10 cursos. Isso é feito para excluir o próprio curso inicial do retorno.

5.7.3 Métricas dos experimentos

Como métricas de avaliação, foram utilizadas: precisão, para medir a proporção de itens relevantes entre os recomendados e NDCG (*Normalized Discounted Cumulative Gain*), para o caso de a ordem das recomendações estar errada e itens relevantes aparecerem antes de itens relevantes. Nesse último caso, foi atribuído por análise própria do autor em uma escala 0 a 5, seguindo a lógica do funcionamento do sistema e a utilização de notas dadas pelos usuários nos cursos, para compor os cálculos do NDCG e obter valores mais condizentes. A ordem ideal considerada para o cálculo do NDCG foi a própria ordem sugerida na seção de discussão de cada caso do experimento, seguindo da maior relevância obtida até a menor. Quanto mais próximo de 1 for o resultado do NDCG, melhores foram as recomendações daquele caso do experimento.

5.7.4 Condução dos experimentos

CASO 1:

O jovem universitário que almeja seguir carreira na área de ciência de dados deseja se aprofundar na parte técnica. Ele realizou o curso básico de Python e está buscando mais materiais:

- Nome: Vinicius
- Curso realizado: Programação para todos (Primeiros passos no Python)
- ID: 'python'

O perfil do usuário será representado por **'python'**, o ID do curso realizado, que é também uma das entradas do KNN.

Para o valor de $\mathbf{k = 6}$, os resultados:

Porque você avaliou Programming for Everybody (Getting Started with Python):

n	Título	Relevância
1	Python Data Structures	5
2	Using Python to Access Web Data	0
3	Using Databases with Python	4
4	Machine Learning	3
5	Neural Networks and Deep Learning	1

Precisão	NDCG
4/5 = 80%	0,9438

Ordem ideal determinada: 1, 3, 4, 5 e 2

Discussão dos resultados:

O primeiro curso recomendado é de nível básico e não necessita de conhecimento prévio. Ele trata da base do Python dando ênfase em estruturas de dados e operações iniciais com a linguagem. Por isso é um curso muito interessante para iniciantes.

O segundo curso é bem direcionado a *web*, passando por *scraping* e trabalhando com JSON e XML, então é algo bem específico e fora do escopo do que o aluno procura.

O terceiro é um curso focado em bancos de dados utilizando Python, trazendo conceitos de CRUD, SQL, orientação a objetos, entre outros assuntos para uma base sólida, bem importante para ciência de dados.

O quarto é muito interessante, já que ML e *ata science* andam lado a lado na área de dados. O curso traz conceitos utilizados amplamente pelos profissionais de ciência de dados.

O último apenas foi considerado não relevante por se tratar de uma área mais específica de IA e ML, algo que foge um pouco do escopo desejado. Seria um curso mais interessante para um aluno de inteligência artificial, por exemplo.

CASO 2:

O novo pai que procura mais formas de reunir conhecimento sobre a vida diferente e cheia de desafios que o espera:

- Nome: Bruno
- Curso realizado: Everyday Parenting (Paternidade cotidiana): The ABCs of Child Rearing (O ABC da criação de filhos)
- ID: everyday-parenting'

O perfil do usuário será representado por **'everyday-parenting'**, o ID do curso realizado, que é também uma das entradas do KNN.

Para o valor de $\mathbf{k = 6}$, os resultados:

Porque você avaliou Everyday Parenting (Paternidade cotidiana): The ABCs of Child Rearing (O ABC da criação de filhos):

n	Título	Relevância
1	The Science of Well-Being	4
2	Child Nutrition and Cooking	5
3	Learning How to Learn: Powerful mental tools to help you master tough subjects	0
4	Positive Psychology: Resilience Skills	3
5	Psychological First Aid	3

Precisão	NDCG
4/5 = 80%	0,8574

Ordem ideal determinada: 2, 1, 4, 5 e 3

Discussão dos resultados:

O primeiro resultado é um curso que traz diversos desafios para a finalidade de criar hábitos mais saudáveis e produtivos, contribuindo para que a pessoa seja mais feliz. Cursos sobre psicologia de hábitos, rotina e saúde são ótimas sugestões para novos pais.

O segundo curso foca totalmente em nutrição infantil, levantando os mais diversos pontos das causas da má saúde nas crianças, como obesidade infantil, e ensina diversas práticas e habilidades para garantir uma dieta saudável para as crianças. O corpo é o bem mais precioso, e hábitos criados na infância podem durar pela vida inteira. Excelente recomendação.

O terceiro é um curso que ensina técnicas e métodos diferentes para aprender temas difíceis, combater procrastinação, técnicas de memorização, etc. É uma recomendação interessante para todos que estejam estudando para um concurso público, ou ao menos precisam de ajuda para estudar para uma prova muito difícil, ou têm pouco tempo para aprender um tema mais complexo. Para o caso de um pai que deseja um conteúdo mais leve, que seja apenas uma suplementação, não é tão relevante.

O quarto é relevante por ensinar aos alunos a desenvolver resiliência, que é uma virtude importante para todos, especialmente pais que precisam ser fortes em todos os âmbitos para elevar sempre a qualidade da criação dos filhos.

O quinto curso, apesar de ter como objetivo tratar de amparo psicológico emergencial em situações graves, é também muito útil no ambiente familiar. Pode ser de grande valia para lidar com problemas que a criança pode vir a apresentar futuramente, quando tiver mais idade.

CASO 3:

O fã e entusiasta de música eletrônica que foi inspirado pelos seus artistas favoritos

a começar a produzir sua própria música:

- Nome: Paulo
- Curso realizado: A arte da produção musical
- ID: 'producing-music'

O perfil do usuário será representado por **'producing-music'**, o ID do curso realizado, que é também uma das entradas do KNN.

Para o valor de $\mathbf{k = 6}$, os resultados:

Porque você avaliou The Art of Music Production:

n	Título	Relevância
1	The Technology of Music Production	5
2	Developing Your Musicianship	5
3	Music Business Foundations	3
4	Songwriting: Writing the Lyrics	4
5	Machine Learning	0

Precisão	NDCG
$4/5 = 80\%$	0,9909

Ordem ideal determinada: 1, 2, 4, 3 e 5

Discussão dos resultados:

A melhor recomendação está no primeiro resultado. Apesar de ter um nível mais básico do que o curso inicial, ele é composto de aulas sobre o processo completo da produção musical, inclusive aulas teóricas sobre o som em si.

O segundo curso é direcionado totalmente à música, passando por escalas, tonalidades, harmonia e assinaturas diferentes. A habilidade de tocar instrumentos e entender sobre toda a composição de uma música é um tema muito procurado e que contribui muito na carreira de produtores musicais.

O terceiro é focado no mercado da música, ensinando conceitos como direitos autorais, o papel dos agentes, contratos e outras entidades que fazem parte da área comercial da música. Apesar de relevante, é um tema geralmente procurado por artistas e produtores mais avançados, não os que estão ainda começando.

O curso de composição de letras de música é de grande valia a qualquer artista, um aliado a qualquer um que queira aprender a colocar suas ideias, emoções e mensagens que quer passar em suas músicas, principalmente novos e inexperientes artistas.

O último é o curso de ML que já apareceu nos resultados do primeiro caso, e é um dos mais avaliados da Coursera. Pode ser esse o motivo de ele aparecer aqui, assim como, talvez, a falta de mais cursos focados em produção musical na mesma intensidade dos anteriores.

5.7.5 Outras métricas de distância

Ao utilizar outras métricas de distância além do cosseno, os resultados não foram satisfatórios. Mesmo utilizando todas as distâncias disponíveis pelo sklearn, como euclidiana, Manhattan, Minkowski, entre outras, e aplicando normalização nos dados, para todos os três cenários e até outros testes avulsos, os resultados foram parecidos, trazendo recomendações com pouca ou nenhuma relação ao curso de entrada. O exemplo a seguir ilustra uma dessas situações:

Métrica: Manhattan

Curso: 'python'

Resultados:

Entrepreneurship Strategy: From Ideation to Exit
Curanderismo: Traditional Healing Using Plants
Social and Economic Networks: Models and Analysis
Epigenetic Control of Gene Expression
Building Modern Python Applications on AWS

O exemplo ao menos apresentou um resultado satisfatório para o curso inicial, mas utilizando os outros dois cenários, os cursos retornados eram basicamente os mesmos. Há algumas explicações para o motivo disso acontecer, como a distância de cosseno lidar melhor com alta esparsidade, que o caso, também por ser a mais utilizada em modelos de filtragem colaborativa, já que cosseno foca na orientação dos vetores e não na magnitude. Um usuário pode dar apenas notas altas, e outro apenas notas baixas, mas mesmo assim ambos podem ter perfis semelhantes. Por isso pode-se usufruir melhor do que a distância de cosseno proporciona para esse tipo de filtragem.

6 Sistema de recomendação: filtragem baseada em conteúdo

Esta seção aborda o segundo sistema de recomendação na ordem do funcionamento do sistema proposto, com filtragem baseada em conteúdo, e todas as suas etapas. Cada subseção detalha e explica uma das etapas em todo o processo de geração das recomendações.

6.1 Extração de dados

Os maiores desafios da implementação da solução proposta já estão evidenciados desde o início. O *dataset* é, provavelmente, o recurso mais difícil de se obter, e também é o mais importante do projeto. É necessária uma base de dados que compreenda diversos tipos de alunos de cursos, com os mais variados gostos, padrões de aprendizado, objetivos, entre outros aspectos que serão relevantes para as recomendações. Muito diferente do dataset com dados dos cursos *online*, que são de fácil acesso e existem muitos disponíveis nas plataformas mais famosas de bases de dados. Uma outra barreira notável é o próprio desenvolvimento do sistema de recomendação, uma vertente muito famosa do ML e *data mining*, porém, a princípio, com poucos exemplos práticos e didáticos disponíveis em inglês ou português.

Durante a fase de desenvolvimento, foi determinado que o início se daria utilizando a abordagem da filtragem baseada em conteúdo para, primeiramente, obter um algoritmo de recomendação simples para servir como protótipo, utilizando como base de dados um *dataset* obtido no Kaggle, contendo muitos cursos da plataforma Udemy de *e-learning*.

Esse *dataset* [1], que continha mais de 200 mil registros sobre cursos dos mais variados temas e em todas as linguagens disponíveis na plataforma, recebeu um tratamento prévio e sofreu uma redução. As colunas consideradas relevantes para a recomendação foram *title* e *headline*, que são o título do curso e sua descrição, respectivamente.

O *dataframe* utilizado em todo o processo de recomendação é criado utilizando as duas colunas e criando uma nova coluna *text* contendo todo o texto relevante para as *queries*:

[1] https://www.kaggle.com/datasets/hossaingh/udemy-courses?select=Course_info.csv

id	title	headline	num_subscribers	average_rating
4715.0	Online Vegan Vegetarian Cooking School	Learn to cook delicious vegan recipes. Filmed ...	2231.0	3.75
1769.0	The Lean Startup Talk at Stanford E-Corner	Debunking Myths of Entrepreneurship A startup ...	26474.0	4.50
5664.0	How To Become a Vegan, Vegetarian, or Flexitarian	Get the tools you need for a lifestyle change ...	1713.0	4.40
7723.0	How to Train a Puppy	Train your puppy the right way with Dr. Ian Du...	4988.0	4.80
8157.0	Web Design from the Ground Up	Learn web design online: Everything you need t...	1266.0	4.75

Tabela 18 – Início do dataset com algumas das colunas (5 linhas)

Há também uma filtragem no *dataframe* pelos idiomas disponíveis, por meio da coluna *language* e mantendo apenas cursos em inglês, português e espanhol. Esse foi um dos primeiros problemas encontrados pelas recomendações de teste iniciais, pois os cursos levantados eram predominantemente cursos de idiomas asiáticos, como japonês e chinês, provavelmente por uma questão de ordenação de caracteres na criação do dicionário, já que esses idiomas utilizam alfabetos diferentes do romano.

6.2 Pré-processamento dos dados

Utilizando os recursos das bibliotecas NLTK[2] e Scikit Learn[3] do Python, a coluna *text* passa por um pré-processamento com remoção de *stopwords*, stemização com Porter Stemmer e outros tratamentos, que consistem em filtragens por expressões regulares para remover caracteres especiais, *tags* HTML e outros tipos de dígitos, assim como técnicas PLN para que se possa trabalhar com o texto em sua forma pura, e por fim, o texto todo é *tokenizado*.

[2] https://www.nltk.org/
[3] https://scikit-learn.org/

	text
0	onlin vegan vegetarian cook school learn cook ...
1	lean startup talk stanford ecorn debunk myth e...
2	becom vegan vegetarian flexitarian get tool ne...
3	train puppi train puppi right way dr ian dunba...
4	web design ground learn web design onlin every...

Tabela 19 – Coluna courses[‘text’] criada no dataframe após pré-processamento

6.3 Vetorização

Inicialmente, o desenvolvimento do algoritmo foi feito dentro da plataforma Google Collab[4], utilizando Jupyter Notebook. Para a vetorização e criação do vocabulário que terá a similaridade medida, foi definido a ser utilizado o TF-IDF[5], junto com CountVectorizer[6], na coluna *courses[‘text’]*.

A implementação consistia no carregamento por completo do *dataset* em memória, previamente à aplicação dos algoritmos. Por consequência de alguma mudança de política na plataforma, esse carregamento de dados foi alterado para apenas parte do *dataset*, e isso era um fator crucial para o funcionamento do sistema de recomendação em um computador convencional.

Para contornar as limitações de carregamento da plataforma, o código passou a ser desenvolvido e executado localmente, porém isso gerou uma nova situação, a execução de todas as funções que faziam parte da geração de recomendações se tornou pesada demais para a memória do computador. O *dataset* foi carregado por completo novamente, e isso impactou diretamente na etapa do TF-IDF.

O interpretador sempre lançava um erro exigindo quase 50GB de memória RAM para conseguir processar um vocabulário a partir de uma quantidade tão grande de dados, e isso fez com que a abordagem fosse deixada de lado.

Para contornar a situação, o Word2Vec[7], disponível pela biblioteca Gensim, mostrou-se de grande valia para realizar a vetorização de todos os tokens gerados nas etapas anteriores, fornecendo tudo que era necessário para os testes iniciais das primeiras recomendações, utilizando a filtragem baseada em conteúdo, como já apontado, os títulos e descrição dos cursos, comparando com termos de entrada do usuário.

Com os *tokens* gerados na etapa anterior, é criado o modelo de vetores com o Word2Vec para cada um dos cursos disponíveis no *dataframe*, utilizando novamente a coluna *courses[‘text’]*.

[4] https://colab.research.google.com/
[5] https://pt.wikipedia.org/wiki/Tf-idf
[6] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
[7] https://radimrehurek.com/gensim/models/word2vec.html

6.4 Obtenção do perfil

Na geração de recomendações feitas para o caso da recomendação baseada em conteúdo, é usado um sistema que recebe uma *query* com alguns termos que servem como base para recomendar novos cursos. A *query* é composta pela **área de interesse** e **nível de conhecimento** desejados, da seguinte forma:

área de interesse	nível de conhecimento
python	starter

A primeira parte da *query* diz respeito ao tipo de curso procurado pelo aluno. Caso a pessoa esteja procurando um curso sobre a linguagem de programação Python, essa parte terá palavras-chave que se refiram ao tema. Já na segunda parte, o aluno deve escrever termos que se refiram ao nível de conhecimento desejado. Nesse caso, o aluno quer um curso *starter*, ou seja, iniciante no assunto.

Esses são os dados de perfil coletados do usuário. Como já explicado, pela natureza do experimento não ser implementado em um sistema real, e pelos indivíduos que participam dos testes serem definidos pelo autor, o perfil de usuário será totalmente representado pela **consulta** composta pelas duas palavras-chave escolhidas, introduzidas à mão no *notebook* onde o código está disponível.

A consulta então é dividida também em *tokens*:

['python', 'starter']

Após isso, os *tokens* são utilizados para criar os vetores das palavras-chave, e parte dessas estruturas pode ser representada da seguinte forma:

[array([-1.316816 , -0.00765565, -1.5253502 , -0.4065882 , 1.1508659 , 0.69462216,
2.6009755 , 1.1592892 , -0.18154877, 0.10172073, ...)]

6.5 Similaridade

Em sequência da **vetorização**, baseado no modelo gerado pelo Word2Vec em cima da coluna *text*, um array de similaridades vazio é criado, e em um laço de repetição é calculada a similaridade para cada curso presente em *courses['text']*, que compreende o *ataframe* inteiro de cursos, e cada similaridade é introduzida ao *array* de similaridades:

title	similarity
Online Vegan Vegetarian Cooking School	0.254360
The Lean Startup Talk at Stanford E-Corner	0.295459
How To Become a Vegan, Vegetarian, or Flexitarian	0.240542

Tabela 20 – Relação de curso e similaridade calculada

Uma nova coluna *courses['similarity']* é criada no *dataframe* que recebe o *array*:

title	headline	text	similarity
Python for astronomers and astrophysics	Python Astronomy	python astronom astrophys python astronomi	0.975745
Python for Beginners	Python Programming/Scripting	python beginn python programmingscript	0.960129
Python for Beginners	Python 101	python beginn python	0.959776
Complete Python Programming-Python Basics to A...	Learn Python programming Python functions Pyth...	complet python programmingpython basic advanc ...	0.952554
Python from 2 to 3	Migrate your code from Python 2 to Python 3	python migrat code python python	0.949486

Tabela 21 – Novo dataframe com algumas colunas

6.6 Recomendações

Após a criação da coluna *similarity* dentro do *dataframe*, um segundo *dataframe* chamado **recommended_courses** é criado, ordenando o anterior pela coluna *similarity*.

A partir disso, são levantados os **cinco primeiros cursos** com a **maior similaridade** e retornados ao usuário, que nesse caso são as primeiras linhas do dataframe novo, recommended_courses.

O funcionamento completo do algoritmo de recomendação da filtragem por conteúdo é descrito pela figura a seguir:

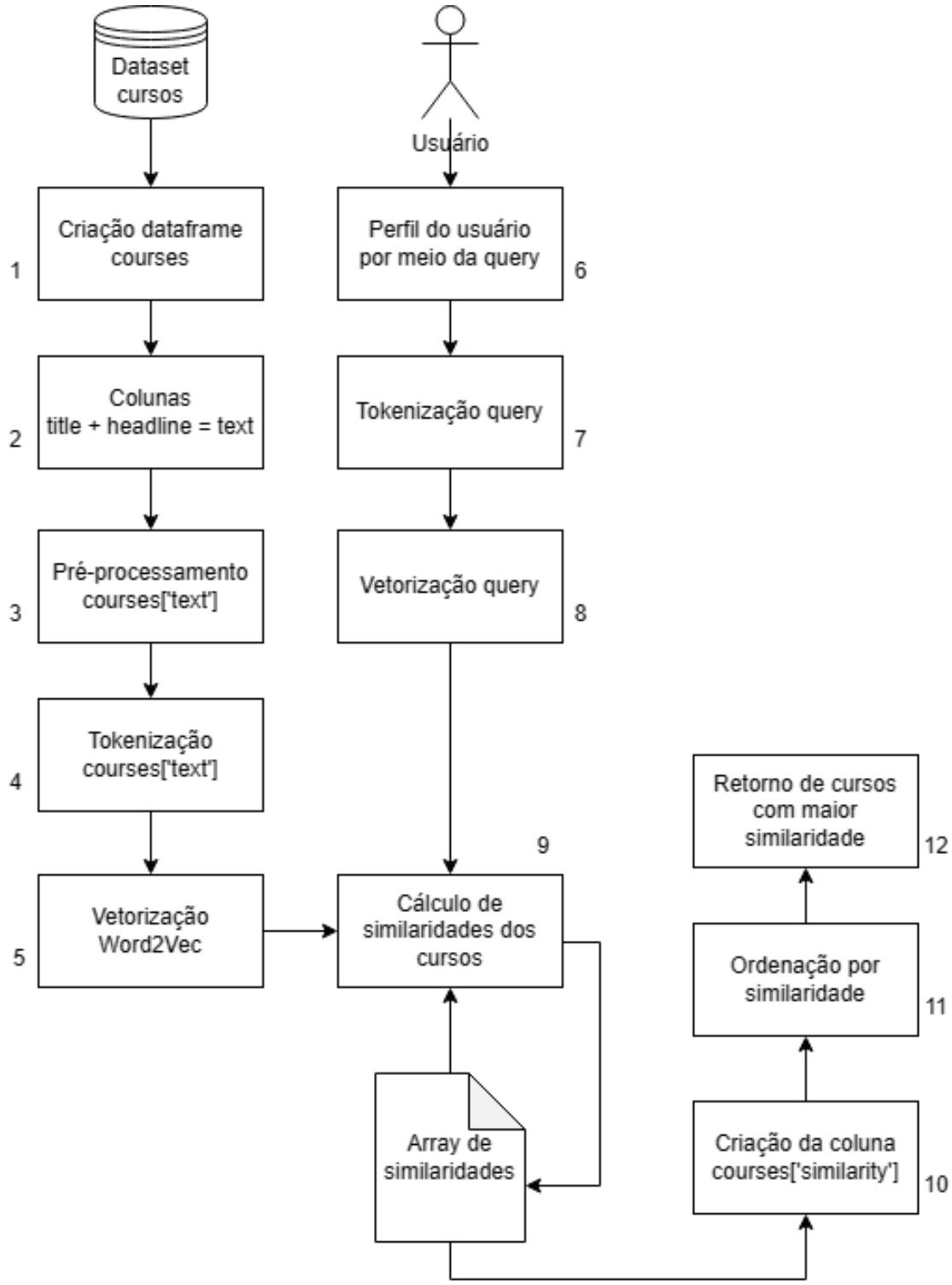

Figura 11 – O fluxo do sistema de recomendação baseado em conteúdo

6.7 Experimentos

O capítulo conduz, exemplifica e discute alguns cenários de testes para os experimentos realizados em cima das bases de dados utilizadas, e do sistema de recomendação proposto.

A primeira seção explica o contexto criado em torno dos experimentos, evidenciando diferentes cenários e situações do uso do sistema. Em seguida, de onde os dados são coletados para delinear os testes realizados. Após isso, as métricas utilizadas e os motivos são descritos. Por fim, a condução dos testes com o detalhamento completo e, ao final, uma discussão sobre os resultados obtidos.

6.7.1 Contexto e cenários

Como avaliação da filtragem baseada em conteúdo, três cenários baseados em uso real de uma plataforma por três indivíduos foram levantados. Os indivíduos do experimento contam com um contexto de vida profissional e pessoal e gostos pessoais diferentes, cada um em busca de um tipo específico de curso online para suprir seus interesses. Implementando o sistema proposto, foram definidos:

1. Um aluno do curso de Sistemas de Informação, em seu primeiro ano, em busca de um curso focado em uma linguagem de programação ou uma *stack* de desenvolvimento de *software* para iniciar sua carreira com um primeiro estágio na área;
2. Um profissional da área de design buscando um curso para ingressar na área de tecnologia como UX/UI;
3. Um entusiasta em busca de um curso para aprender um instrumento novo.

6.7.2 Origem dos dados

Para todos os usuários, a avaliação consiste na captação do perfil dele e na transformação em textos para servirem de termos de uma consulta para retornar cursos que satisfaçam aquela pesquisa com o motor de recomendação. Os textos são separados em *tokens* que participam do processo de vetorização. É criado um *array* de similaridades vazio que é populado com a similaridade calculada entre cada *token* do *dataframe* (coluna *courses['text']*) com cada token da consulta. Ao final, uma coluna *courses['similarity']* é criada, **ordenada por similaridade**, e os **cinco mais altos** no ranking são os **recomendados**.

Neste caso, vamos considerar que já foi realizada a extração e foram levantados os termos levando em conta:

- A área de interesse do curso procurado;

- O nível do curso desejado pelo usuário, se precisa ser para um iniciante ou um usuário avançado, por exemplo.

6.7.3 Métricas dos experimentos

Como métricas de avaliação, foram utilizadas: precisão, para medir a proporção de itens relevantes entre os recomendados e NDCG (*Normalized Discounted Cumulative Gain*), para o caso de a ordem das recomendações estar errada e itens relevantes aparecerem antes de itens relevantes. Nesse último caso, foi atribuído por análise própria do autor como 1 para o item relevante e zero para o item não relevante, para compor os cálculos do NDCG. A ordem ideal considerada para o cálculo do NDCG foi a própria ordem sugerida na seção de discussão de cada caso do experimento.

Foi escolhido manter as métricas do outro experimento para fins de comparação ao final do capítulo e conseguir esclarecer e evidenciar vantagens e desvantagens de um tipo de filtragem em relação ao outro.

6.7.4 Condução dos experimentos

CASO 1:

O aluno está em busca de um curso de programação para ingressar no mercado. Ele possui noções básicas de JavaScript, CSS e HTML e está querendo aprender um *framework* para desenvolvimento *web* e ouviu falar do VueJS.:

- Nome: Augusto
- Área de interesse: VueJS
- Nível de conhecimento: Iniciante

O perfil do usuário será representado por **'vue starter'**, sendo vue a área de interesse e starter (iniciante) o nível do público alvo do curso.

Tokens da query:

['vue', 'starter']

Utilizando como entrada **'vue starter'**, com a finalidade de encontrar um curso que ensine o básico do *framework* VueJS, teve como retorno o seguinte resultado:

Cursos recomendados:

Título: Complete Vuejs Course: Vue.js + Nuxt.js + PHP + Express.js

Descrição: VueJS, Command Line, Babel, NPM, Webpack, Vue JS CLI, Vue.js Router, VueX, Axios, iView, Express.js, Nuxt.js

Similaridade: 0.9106954336166382

Relevância: 1

Título: Webpack 4: Beyond the Basics

Descrição: Quick, code-driven, follow-along Javascript tutorials of Webpack, Babel, React, Angular, Vue, Redux, SSR, Typescript

Similaridade: 0.907133162021637

Relevância: 0

Título: React JS, Angular & Vue JS - Quickstart & Comparison

Descrição: Angular (Angular 2+), React or Vue? Get a Crash Course on each of them and a detailed comparison!

Similaridade: 0.9065245985984802

Relevância: 1

Título: Complete Vue.js 3 (Inc. Composition API, Vue Router, Vuex)

Descrição: (RE-RECORDED April 2021) Vue.js 3 is here! Learn from "Hello, Vue!"to building large apps with Vuex and Vue Router.

Similaridade: 0.9039604663848877

Relevância: 1

Título: React, NextJS and NestJS: A Rapid Guide - Advanced

Descrição: React with Typescript, Next.js, Redux, NestJS, Docker, Redis, Stripe, Frontend & Backend Filtering

Similaridade: 0.9017014503479004

Relevância: 0

Precisão	NDCG
3/5 = 60%	0,9060

Ordem ideal determinada: 1, 4, 3, 2 e 5

Discussão dos resultados:

O primeiro curso recomendado realmente é direcionado para iniciantes, com a apresentação do básico de uma *stack* inteira, não apenas o VueJS, para criar um projeto completo do zero, o que provavelmente é uma das tarefas desse curso.

O segundo curso já não é tão interessante por ser focado em Webpack, e o Vue é apenas uma parte dele, o que foge do objetivo inicial da procura.

O terceiro, apesar de não ser realmente o que o usuário está buscando, é um curso que apresenta comparativos entre alguns dos maiores *frameworks* de JS do mercado, o que pode dar ao usuário um quadro geral e uma visão mais completa sobre cada um, para que assim se decida melhor sobre qual deles escolher para utilizar em seus projetos.

O quarto é outra ótima recomendação, porque parece ser uma atualização do primeiro, dessa vez com a terceira versão do Vue, assim como algumas tecnologias de auxílio para o desenvolvimento utilizando essa ferramenta, como Vuex e Composition API.

O último é um curso de ReactJS, então não é exatamente o que o usuário está procurando, mas ainda está no mesmo grupo de tecnologias auxiliares para desenvolvimento web com JS.

CASO 2:

O profissional de design em busca de iniciar sua carreira na área de tecnologia como UX/UI. Ele possui conhecimento básico sobre Photoshop e Figma e quer aprender o básico necessário para uma vaga de entrada como UX:

- Nome: Pedro
- Área de interesse: Design
- Nível de conhecimento: Básico

O perfil do usuário será representado por **'ux basics'**, sendo ux a área de interesse e basics (básicos) o nível do público alvo do curso.

Tokens da query:

['ux', 'basics']

Utilizando como entrada **'ux basics'**, com a finalidade de encontrar um curso que ensine o básico sobre UX, teve como retorno o seguinte resultado:

Cursos recomendados:

Título: Wireframing For UI / UX Designers

Descrição: Simplify your UI / UX process with paper and digital wireframes

Similaridade: 0.828489363193512

Relevância: 0

Título: Lean UX: Faster, Better UX (User Experience) Design

Descrição: Design better and faster user experiences (UX) with Lean UX

Similaridade: 0.8075733780860901

Relevância: 0

Título: Adobe XD - UX UI design For beginners

Descrição: Adobe XD - UX UI Basics

Similaridade: 0.7924297451972961

Relevância: 1

Título: Adobe XD for Web Design: Essential Principles for UI UX

Descrição: Learn Responsive Web Design, UI UX Principles, Prepare The Design For Coding, Increasing Sales via A Great Design

Similaridade: 0.7726107835769653

Relevância: 1

Título: Adobe Illustrator CC Graphic Web Design: UI Logo Design

Descrição: Adobe Illustrator CC Graphic Design: Web Design, UI Design, Logo Design, Sketch UX Design For Designers Developers

Similaridade: 0.7722707986831665

Relevância: 1

Precisão	NDCG
3/5 = 60%	0,6183

Ordem ideal determinada: 3, 4, 5, 1 e 2

Discussão dos resultados:

O primeiro curso foca em *wireframing*, que é um dos processos principais de UX, mas não é abrangente para todos os conceitos básicos, o que pode ser menos interessante

para o usuário.

O segundo curso se encaixaria mais em conhecimentos avançados, para melhorar e otimizar o trabalho do profissional de UX.

O terceiro já apresenta um conteúdo focado em iniciantes, apresentando os básicos do Adobe XD.

O quarto curso é bem parecido com o terceiro, mesmos princípios.

O último é um curso sobre Illustrator CC, com vários conceitos e técnicas de web design utilizando a ferramenta, mas nada nele diz que é um curso iniciante, por isso é menos relevante.

CASO 3:

O entusiasta médio deseja aprender um instrumento novo, nesse caso o piano, e como é um instrumento mais complexo, resolveu pesquisar por algum curso que cubra os conhecimentos básicos para começar a tocar pequenas peças o mais rápido possível:

- Nome: Julio
- Área de interesse: Instrumentos
- Nível de conhecimento: Iniciante

O perfil do usuário será representado por **'piano iniciante'**, sendo piano a área de interesse e iniciante o nível do público alvo do curso.

Tokens da query:

['piano', 'iniciante']

Utilizando como entrada **'piano iniciante'**, com a finalidade de encontrar um curso com ensinamentos para principiantes no piano, teve como retorno o seguinte resultado:

Cursos recomendados:

Título: We ArtPlay Piano

Descrição: Composing and Improvising on the Piano

Similaridade: 0.977283239364624

Relevância: 0

Título: Piano With Willie: Piano Chords Vol. 1

Descrição: Learn to play chords on the piano

Similaridade: 0.9478051066398621

Relevância: 1

Título: Piano With Willie: Piano Chords Vol. 2

Descrição: Learn to play chords on the piano

Similaridade: 0.9478051066398621

Relevância: 1

Título: Guitar, Ukulele Piano for Praise Worship

Descrição: Playing for Worship

Similaridade: 0.9309934377670288

Relevância: 0

Título: O Hanon do Piano Popular

Descrição: Rítmica, Coordenação motora, Independência, Grooves

Similaridade: 0.9289443492889404

Relevância: 0

Precisão	NDCG
2/5 = 40%	0,5307

Ordem ideal determinada: 2, 3, 5, 4 e 1

Discussão dos resultados:

O primeiro resultado traz conhecimentos para pianistas mais avançados, como composição e improvisação, algo além do que busca um iniciante.

O segundo curso já conta com ensinamentos iniciais de acordes, ótimos para quem está começando.

O terceiro é como o segundo, mas é o próximo volume, podendo ser considerado um passo dois para o aprendizado de acordes.

O quarto resultado está trazendo uma gama de instrumentos, não apenas o piano, voltados para músicas de adoração, o que não é interessante para quem quer apenas aprender piano.

O último curso é o único em português, que foi o idioma utilizado nos termos de entrada do perfil do usuário, mas o conteúdo é para instrumentistas avançados, como grooves, aprendizado sobre ritmo, etc.

6.8 Comparativo dos sistemas

Como um comparativo do desempenho dos dois sistemas, mesmo que isolados e com bases de dados diferentes, a tabela a seguir apresenta os resultados:

Filtragem	Perfil	Precisão	NDCG
Conteúdo	'vue starter'	60%	0,9060
Conteúdo	'ux basics'	60%	0,6183
Conteúdo	'piano iniciante'	40%	0,5307
Colaborativa	'python'	80%	0,9438
Colaborativa	'everyday-parenting'	80%	0,8574
Colaborativa	'producing-music'	80%	0,9909

Tabela 22 – Comparativo dos sistemas de recomendação

Os maiores destaques para esse comparativo são o melhor desempenho da colaborativa, tendo as três maiores precisões, e dois dos maiores NDCG obtidos. Sobre o NDCG, porém, observa-se que o perfil 'vue starter' obteve um dos maiores. Existem algumas possíveis explicações para os resultados. Poucos cursos disponíveis na base de dados da colaborativa promovem poucas chances de um curso irrelevante ser recomendado. O mesmo vale para a baseada em conteúdo, já que muitos dados textuais podem exigir um tratamento mais minucioso para que os cursos recomendados sejam sempre 100% relevantes. Uma outra vantagem da filtragem colaborativa é priorizar cursos de notas mais altas, garantindo de uma forma melhor com que cursos melhores sejam recomendados e em uma ordem mais satisfatória.

7 Conclusão

É certo que na área de machine learning existem diversas abordagens diferentes para o mesmo objetivo, e é por isso que os resultados diferem tanto. Mesmo se for uma parte mais específica dessa área, como os sistemas de recomendação, a variedade de ferramentas, modos de se guiar e passos a seguir ainda é enorme. Nesse trabalho, a maioria das decisões foi tomada com base totalmente nas fontes de dados encontradas.

Depois de analisado o estado-da-arte, algumas implementações foram selecionadas dentre as que mais se destacaram, trazendo os resultados mais satisfatórios e mais recentes. Com destaque para a solução apresentada por [Vita e Cioffi 2022], com a filtragem híbrida, que conta com os dois maiores tipos de filtragem, um trazendo interações de usuários e o outro utilizando a PLN para reduzir descrições e textos em tokens e enriquecer as recomendações. Solução essa que resolve o problema de *cold start*, visto que quando o usuário é novo no sistema e não possui dados de interações, ainda há a filtragem por conteúdo para garantir que as recomendações sejam geradas.

As fontes de dados obtidas para desenvolver o trabalho foram satisfatórias, na medida do possível, mas poderiam ter sido melhor direcionadas e mais versáteis, para realizar mais do que o que foi feito. O primeiro dataset, de cursos da Udemy, é completo para realizar um trabalho de recomendação por conteúdo, mas não há disponível um segundo com dados de interações de usuário para realizar a recomendação colaborativa. Já o segundo dataset encontrado, da Coursera, que na verdade é um conjunto de dados de cursos e dados de avaliações, fornece muitos dados de interações entre usuários e cursos, mas não possui dados textuais para aplicar PLN na recomendação por conteúdo. Assim, cada um desempenhou muito bem seu papel, sendo cada uma das filtragens propostas, mas nenhum conseguiria realizar ambas, para que se pudesse até mesmo abordar a filtragem híbrida.

Na parte da PLN, como já foi explicado no parágrafo anterior, devido à natureza dos dados obtidos e à forma como funcionam as duas formas de filtragem estudadas e trabalhadas, as técnicas de PLN se fizeram relevantes apenas na filtragem baseada em conteúdo, pelo *dataset* utilizado conter os dados textuais que serviram de entrada para a geração das recomendações. Já no caso filtragem colaborativa, além de o *dataset* utilizado não conter colunas com dados textuais relevantes para o modelo, a abordagem escolhida elegeu apenas a coluna com os dados de avaliações numéricas como a entrada principal para o funcionamento do modelo, e também a única realmente relevante.

Durante o desenvolvimento, dois modelos diferentes de sistema de recomendação foram produzidos, cada um com características diferentes e específicas para cada caso, e

de acordo com cada base de dados utilizada.

O primeiro, um SR com filtragem baseada em conteúdo, onde os dados utilizados são os títulos e as descrições dos cursos que, após aplicação de PLN, compõem a base de dados vetoriais onde foi feito o cálculo de similaridade com o perfil de entrada do usuário. Já o segundo modelo utilizou filtragem colaborativa, por meio de dados de avaliações de usuários, com KNN retornando os cursos mais próximos do curso de entrada. A ideia inicial era uma espécie de sistema híbrido, consistindo da filtragem colaborativa como a principal, e a filtragem por conteúdo servindo como um refinamento das primeiras recomendações.

Os impedimentos para levar essa ideia adiante foram muitos, como inexperiência com o assunto, tempo curto, falta de base de dados de qualidade. O que levou a ideia a se transformar em outra, e o primeiro SR a servir como uma prática e uma demonstração de um sistema de recomendação mais "simples" e como PLN pode ser útil para aprimorar os resultados. Ao final, com a forma como foi conduzida e desenvolvida, a solução se assemelhou mais a um buscador, em que o usuário deliberadamente introduz um assunto que procura e o nível desejado, e recebe indicações de cursos com base na entrada. Ao passo que, o SR colaborativo é o que mais se parece com soluções vistas por toda a Internet hoje em dia, como Netflix, Amazon e outros serviços, em que os gostos de usuários de perfis semelhantes geralmente são parecidos. E mesmo com o perfil sendo captado à mão para realizar os experimentos aqui no trabalho, se o sistema for introduzido corretamente em um contexto real, como uma plataforma *online* de cursos, e feitos os ajustes necessários para que o perfil seja obtido da forma como já é (disponível pelos dados do usuário na plataforma), ele irá funcionar corretamente.

Com os experimentos realizados, a conclusão foi que os dois SR mostraram um resultado satisfatório, porém, com uma superioridade considerável para o colaborativo. Além do fato de que não há nenhuma garantia de que os cursos recomendados pelo SR de conteúdo são bons e bem produzidos, mas para o colaborativo há sim, já que são bem avaliados por usuários de perfis semelhantes. E com isso em mente, pode-se realizar um exercício de raciocínio. Quando se procura um curso *online*, as pessoas sempre priorizam a qualidade. Primeiro é feita a busca por um assunto, e depois de encontrar os cursos daquele tema, procura-se sempre o curso de melhor qualidade e mais bem avaliado. Dessa forma, a filtragem pelo assunto ocorre primeiro, e após isso é que entram as notas. E o tema que interessa um aluno geralmente é sempre o mesmo. De acordo com [Agarwal 2019], 75% das pessoas que estudam online estão buscando emprego ou mudança de carreira, por esse motivo faz sentido pensar que essas pessoas focam em uma área apenas, então provavelmente avaliam cursos sempre do mesmo tema. O que corrobora com o sucesso do SR colaborativo.

Ao levar em conta o primeiro SR baseado em conteúdo, um possível trabalho futuro, considerando os devidos contextos em que ele é inserido, seja uma plataforma *online*, ou

um *app*, por exemplo, para que que ele deixe de ser apenas um "buscador" e passe a realizar o trabalho completo de um SR do mercado, a base de dados deve ter, além das informações textuais de cada curso, os dados dos perfis de usuário para que sejam extraídos. Com isso, deixa de ser uma necessidade utilizar os dados anteriormente explicados, e se passa a ter disponível os perfis para gerar as recomendações.

Para a ideia dos dois SR se complementando, ou seja, o sistema híbrido, um possível trabalho futuro é, ao invés de utilizar PLN e dados textuais para fazer o refinamento das recomendações, utilizar esses dados para realizar uma amostragem inicial, retornando cursos apenas daquele tema, e com esses cursos filtrados, aplicar a filtragem colaborativa para priorizar apenas os mais avaliados e de maior qualidade, seguindo o mesmo princípio de primeiro escolher um tema e depois manter apenas os melhores cursos.

Em futuros experimentos, deve-se também priorizar uma forma mais tradicional para a realização dos testes com os modelos, retirando do conjunto de treinamento cursos bem avaliados para algumas áreas e assuntos diferentes e usá-los como conjunto de testes.

Algumas plataformas de cursos disponibilizam o conteúdo das aulas transcritos para os alunos. Uma ideia interessante para um futuro trabalho seria obter esses arquivos de texto transcritos e utilizá-los para treinar o modelo baseado em conteúdo, o que seria de grande valia para a aplicação das técnicas de PLN.

Referências

AAMIR, M.; BHUSRY, M. Recommendation system: state of the art approach. *International Journal of Computer Applications*, Citeseer, v. 120, n. 12, 2015. Citado na página 23.

AGARWAL, P. *Why is no one talking about employability in online education?* 2019. Disponível em: <https://blog.efmdglobal.org/2019/05/22/why-is-no-one-talking-about-employability-in-online-education/>. Citado na página 90.

BALUSH, I.; VYSOTSKA, V.; ALBOTA, S. Recommendation system development based on intelligent search, nlp and machine learning methods. In: *MoMLeT+ DS*. [S.l.: s.n.], 2021. p. 584–617. Citado 4 vezes nas páginas 17, 29, 30 e 31.

BROWN, S. *Machine learning, explained*. 2021. Disponível em: <https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained>. Citado 2 vezes nas páginas 24 e 32.

EDUCATION, I. C. *What is natural language processing?* 2020. Disponível em: <https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-ml/>. Citado na página 31.

KANADE, V. *What Is Machine Learning? Definition, Types, Applications, and Trends for 2022*. 2022. Disponível em: <https://www.ibm.com/cloud/learn/natural-language-processing>. Citado na página 32.

KAPOOR, N.; VISHAL, S.; KRISHNAVENI, K. Movie recommendation system using nlp tools. In: IEEE. *2020 5th International Conference on Communication and Electronics Systems (ICCES)*. [S.l.], 2020. p. 883–888. Citado na página 49.

KRAMER, O. K-nearest neighbors. In: _____. *Dimensionality Reduction with Unsupervised Nearest Neighbors*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013. p. 13–23. ISBN 978-3-642-38652-7. Disponível em: <https://doi.org/10.1007/978-3-642-38652-7_2>. Citado na página 32.

KUMAR, S. *Fundamental of Matrix Factorization For Recommender System*. 2021. Disponível em: <https://www.linkedin.com/pulse/fundamental-matrix-factorization-recommender-system-saurav-kumar/>. Citado na página 33.

LU, J. et al. Recommender system application developments: a survey. *Decision Support Systems*, Elsevier, v. 74, p. 12–32, 2015. Citado na página 24.

MUREL, J.; KAVLAKOGLU, E. *What is content-based filtering?* 2024. Disponível em: <https://www.ibm.com/topics/content-based-filtering>. Citado 2 vezes nas páginas 29 e 30.

NAREN, J.; BANU, M. Z.; LOHAVANI, S. Recommendation system for students' course selection. In: *Smart Systems and IoT: Innovations in Computing*. [S.l.]: Springer, 2020. p. 825–834. Citado 4 vezes nas páginas 15, 42, 43 e 44.

RAMAN, K. *Matrix Factorization Explained | What is Matrix Factorization?* 2024. Disponível em: <https://www.mygreatlearning.com/blog/matrix-factorization-explained/>. Citado na página 33.

SHARMA, A. K. et al. An efficient approach of product recommendation system using nlp technique. *Materials Today: Proceedings*, Elsevier, 2021. Citado na página 48.

TORFI, A. et al. Natural language processing advancements by deep learning: A survey. *arXiv preprint arXiv:2003.01200*, 2020. Citado na página 23.

VITA, D. L.; CIOFFI, G. Hybrid movie recommender system. 2022. Citado 7 vezes nas páginas 15, 30, 35, 36, 38, 39 e 89.

WANG, Z. et al. Using natural language processing techniques to provide personalized educational materials for chronic disease patients in china: development and assessment of a knowledge-based health recommender system. *JMIR medical informatics*, JMIR Publications Inc., Toronto, Canada, v. 8, n. 4, p. e17642, 2020. Citado 3 vezes nas páginas 15, 44 e 47.

WOLFF, R. *Top 10 NLP Tools and Services in 2022*. 2020. Disponível em: <https://monkeylearn.com/blog/natural-language-processing-tools/>. Citado na página 24.

YADALAM, T. V. et al. Career recommendation systems using content based filtering. In: IEEE. *2020 5th International Conference on Communication and Electronics Systems (ICCES)*. [S.l.], 2020. p. 660–665. Citado na página 48.

Apêndices

APÊNDICE A – Código-fonte

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

n_ratings = len(ratings)
n_courses = ratings['course_id'].nunique()
n_users = ratings['reviewers'].nunique()

print(f'N mero de avalia   es: {n_ratings}')
print(f'N mero de courseIds  nicos  : {n_courses}')
print(f'N mero de usu rios  nicos  : {n_users}')
print(f'Average number of ratings per user: {round(n_ratings/n_users, 2)}')
print(f'Average number of ratings per course: {round(n_ratings/n_courses,
    ↪ 2)}')

sns.countplot(x='rating', data=ratings, palette='viridis')
plt.title('Distribui   o de avalia   es de cursos', fontsize=14)

course_ratings = pd.merge(courses, ratings, on='course_id', how='left')
course_ratings['name'].value_counts()[0:10]

reviewerIds = {reviewer: idx for idx, reviewer in enumerate(ratings['
    ↪ reviewers'].unique())}
ratings['userId'] = ratings['reviewers'].map(reviewerIds)
ratings.rename(columns={'course_id': 'courseId'}, inplace=True)
ratings = ratings[['userId', 'courseId', 'rating']]

ratings.head()

new_ratings = ratings.drop_duplicates(subset=['userId', 'courseId'])
print("\nDataFrame ap s a remo   o de duplicatas:")
print(new_ratings)

from scipy.sparse import csr_matrix

def create_X(df):
    '''
    gera uma matriz esparsa com o dataframe de avalia   es

    args (df): dataframe pandas com 3 colunas (userId, courseId, rating)

    retorno:
```

```
42          X: matriz
43          user_mapper: dicion rio que mapeia ids de usu rios para  ndices
44          user_inv_mapper: dicion rio que mapeia  ndices  para ids
45          course_mapper: dicion rio que mapeia ids de cursos para  ndices
46          course_inv_mapper: dicion rio que mapeia  ndices  para ids
47      '''
48      M = df['userId'].nunique()
49      N = df['courseId'].nunique()
50
51      user_mapper = dict(zip(np.unique(df['userId']), list(range(M))))
52      course_mapper = dict(zip(np.unique(df['courseId']), list(range(N))))
53
54      user_inv_mapper = dict(zip(list(range(M)), np.unique(df['userId'])))
55      course_inv_mapper = dict(zip(list(range(N)), np.unique(df['courseId']))
        ↪ )
56
57      user_index = [user_mapper[i] for i in df['userId']]
58      item_index = [course_mapper[i] for i in df['courseId']]
59
60      X = csr_matrix((df['rating'], (user_index,item_index)), shape=(M,N))
61
62      return X, user_mapper, course_mapper, user_inv_mapper,
        ↪ course_inv_mapper
63
64 X, user_mapper, course_mapper, user_inv_mapper, course_inv_mapper =
       ↪ create_X(new_ratings)
65
66 n_total = X.shape[0]*X.shape[1]
67 n_ratings = X.nnz
68 sparsity = n_ratings/n_total
69 print(f'Esparsidade: {round(sparsity*100,2)}%')
70
71 n_ratings_per_user = X.getnnz(axis=1)
72
73 n_ratings_per_course = X.getnnz(axis=0)
74 len(n_ratings_per_course)
75
76 print(f'O usu rio mais ativo avaliou {n_ratings_per_user.max()} cursos.')
77 print(f'O usu rio menos ativo {n_ratings_per_user.min()} cursos.')
78
79 print(f'O curso mais avaliado tem {n_ratings_per_course.max()} avalia   es
       ↪ .')
80 print(f'O curso menos avaliado possui {n_ratings_per_course.min()}
       ↪ avalia   es.')
81
82 from sklearn.neighbors import NearestNeighbors
83
```

```
84 def find_similar_courses(course_id, X, course_mapper, course_inv_mapper, k,
       ↪ metric='cosine'):
85     '''
86     encontra os k vizinhos para um dado id de curso
87 
88     args:
89         course_id: id do curso para o perfil interessado
90         X: matriz de utilidade user-item
91         k: n mero de cursos similares a serem retornados
92         metric: m trica para a dist ncia do kNN
93 
94     sa da: retorna uma lista de k ids de cursos similares
95     '''
96     X = X.T
97     neighbour_ids = []
98 
99     course_ind = course_mapper[course_id]
100    course_vec = X[course_ind]
101    if isinstance(course_vec, (np.ndarray)):
102        course_vec = course_vec.reshape(1,-1)
103 
104    # k+1 para excluir o id de interesse utilizado
105    kNN = NearestNeighbors(n_neighbors=k+1, algorithm="brute", metric=
       ↪ metric)
106    kNN.fit(X)
107    neighbour = kNN.kneighbors(course_vec, return_distance=False)
108    for i in range(0,k):
109        n = neighbour.item(i)
110        neighbour_ids.append(course_inv_mapper[n])
111    neighbour_ids.pop(0)
112    return neighbour_ids
113 
114 similar_courses = find_similar_courses('python', X, course_mapper,
       ↪ course_inv_mapper, k=6)
115 similar_courses
116 
117 course_titles = dict(zip(courses['course_id'], courses['name']))
118 
119 course_id = 'producing-music'
120 
121 new_similar_courses = find_similar_courses(course_id, X, course_mapper,
       ↪ course_inv_mapper, k=6)
122 course_title = course_titles[course_id]
123 
124 print(f'Porque voc  avaliou {course_title}:')
125 for i in new_similar_courses:
126     print(course_titles[i])
```

```
127
128 from sklearn.decomposition import TruncatedSVD
129
130 svd = TruncatedSVD(n_components=30, n_iter=40)
131 Q = svd.fit_transform(X.T)
132 Q.shape
133
134 course_id = 'python'
135 this_similar_courses = find_similar_courses(course_id, Q.T, course_mapper,
    ↪ course_inv_mapper, metric='cosine', k=6)
136 course_title = course_titles[course_id]
137
138 print(f"Because you reviewed {course_title}:")
139 for i in this_similar_courses:
140     print(course_titles[i])
```

colaborativa.txt

```
1 import numpy as np
2 import pandas as pd
3
4 # Lista de idiomas permitidos
5 idiomas_permitidos = ['Portuguese', 'English', 'Spanish']
6
7 # Filtrar cursos com idiomas permitidos
8 courses = courses[courses['language'].isin(idiomas_permitidos)]
9
10 # Filtrar apenas cursos com no m nimo 1000 reviews
11 courses_filtered = courses.loc[courses['num_reviews'] > 500]
12
13 # Agora, 'cursos_filtrados' cont m apenas cursos em portugu s, ingl s e
    ↪ espanhol
14 courses_filtered
15
16 courses_filtered = courses_filtered[['id', 'title', 'headline', 'avg_rating
    ↪ ', 'num_reviews']]
17
18 # pre processing data
19 import re
20 import nltk
21
22 nltk.download('punkt')
23 nltk.download('averaged_perceptron_tagger')
24 nltk.download('stopwords')
25
26 from nltk.corpus import stopwords
27 from nltk.stem.porter import PorterStemmer
28
```

```
stop_words = set(stopwords.words('english'))
porter_stemmer = PorterStemmer()

from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

def pre_process(text):
    # Converte o texto para minsculas
    text = text.lower()
    # Remove tags HTML usando express o regular
    text = re.sub(r'<.*?>', ' ', text)
    # Remove d gitos e caracteres n o alfab ticos, mantendo apenas
    ↪ letras e espa os
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    # Tokenize o texto usando NLTK
    words = nltk.word_tokenize(text)
    # Remove palavras stopwords
    words = [word for word in words if word not in stop_words]
    # Aplica stemming s palavras restantes
    stemmed_words = [porter_stemmer.stem(word) for word in words]
    # Reconstr i o texto
    new_text = ' '.join(stemmed_words)
    return new_text

courses_filtered['text'] = courses_filtered[['title', 'headline']].astype(
    ↪ str).agg(' '.join, axis=1)
courses_filtered['text'] = courses_filtered['text'].astype(str).apply(
    ↪ pre_process)
courses_filtered['text']

new_courses = courses_filtered.reset_index(drop=True)

# vocabulary and word counts
from sklearn.feature_extraction.text import CountVectorizer

docs = new_courses['text'].tolist()

cv = CountVectorizer(max_df=0.85, stop_words=stopwords.words('english'))
word_count_vector = cv.fit_transform(docs)

from gensim.models import Word2Vec

documents = [doc.split() for doc in new_courses['text']]

model = Word2Vec(sentences=documents, vector_size=100, window=5, min_count
    ↪ =1, sg=0)
```

```
72
73 user_query = "angular starter"
74 user_query_tokens = user_query.split()
75
76 word_vectors = [model.wv.get_vector(word) for word in user_query_tokens if
       ↪ word in model.wv]
77
78 similarities = []
79 for course_text in new_courses['text']:
80     course_words = course_text.split()
81     if not course_words:
82         similarities.append(0.0)  # 0 de similaridade para cursos sem
       ↪ palavras
83     else:
84         similarity = model.wv.n_similarity(user_query_tokens, course_words)
85         similarities.append(similarity)
86
87 new_courses
88
89 import pandas as pd
90
91 # adiciona as similaridades ao dataframe de cursos
92 new_courses['similarity'] = similarities
93
94 # ordena os cursos com base nas similaridades em ordem decrescente
95 recommended_courses = new_courses.sort_values(by='similarity', ascending=
       ↪ False)
96
97 # seleciona os cursos mais similares
98 top_recommendations = recommended_courses.head(5)
99
100 # Exiba os cursos recomendados
101 print("Cursos recomendados:")
102 for index, row in top_recommendations.iterrows():
103     print(f"T tulo: {row['title']}")
104     print(f"Descri   o: {row['headline']}")
105     print(f"Similaridade: {row['similarity']}")
106     print()
```

conteudo.txt

APÊNDICE B – Artigo Formato SBC

Aplicação de PLN na recomendação de conteúdo educacional

Marcelo M. V. Filho[1], **Elder Rizzon Santos**[1]

[1]Universidade Federal de Santa Catarina

Abstract. *Several services have their own recommender system, which essentially presents people products or services that they might want to consume, in a determined list utilizing different techniques, which is based on previous consumption data. This can be seen by receiving suggestions for new movies on a streaming service or suggestions for new products on an online store. NLP, or natural language processing, a branch of Artificial Intelligence that assists in machine understanding of human-generated data, is a useful and powerful tool for capturing user's tastes and preferences. In the academic world, students often find themselves lost with so much educational content available and are unable to chart a clear and well-defined study path to achieve their goals, as a student or professional, amidst this plethora of materials. Using NLP techniques, this work proposes to outline a RS model fed by a user database that recommends items according to the user's profile.*

Resumo. *Muitos serviços contam com seu próprio sistema de recomendação, que essencialmente apresenta às pessoas produtos ou serviços que elas possam querer consumir, em uma lista determinada utilizando diferentes técnicas, que tem como base dados prévios de consumo. É possível observar isso ao receber sugestões de novos filmes em um serviço de streaming ou sugestões de novos produtos em uma loja online. O PLN, ou processamento de linguagem natural, uma vertente da Inteligência Artificial que atua na assistência da compreensão de dados gerados por humanos pelas máquinas, é uma ferramenta útil e poderosa para captar os gostos e preferências do usuário. No mundo acadêmico, os alunos se encontram muitas vezes perdidos com tanto conteúdo educacional disponível e não conseguem traçar uma rota de estudos clara e bem definida para atingir seus objetivos, como estudante ou profissional, em meio a essa infinidade de materiais. Utilizando técnicas de PLN, o trabalho se propõe a delinear um modelo de SR alimentado por uma base de dados de usuários que recomende materiais aos alunos de acordo com o perfil do usuário.*

1. Introdução

Em frente a uma grande quantidade de conteúdo disponível, as pessoas muitas vezes se encontram perdidas e empregam muito tempo escolhendo um conteúdo dentre o leque infinito de possibilidades, como explicam [Aamir and Bhusry 2015]. Por essa e várias outras razões os sistemas de recomendação são uma necessidade no contexto atual do mundo. Um sistema de recomendação utiliza da combinação de técnicas de Machine learning (aprendizagem de máquina) e uma base de dados fortemente alimentada, porque é necessário que o algoritmo aprenda e se desenvolva a partir dos gostos, interesses e características de cada usuário e faça suposições sobre isso. De acordo com [Lu et al. 2015], os SR são programas que tentam recomendar os itens (serviços ou produtos) mais relevantes para determinados usuários (empresas ou indivíduos), utilizando da predição do

interesse do usuário em um item, baseado em informações relacionadas aos itens, usuários e interações entre itens e usuários. O objetivo é direcionar o conteúdo ao usuário com base no que mais condiz com seu perfil.

Machine learning, de acordo com [Brown 2021], é uma subárea da inteligência artificial, amplamente definida como a capacidade de uma máquina imitar o comportamento da inteligência humana. Sistemas de inteligência artificial são usados para realizar complexas tarefas de uma maneira similar à maneira que humanos resolvem problemas.

Hoje em dia existem diversas soluções que utilizam PLN em suas funcionalidades, como o IBM Watson, Amazon Comprehend, MonkeyLearn e Aylien [Wolff 2020]. O PLN pode ser inserido em vários contextos para resolver problemas como análise de sentimento em textos online, transformação de texto em fala e detecção de urgência em textos.

Em meio a tantos materiais disponíveis gratuitamente por toda a Internet, o acadêmico, hoje em dia, por muitas vezes se encontra perdido e sem uma rota de estudos traçada para atingir seus objetivos. Isso é ainda mais forte na área de computação, em que o profissional deve manter-se sempre atualizado com o que está mais forte no mercado e procurar se especializar em determinados assuntos e tecnologias, ao invés de tentar entender de tudo. Qual linguagem de programação, framework, ferramenta ou tecnologia aprender e direcionar seu foco e esforços é uma incógnita para muitos.

1.1. Objetivos

O objetivo é aplicar técnicas de PLN para treinar um modelo de sistema de recomendação de conteúdos educacionais *online*, com possibilidade de aplicação em livros ou cursos *e-learning*, para alavancar o aprendizado de usuários que procuram orientação numa infinidade de materiais disponíveis na Internet. Para atingir o objetivo geral, pretende-se cumprir os seguintes objetivos específicos:

- Analisar estado da arte de SRs que utilizam PLN;
- Definir os requisitos para uma solução desse contexto;
- Obter uma fonte de dados sobre cursos e material didático;
- Extrair e tratar os dados obtidos com técnicas de PLN;
- Desenvolver um sistema de recomendação com base nesses dados;
- Delinear um experimento para avaliar o protótipo do SR;
- Analisar e avaliar o SR com base no experimento.

2. Metodologia

A metodologia de pesquisa e desenvolvimento deste trabalho dividiu-se nas seguintes etapas:

- **Etapa 1:** Pesquisa bibliográfica e levantamento dos assuntos necessários para desenvolver a aplicação, investigando o estado da arte em sistemas de recomendação, em sua maioria os que utilizam PLN. A pesquisa foi realizada em artigos, documentação das ferramentas, *surveys* e outras publicações relacionadas ao tema;
- **Etapa 2:** Análise e definição dos requisitos necessários para a construção de uma solução dessa natureza. Também foram consideradas pesquisas acerca das tecnologias a serem utilizadas e diferentes técnicas para atuarem em todas as etapas da recomendação;

- **Etapa 3:** Pesquisa sobre bases de dados robustas e relevantes ao tema, para que possa alimentar o SR e gerar recomendações precisas.
- **Etapa 4:** Extração e tratamento dos dados obtidos pelos *datasets* com técnicas de PLN;
- **Etapa 5:** Desenvolvimento de um protótipo do SR com base nos dados;
- **Etapa 6:** Análise e avaliação do SR a partir de um experimento.
- **Etapa 6:** Documentar e relatar todo o procedimento do trabalho para as disciplinas de Introdução a Projetos, Projetos 1 e Projetos 2.

3. Fundamentação Teórica

Este capítulo lista os conceitos necessários para o entendimento e o desenvolvimento do trabalho. Serão apresentados alguns assuntos relevantes a sistemas de recomendação e processamento de linguagem natural, as bases do presente trabalho.

3.1. Sistemas de Recomendação

Algoritmos com o objetivo de predizer a relevância de um item e, com base nisso, realizar sugestões ao usuário são os chamados sistemas de recomendação. Essas sugestões podem partir de características de itens similares ou de avaliações feitas por usuários com perfis parecidos com o do usuário em particular. A maioria dos serviços prestados na Internet, hoje em dia, conta com seu próprio sistema de recomendação. *Streaming* (Netflix, HBO Max e Disney+), plataformas de vídeos (YouTube), *e-commerce* (Amazon) e redes sociais (Instagram).

Esses sistemas funcionam baseados em algoritmos de *machine learning* que fazem a extração de uma base de dados, tratam os dados e definem padrões de comportamento de usuários. Podem ser classificados quanto ao tipo de filtragem utilizado, seja baseada em conteúdo, colaborativa ou híbrida.

3.2. Filtragem baseada em conteúdo

A filtragem baseada em conteúdo utiliza a similaridade de vetores para recomendar itens a partir de features, ou seja, características extraídas da base de dados, para uma determinada entrada desejada.

Há um processo de examinação do perfil de um novo usuário quanto a seus interesses, baseado nas funções disponíveis nos objetos que o usuário avaliou. É um sistema de recomendação para palavras-chave específicas. Esse tipo de filtragem utiliza algoritmos que recomendam itens semelhantes aos que o usuário gosta [Balush et al. 2021].

Filtragem baseada em conteúdo é um método de obtenção de informação que usa características de itens para selecionar e retornar itens relevantes a uma consulta do usuário. Este método muitas vezes toma características de outros itens em que o usuário expressou interesse [Murel and Kavlakoglu 2024].

3.3. Filtragem colaborativa

No caso da filtragem colaborativa, a similaridade é calculada baseada em preferências de usuários com perfis e gostos parecidos para, após reunir várias informações, realizar recomendações de determinados itens sem que o usuário tenha interagido com qualquer um daqueles itens.

Sistemas de recomendação que sumarizam avaliações de itens ou recomendações, reconhece semelhanças entre usuários, baseado em suas avaliações, e gera novas recomendações baseadas em comparações entre usuários. Funciona bem no caso de objetos complexos, onde variações de gosto são responsáveis pela maior parte das mudanças de preferências. Essa filtragem é baseada na situação de assumir que pessoas que concordaram no passado, decidirão no futuro [Balush et al. 2021].

Esta abordagem aproxima grupos de usuários a grupos distintos baseando-se em seus comportamentos, o que algoritmos como o KNN (*K-nearest Neighbors*) fazem, retornando os itens mais próximos com base em um número determinado.. Usando características gerais dos grupos, ela retorna itens específicos para um grupo inteiro pelo princípio de que usuários similares se interessam por itens similares [Murel and Kavlakoglu 2024]

3.4. Filtragem híbrida

Os sistemas híbridos unem a exploração de dados de interação entre usuários e itens (e metadados de usuários e itens). Assim, o sistema híbrido é o melhor considerado a lidar com diversos tipos de problemas com os quais os outros tipos de filtragem têm dificuldades, já que pode contar com as informações de metadados quando não há dados de interações disponíveis [Vita and Cioffi 2022].

Combinando ambos os tipos de sistema, essa filtragem fornece recomendações comparando os hábitos de consumo e pesquisa de usuários semelhantes, assim como gera recomendações de itens com características semelhantes a itens bem avaliados pelos usuários [Balush et al. 2021].

3.5. Processamento de Linguagem Natural

O processamento de linguagem natural é uma área dentro da Inteligência Artificial que atua como uma ponte entre a linguagem natural, isto é, a linguagem que os humanos entendem, para a linguagem entendida pelo computador. O PLN é responsável por extrair informações a partir do contato com usuários e permitir que a máquina compreenda o que foi escrito pelos seres humanos.

Entre os usos do PLN é possível citar as aplicações que dependem que o contexto seja explorado e entendido perfeitamente, como interpretação de textos, análise de sentimentos, tradução de acordo com significado semântico, etc. Plataformas de busca, como o Google, utilizam diversas técnicas de PLN para entender rapidamente o que o usuário está querendo dizer ao buscar um termo ou uma frase. As assistentes virtuais, muito fortes hoje em dia, e cada vez mais se tornando parte da vida das pessoas, realizam tarefas e atendem a comandos diariamente aplicando PLN em seu funcionamento [Education 2020].

3.6. *Machine learning*

Machine learning é uma subárea da Inteligência Artificial, amplamente definida como a capacidade de uma máquina imitar comportamentos inteligentes do ser humano. Sistemas de IA são usados para realizar tarefas complexas de uma maneira similar à que humanos resolvem problemas.

No *machine learning* os dados são coletados e preparados para serem usados como dados de treinamento ou as informações nas quais o modelo de aprendizado de máquina será treinado. Quanto mais dados, melhor o programa [Brown 2021].

Suas aplicações são diversas, como os próprios sistemas de recomendação, carros com piloto automático (*Tesla*), algoritmos de análise de imagens. Muito forte e presente na tomada de decisões, é uma grande aliada de diversas empresas atualmente, principalmente durante as decisões mais arriscadas, no direcionamento de operações de negócios e nos possíveis caminhos que a empresa pode tomar [Kanade 2022].

3.7. K-nearest Neighbors

É um algoritmo de classificação de aprendizado supervisionado muito utilizado na área de *machine learning* e *data mining* para resolver problemas de classificação e regressão. Ele utiliza métricas de distância entre vetores para classificar grupos de dados.

A classificação por *nearest neighbor*, também conhecida por KNN, é baseada na ideia de que os padrões mais próximos de um padrão X alvo, o qual procura-se classificar, entrega informações úteis de classificação. O KNN atribui o rótulo da classe à maioria dos K mais próximos padrões em dados dimensionais [Kramer 2013].

3.8. Matrizes item-item e usuário-item

São matrizes cujas dimensões podem representar usuários e itens ou itens e itens. Em cada registro entre um usuário ou item x com um item y há uma pontuação em relação à interação entre eles, seja uma avaliação ou uma revisão.

3.9. Fatoração de matriz

Fatoração de matriz é um modelo matemático que ajuda o sistema a dividir uma entidade em múltiplas entradas menores, por meio de um conjunto retangular de números ou funções ordenadas, a fim de descobrir *features* ou informações ocultas nas interações entre usuários e itens [Raman 2024].

É uma técnica extensivamente utilizada em SRs de filtragem colaborativa. Seu objetivo é fatorar uma matriz usuário-item em duas matrizes de ranque menor, a matriz de usuário-fator e a matriz de item-fator, para que possam predizer novos itens nos quais os usuários possam se interessar [Kumar 2021].

4. Trabalhos Correlatos

Com o objetivo de realizar uma busca por trabalhos que tratassem de sistemas de recomendação utilizando PLN e relacionados, foi utilizada a ferramenta Google Scholar, um indexador de artigos científicos. Para levantar o estado-da-arte, assim como outras soluções relevantes ao tema, foram utilizados termos de busca como *"recommender systems nlp"*, *"recommendation systems nlp"*, *"recommendation machine learning nlp"* e *"recommendation systems"*. Dentre os artigos encontrados, foram avaliados e selecionados os três melhores e mais relacionados com o presente trabalho para a primeira seção, e outros três servindo como mais exemplos de trabalhos com relação ao tema.

4.1. Hybrid Movie Recommender System

Em sua tese, [Vita and Cioffi 2022] decidem resolver o problema da quantidade massiva de informação disponível *online* com um sistema de recomendação híbrido, defendendo que a forma como eles funcionam pode melhorar muito a qualidade das recomendações com relações usuário-item muito mais complexas e melhorias no problema de partida a frio dessas relações, comparando com os métodos baseados em conteúdo e filtragem colaborativa.

O modelo híbrido teve uma performance muito superior ao colaborativo, especialmente em precisão. Ele se provou menos provável a ter *overfit* nos dados de treino. A única desvantagem encontrada em inserir metadados de itens no LightFM foi o aumento no tempo de treinamento e predição. Também foi inspecionado o quanto o tamanho do *dataset* influenciava na performance dos dados de teste. O Precision@K mostrou direta proporcionalidade com o crescimento dos dados de treino, ou seja, quanto mais filmes um usuário assistisse e avaliasse, melhor seria a acurácia das recomendações. Foi realizado então o teste com ambos metadados de itens e usuários, mas não se obteve uma mudança considerável nos resultados. No fim, o modelo com os dois tipos de metadados não superou o modelo que utilizava apenas os de itens.

Em conclusão, o autores reafirmam que o propósito da tese era investigar tipos diferentes de sistemas de recomendação e construir um sistema híbrido capaz de explorar os pontos mais positivos deles, tentando eliminar as desvantagens que eles trazem. Também foi um objetivo estudar como a linguagem natural poderia ser explorada para aumentar a qualidade das sugestões e melhorar diversos problemas.

4.2. Recommendation System for Students' Course Selection

O trabalho de [Naren et al. 2020] tem como objetivo ajudar estudantes de graduação a escolherem novas disciplinas a cada semestre, levando em conta suas próprias capacidades, tendências do mercado e outros fatores, utilizando para isso técnicas de processamento de linguagem natural e mineração de dados, em conjunto com scripts de *frontend* e *backend* escritos em Python e hospedados na Internet. *Data mining*, também, segundo os autores, popularmente conhecida como *knowledge discovery from data* (KDD), foi utilizada para coletar e analisar dados de estudantes sobre cursos finalizados junto com suas notas. O PLN, ou processamento de linguagem natural, foi utilizado na mineração e tratamento textual para o entendimento da máquina, com técnicas como tokenização, remoção de *stop words*, stemização, lematização, entre outras.

O modelo proposto por [Naren et al. 2020] conta com a construção de uma plataforma para os estudantes realizarem a escolha dos cursos, baseada em disciplinas anteriores e notas obtidas. As notas indicam o nível de aptidão do estudante para uma disciplina. É feito um cálculo da similaridade entre as disciplinas finalizadas e as que podem ser escolhidas. As cinco primeiras do ranking com a maior similaridade são tomadas e o valor cumulativo de nota delas é calculado. A partir disso, as duas disciplinas com o maior valor calculado são recomendadas ao estudante. Foi utilizado o *framework* de código aberto Flask, que é um serviço *web* para o Python, com requisições HTTP feitas por meio de API REST e *template* HTML tratado com Jinja 2.

De acordo com os autores, os coeficientes cosseno e Sørensen–Dice obtiveram os melhores resultados, dentre os demais, dado o contexto e o uso específico em documen-

tos textuais. Os autores apresentam os resultados de uma amostra teste da computação dos coeficientes de similaridade, de diferentes tipos, verificando a similaridade entre 11 disciplinas com a disciplina de Python. Comparando todas as formas de medida, como já previamente constatado, os coeficientes de Dice e cosseno obtiveram os melhores resultados, sendo Dice declarado o melhor. Outras formas de medida como distância Euclidiana, distância Manhattan e coeficiente de Pearson não se encaixaram no contexto do sistema e foram desconsideradas.

4.3. Using Natural Language Processing Techniques to Provide Personalized Educational Materials for Chronic Disease Patients in China

Utilizando materiais educacionais para instruir os pacientes chineses disponíveis na Internet, [Wang et al. 2020] apresenta um estudo com a finalidade de criar um sistema de recomendação na área da saúde para fornecer material educacional apropriado para pacientes com doenças crônicas na China, e avaliar os efeitos do sistema. O método utilizado foi por meio de uma ontologia e várias técnicas de processamento de linguagem natural.

Os autores utilizam diversas fontes para reunir os materiais educacionais para os pacientes e um *dataset* de pacientes coletado de um sistema de telessaúde. O objetivo é projetar e implementar um sistema capaz de descobrir as necessidades potenciais dos pacientes e lhes recomendar material de estudo relevante.

A parte principal do processo de recomendação é uma ontologia chamada *Chronic Disease Patient Education Ontology* (CDPEO), que descreve características de pacientes para geração de recomendação. Os dados de pacientes e dos materiais educacionais são convertidos em vetores de mesmo tamanho que serão comparados e terão sua similaridade checada ao final. Os dados de pacientes são convertidos por meio de uma abordagem baseada em regras, já os dados de materiais são convertidos por meio de uma abordagem baseada em PLN.

Como resultados, a ontologia contou com 40 classes, 31 propriedades de objetos, 67 propriedades de dados e 32 indivíduos. São definidas 80 regras SWRL para a semântica da lógica do mapeamento de dados de pacientes para o espaço vetorial da ontologia. O sistema de recomendação é implementado como uma aplicação móvel utilizada nos telefones dos pacientes. A precisão macro foi de 0,970 para a recomendação *top ranking* e uma pontuação MAP de 0,628, mostrando que a recomendação na área da saúde tem um grande potencial na automatização da educação dos pacientes com doenças crônicas. As técnicas PLN aliadas com estratégias de IA foram muito eficientes para melhorar a performance do sistema.

4.4. Comparativo

Como é possível observar, já existem diversas soluções de diferentes formas para a questão da recomendação de itens utilizando técnicas de PLN. Como destaque, duas se sobressaem, tanto pela eficiência dos modelos propostos, quanto pela criatividade e boas escolhas nas abordagens utilizadas. O modelo de [Wang et al. 2020] tratou de um tema bastante relevante e de grande utilidade, que é a automatização do processo de aprendizado de pacientes com doenças crônicas, fazendo uso de uma ontologia para gerar recomendações relevantes de material didático, por meio de um aplicativo. [Naren et al. 2020] trouxe uma solução para um problema parecido com o levantado por este trabalho, e apresentou re-

sultados bastante satisfatórios, principalmente pelo uso do coeficiente de Dice, junto com o cosseno, garantindo recomendações precisas e relevantes.

Das técnicas PLN utilizadas, serão aproveitadas para a solução proposta a tokenização, a lematização (pela sua vantagem em relação à stemização), TF-IDF e Word2Vec. Para o cálculo da similaridade, a abordagem do cosseno foi a mais utilizada e, portanto, é a candidata mais forte para compor a etapa de comparação.

5. Proposta de sistema de recomendação híbrido

Ao analisar o estado-da-arte levantado nas pesquisas e os respectivos resultados de cada solução, a proposta para este trabalho é definida como um sistema de recomendação híbrido, em que há a ação de ambas filtragem colaborativa e por conteúdo. No funcionamento do sistema proposto, primeiramente há a filtragem colaborativa, contando com os dados de perfis de usuário, retornando recomendações de itens similares aos itens relacionados ao perfil de entrada, e em um segundo momento a filtragem baseada em conteúdo, fazendo um refinamento das recomendações e priorizando os itens de maior relevância para a entrada fornecida.

O sistema terá como entrada dados de interações de usuários com os itens que podem ser recomendados, disponíveis pela base de dados. Esses itens são todos os cursos disponíveis pelo *dataset* de uma plataforma de cursos *online*, e o sistema será baseado em uma aplicação real, utilizando dados reais, porém trazidos para um contexto de testes e experimentos, com usuários de teste, situações, variáveis e cenários definidos pelo autor do trabalho. Em seguida, será descrito o funcionamento do sistema e como são definidas as duas grandes etapas, a filtragem colaborativa e a baseada em conteúdo.

O início do sistema se dá pela etapa 1, **preparação dos dados**, na etapa da filtragem colaborativa, com a extração dos dados, que é feita em dois âmbitos: os disponíveis pelas bases de dados e os fornecidos pelos usuários dos experimentos. Com os dados extraídos, inicia-se o **pré-processamento de dados**. Nesta etapa, há a análise dos dados obtidos e o tratamento, para que se adequem às técnicas, algoritmos e ferramentas utilizadas no processo de geração das recomendações. Os dados mais relevantes aqui são os que dizem respeito às interações dos usuários com os itens disponíveis no *dataset*, ou seja, avaliações textuais, avaliações numéricas, tempo de permanência em um curso, se o usuário terminou o curso, se o usuário indica o curso, entre outras informações relacionadas. Os dados, ao final do processo, passam por uma outra transformação para que possam ser recebidos pelo algoritmo em seguida. Esse processo é chamado de vetorização.

Na **vetorização**, os dados já tratados e pré-processados são transformados em estruturas numéricas chamadas vetores, e são eles os elementos que participam do processo de comparação para que se encontre similaridades entre eles, e também compõem a estrutura da etapa 2, a **matriz de interações**. A matriz criada é uma estrutura bidimensional em que se encontram os vetores gerados pelos dados extraídos e tratados, e é percorrida pelo sistema ao realizar a comparação de vetores. Esse processo é denominado cálculo de similaridade, e é muito comum em sistemas de recomendação, pois o que se quer é encontrar sempre itens de maior similaridade entre si, que é o núcleo de uma recomendação.

Com todos os dados das bases disponíveis como vetores, pode-se também **extrair o perfil do usuário** que faz uso do sistema. Todos os processos descritos anteriormente

valem para os dados de usuário. Então, eles são extraídos, pré-processados, transformados em vetores, e após essa preparação, os dados de perfis estão prontos para participar da etapa 3, a última antes da geração de recomendações, já citada anteriormente, o **cálculo da similaridade**.

A partir dos vetores disponíveis pela base de dados, o vetor que modela o perfil de usuário é selecionado e utilizado para a comparação. Nesse processo, o código compara todas as estruturas vetoriais disponíveis com o vetor do usuário e seleciona as que tiverem a maior **similaridade**, uma métrica entre 0 e 1 para determinar quão próximos são dois vetores. Quanto mais próximo de 1, maior é a similaridade. Com todos os itens comparados, os de maior similaridade são selecionados e retornados no sistema, compondo a etapa 4, com o conjunto inicial de **recomendações geradas**.

Após o primeiro levantamento de itens relevantes para o usuário, começa a segunda grande parte da recomendação, o refinamento com base nos dados textuais de conteúdo dos cursos, por meio da filtragem baseada em conteúdo. Do *dataset* de cursos, a extração dá ênfase ao título, descrição e quaisquer outros dados de texto que descrevem o conteúdo dos cursos. Esses dados são priorizados e extraídos para participarem da etapa seguinte.

Com os dados extraídos, inicia-se a preparação dos dados. Nesta etapa, há a análise dos dados obtidos e o tratamento, para que se adequem às técnicas, algoritmos e ferramentas utilizadas no processo de refinamento das recomendações. É também nesta etapa que entra a **PLN**, onde se aplicam diversas formas de transformação de dados textuais, remoção de palavras irrelevantes, redução de palavras para o radical de maior peso, remoção de pontuações, entre outras. Após esse processo, as palavras que restam são transformadas em ***tokens***, que por sua vez são a entrada para o processo seguinte, a vetorização.

O processo de vetorização segue igual ao descrito anteriormente, transformando os *tokens* extraídos em estruturas vetoriais que são a entrada do processo de comparação de vetores e cálculo de similaridade.

A última etapa, de número 5, é definida por um novo **cálculo de similaridade**. Para este cálculo, o primeiro conjunto retornado anteriormente é selecionado para a comparação com os dados textuais de cursos que foram transformados em vetores. Ao final do processo, os itens de maior similaridade são ranqueados (*top-n* itens) e retornados para o usuário, de acordo com o número de cursos escolhido para serem recomendados.

Os capítulos [COLABORATIVO]5 e [CONTEUDO]6 apresentam e descrevem com detalhes cada filtragem, o primeiro com a filtragem colaborativa, e o segundo com a filtragem baseada em conteúdo. Cada uma das seções dará detalhes maiores sobre cada filtragem, explicando cada uma das etapas maiores e menores em cada um dos tipos de filtragem. A primeira, colaborativa, é desenvolvida para gerar as recomendações que são refinadas em seguida, e levantar o primeiro grupo de cursos recomendados, de onde saem os cursos que são recomendados ao final, para o usuário. A segunda, baseada em conteúdo, é desenvolvida para servir ao propósito do refinamento, definido na seção atual.

Nas últimas subseções de cada uma das próximas seções há a parte de testes e experimentos. Os testes deste trabalho dizem respeito a um contexto teórico e explicativo, em que os dados utilizados são reais, mas inseridos em um sistema simulado, com

usuários e cenários definidos pelo autor. Cada indivíduo que participa dos experimentos tem seu perfil extraído e definido manualmente, algo que ocorre naturalmente em aplicações e plataformas *online* que possuem sistemas de recomendação, como Udemy, Udacity, Netflix e Amazon. Nesses serviços, o perfil de usuário está sempre disponível e os dados são mantidos para gerar e aprimorar as recomendações. Aqui, a forma como esses dados são obtidos é explicada em cada seção, e são utilizados por meio da entrada do usuário no sistema simulado.

O sistema será desenvolvido em código aberto em um Jupyter Notebook[1], disponível para executar em qualquer editor de texto com suporte e qualquer máquina com Python 3 e as devidas dependências instalados. Cada etapa terá pedaços de texto ou comentários explicando o que está sendo feito e pedaços de código construindo e executando o sistema proposto. O ChatGPT entra como um auxílio nas partes mais detalhistas do código e mais específicas, para otimizar o desenvolvimento. Como explicado anteriormente, os perfis de usuário serão a entrada de cada sistema, sendo introduzidos manualmente pelo usuário ativo nas funções que retornam as recomendações de cursos.

Como principais ferramentas no desenvolvimento do modelo colaborativo e do modelo baseado em conteúdo serão utilizadas as seguintes bibliotecas e recursos:

- NumPy[2]
- pandas[3]
- SciPy[4]
- scikit-learn[5]
- re (Python)[6]
- NLTK[7]
- GenSim[8]

6. Sistema de recomendação: filtragem colaborativa

Para o caso da filtragem colaborativa, um dataset de interações de usuário se fez necessário para compor a matriz que faria o papel do elemento principal para a geração de recomendações. Com base nisso, dados de avaliações de alunos dos cursos da Coursera foram levantados. O conjunto conta com dois *datasets*, um de avaliações com mais de 200 mil registros de avaliações de usuários, e um de cursos, com mais de 600 cursos, ambos coletados no ano de 2020.

O funcionamento geral do SR baseado em filtragem colaborativa desenvolvido por este trabalho pode ser resumido em 5 etapas: **reconhecimento dos dados**, onde um primeiro estudo é feito sobre as bases de dados obtidas, **preparação dos dados**, em que é feita uma adaptação para se enquadrar nos métodos que serão utilizados e nas técnicas

[1]https://github.com/marcelomuller/sr-projeto
[2]https://numpy.org/
[3]https://pandas.pydata.org/
[4]https://scipy.org/
[5]https://scikit-learn.org/
[6]https://docs.python.org/3/library/re.html
[7]https://docs.python.org/3/library/re.html
[8]https://pypi.org/project/gensim/

para gerar as recomendações, **criação da matriz esparsa**, em que uma matriz de usuário-item é gerada para ser a peça central na geração de novos itens, **geração de curso recomendado**, uma função que utiliza o KNN para gerar novos cursos com base em um curso de entrada, e **fatoração de matriz**, uma etapa extra, não necessária, que ilustra um processo de compactação da matriz utilizada anteriormente que, em alguns casos, pode gerar melhores recomendações.

O início se dá pelo carregamento dos dados de ambos os *datasets*, avaliações e cursos. Os processos seguem em ordem. **Pré-processamento**, que compreende um tratamento inicial nas colunas disponíveis em cada base de dados, como remoção de colunas irrelevantes e adequação de identificadores. Em seguida, **matriz esparsa**, em que é criada uma matriz usuário-item, utilizando IDs de usuários e cursos, para servir como base do funcionamento do algoritmo de recomendação colaborativa. Após isso, a aplicação da técnica do **KNN (k-Nearest Neighbors)** para gerar k cursos mais relevantes, de acordo com uma métrica estabelecida, um dado valor de k e um curso inicial para obter os próximos mais similares. Por fim, a **geração das recomendações**, fornecendo como entrada e como perfil de usuário um curso X para encontrar os K cursos mais próximos a ele. A última etapa, sendo opcional, **fatoração da matriz**, reduz a dimensionalidade da matriz inicial criando duas novas matrizes, podendo ou não obter resultados melhores do que a primeira matriz.

6.1. Obtenção e carregamento dos dados

O caso da filtragem colaborativa iniciou com um grande problema, encontrar uma base de dados rica em interações de usuários para os cursos trabalhados anteriormente na filtragem baseada em conteúdo.

A primeira proposta para tratar esse desafio foi criar um *dataset* do zero, com o mínimo possível de colunas (id de usuário, id do curso e avaliação, por exemplo) e utilizando apenas alguns assuntos de todos os disponíveis, e desses assuntos, uns cinco ou seis cursos apenas. A ideia era realizar uma espécie de experimento direcionado, contando com a participação de colegas e amigos de diversas áreas que pudessem contribuir com notas, porém, com base apenas no título, descrição e nota média de cada curso, já que eles não eram alunos de verdade.

Em meio à implementação dessa ideia, um conjunto de *datasets*[9] disponíveis no Kaggle foi encontrado. Um deles com cursos da Coursera, com título, o nome da instituição que o disponibilizou na plataforma, a URL e o id. E o outro com avaliações escritas e numéricas desses cursos. Levando em consideração que eram bases de dados muito mais completas do que a da proposta anterior, e isso geraria uma matriz menos esparsa e recomendações melhores, esta solução foi a escolhida para implementar o SR colaborativo.

O primeiro *dataset*, o de cursos, contém 604 cursos únicos e quase 1 milhão e meio de avaliações. Dessas avaliações, há mais de 287 mil usuários únicos, sendo que todos os cursos contam com pelo menos uma avaliação.

Com inicialmente cinco colunas, o *dataset* mostra as avaliações textuais na coluna *reviews*, os nomes de cada aluno que escreveu a avaliação na coluna *reviewers* (sendo um

[9]https://www.kaggle.com/datasets/imuhammad/course-reviews-on-coursera

padrão o primeiro nome seguido pelas iniciais dos sobrenomes, precedido por um 'By'), a data da postagem da avaliação em *date_reviews*, a nota numérica em *rating* (a coluna mais importante nas recomendações), e *course_id* com os IDs de cada curso.

A matriz utilizada no processo de recomendação é criada utilizando um *dataframe* chamado *ratings*, o qual armazena as avaliações dos usuários, com id de usuário, id do curso e avaliação numérica.

6.2. Pré-processamento dos dados

O pré-processamento de dados nesse modelo é diferente da versão baseada em conteúdo, e precisa de menos ajustes. Não há colunas a serem usadas com arquivos de texto nesse caso, então as técnicas PLN utilizadas no pré-processamento do modelo baseado em conteúdo, como remoção de *stopwords*, stemização e aplicação de filtros por expressões regulares não são relevantes aqui. Isso será melhor comentado na seção de trabalhos futuros.

Duas alterações nos dados foram necessárias. Adicionar uma coluna de id de usuário (userId) no *dataframe* de avaliações, *ratings* (citado anteriormente), que inicialmente contava apenas com o nome do aluno que avaliou o curso, e como o nome dos alunos não é relevante para o modelo, a coluna inteira foi removida também. E, para evitar que o mesmo usuário avalie duas vezes o mesmo curso, os dados duplicados com base em um par de *userId* X e *courseId* Y foram removidos, mas sem levar em conta a data das avaliações. Isso evita com que a matriz acabe unindo as avaliações duplas e seja preenchida por números maiores do que o intervalo possível de valores para o *rating*, que é de 0 a 5.

6.3. Criação da matriz esparsa usuário-item

Esta etapa é responsável por criar uma matriz esparsa que age como o principal componente na geração de recomendações com base nos dados apresentados. A entrada da função que cria a matriz é o *dataframe ratings*, e a partir dos campos de id *courseId* e *userId*, os mapeadores e mapeadores inversos são criados em conjunto, e a partir deles pode-se extrair as duas dimensões da matriz, usuário e item (curso). O processo gera uma matriz de *shape* 287808x604, sendo usuários e cursos, respectivamente. Após a criação, a matriz será uma estrutura grande e formada majoritariamente por zeros (por isso o nome, esparsa), com os registros não-zero preenchidos pelas avaliações do par *userId-courseId*.

6.4. Obtenção do perfil

Diferente da filtragem baseada em conteúdo que será apresentada no próximo capítulo, a forma como os perfis são obtidos aqui é diferente. Nos *datasets*, não há possibilidade de classificar cursos quanto ao nível (iniciante, avançado etc.) como na outra filtragem, visto que aqui conta-se apenas com a nota atribuída por cada usuário, já que, como explicado anteriormente, as avaliações em texto não são consideradas. Dito isso, os usuários que fazem parte do experimento são usuários encontrados na base de dados. Fazendo dessa forma pode-se mitigar também o problema do *cold start*, ao menos para o objetivo do experimento feito neste trabalho, porque não há possibilidade de um usuário não ter ainda avaliado pelo menos um curso.

O perfil dos usuários é representado por uma *string*, com palavras separadas por hífens, que é ou o título do curso respectivo, ou parte do título. Essas *strings* são os

próprios IDs dos cursos no *dataset* utilizado e são elas que servem de entrada para o modelo de recomendação, pois ele traz cursos similares a um curso de entrada.

Esse perfil é totalmente representado apenas pelo primeiro curso dentro do *dataset* que o indivíduo tenha feito. Essa é uma limitação da abordagem escolhida, também devido ao fato de não se ter um conjunto de testes para realizar os experimentos, como é comum em outras técnicas, abordagens e áreas do ML. No modelo proposto, não se tem informações de nível de usuário, área, gostos ou preferências. A opção escolhida que mais se encaixou com a forma como as bases de dados são e o funcionamento do motor de recomendação com o KNN foi representar o perfil sendo a *string* ID do curso inicial, com o qual ele deseja encontrar outros cursos parecidos.

7. Aplicação do KNN e recomendações

O método escolhido para gerar as recomendações foi o KNN (k-Nearest Neighbors), que traz um número k de vizinhos para um dado item, baseado em alguma métrica. Na seção de experimentos algumas métricas são utilizadas e comparadas, como a distância de cosseno e distância *Manhattan*. A métrica padrão é o cosseno.

A função que realiza o cálculo do KNN recebe o id do curso do perfil de interesse, os mapeadores criados junto com a matriz esparsa, a própria matriz, o valor de k e a métrica escolhida. O mapeador aponta na matriz o vetor que deve ser utilizado, com base no ID do curso utilizado na entrada. Com o vetor levantado, os k vizinhos são encontrados (k + 1 para excluir o próprio curso da entrada) e os IDs dos cursos são retornados.

Na matriz, os cursos têm nomes mapeados para valores numéricos. Supondo que o curso principal da linguagem Python tenha ID 233, ao encontrar registros na matriz de avaliações de usuários para aquele curso, os perfis são considerados semelhantes.

Os mapeadores transformam os valores de *userId* e *courseId* em índices de 0 a n, e os inversos fazem o caminho contrário.

No KNN, esse ID por escrito serve de entrada para retornar IDs de cursos vizinhos, para o determinado K, utilizando determinada métrica para o cálculo da distância (padrão é cosseno).

Após esse processo, os nomes dos cursos são listados e levantados de acordo com os IDs obtidos a partir do KNN, e dessa forma as recomendações são retornadas com os nomes dos cursos:

Curso de interesse: 'python'

K: 5

Cursos retornados:

- Python Data Structures
- Using Python to Access Web Data
- Using Databases with Python
- Machine Learning

7.1. Fatoração de matriz

Após o uso completo da matriz criada anteriormente, há também a possibilidade de realizar um processo a mais para que, em alguns casos, obtenha-se melhores

recomendações. Esta última etapa é opcional e será comentada na seção de experimentos, para fins de comparações entre a matriz original e a matriz fatorada.

Quando se busca aprimorar recomendações visando *features* latentes, que não se consegue enxergar apenas observando a base de dados como é, a fatoração de matriz pode ser uma opção de técnica interessante para representar de forma mais compacta os gostos dos usuários e descrições de itens. Isso se torna ainda mais interessante quando os dados trabalhados são muito esparsos. O algoritmo fatora a matriz original usuário-item em duas matrizes: usuário-fator (n de usuários, k) e item-fator (k, n de itens).

A redução de dimensão da matriz busca evidenciar os gostos. O que cada *feature* k representa não é exato, então o valor real dessas features é interpretativo de acordo com o *dataset* e com os perfis dos usuários.

A técnica utilizada foi o TruncatedSVD[10] do sklearn, e ela permite que o usuário altere como quiser os valores de componentes presentes na matriz e também o número de iterações. Conforme esses valores mudam, melhores ou piores podem se tornar as recomendações retornadas. É preciso encontrar uma combinação "especial" para cada base de dados usada, e até para cada caso trabalhado. Exemplo de uso:

N componentes: 30

N iterações: 40

Shape: 604x30

Curso: 'python'

- Foundations of Data Science: K-Means Clustering in Python
- Corporate & Commercial Law I: Contracts & Employment Law
- Introduction to Economic Theories
- Programming Fundamentals

Como a saída evidencia, não necessariamente se tem resultados melhores do que os apresentados pela matriz X anterior. Uma de várias explicações é o tamanho da base de dados, já que a fatoração se beneficia de bases de dados muito grandes e esparsas. E também, como já comentado, é preciso encontrar o *sweet spot* na configuração de todos os valores.

7.2. Experimentos

Esta seção trata da condução completa dos experimentos de testes realizados com o modelo construído e a base de dados utilizada.

Na primeira parte, uma breve descrição contextualizando os indivíduos, cenários e diferentes estilos para cada exemplo de experimento. Em seguida, a origem dos dados é detalhada, evidenciando a captação do perfil de cada usuário que participa do experimento e a forma didática como esses perfis fazem para gerar as recomendações.

Na próxima subseção, as métricas utilizadas são citadas, explicando motivos para o uso e decisões, conforme se deu a condução do experimento e a criação do modelo.

[10]https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html

Por fim, a condução dos experimentos para cada caso, e ao final de cada um, uma discussão sobre os resultados obtidos e esperados, e algumas interpretações sobre as recomendações para cada cenário.

7.2.1. Contexto e cenários

Para este experimento, três cenários com três indivíduos de diferentes objetivos e contextos de vida foram reunidos para compor o elenco de sujeitos de teste. São situações realmente diferentes para reforçar que esses cursos servem para todo tipo de objetivo e assunto desejado. Cada um busca uma categoria específica de curso online para suprir seus interesses e agregar à sua vida acadêmica e profissional, para um hobby, ou até mesmo por pura curiosidade. Implementando o sistema proposto, foram definidos:

1. Um aspirante a cientista de dados que quer ingressar na carreira e já realizou o curso inicial de Python;
2. Um pai de primeira viagem em busca do máximo possível de materiais que o ajudem a levar a vida nova, e que já realizou o curso Everyday Parenting;
3. Um entusiasta da música eletrônica que deseja se aprofundar na parte de produção musical, e já cursou A arte da produção musical.

7.2.2. Origem dos dados

A avaliação consiste na captação do perfil do usuário, utilizando o id do curso inicial realizado, que servirá de base para o KNN retornar os k cursos mais próximos. O modelo utiliza como base de dados um *dataframe* com as avaliações dos outros usuários da plataforma. Uma matriz esparsa é criada com base nesse *dataframe*, e utilizando os mapeadores, o KNN percorre a matriz e traz os k cursos de maior relevância para a pesquisa, de acordo com outras avaliações de usuários que realizaram também aquele curso, ou seja, usuários de perfil similar.

O valor de k utilizado no experimento para o KNN será k - 1, ou seja, para um k = 11, são retornados 10 cursos. Isso é feito para excluir o próprio curso inicial do retorno.

7.2.3. Métricas dos experimentos

Como métricas de avaliação, foram utilizadas: precisão, para medir a proporção de itens relevantes entre os recomendados e NDCG (*Normalized Discounted Cumulative Gain*), para o caso de a ordem das recomendações estar errada e itens relevantes aparecerem antes de itens relevantes. Nesse último caso, foi atribuído por análise própria do autor em uma escala 0 a 5, seguindo a lógica do funcionamento do sistema e a utilização de notas dadas pelos usuários nos cursos, para compor os cálculos do NDCG e obter valores mais condizentes. A ordem ideal considerada para o cálculo do NDCG foi a própria ordem sugerida na seção de discussão de cada caso do experimento, seguindo da maior relevância obtida até a menor. Quanto mais próximo de 1 for o resultado do NDCG, melhores foram as recomendações daquele caso do experimento.

7.2.4. Condução dos experimentos

CASO 1:

O jovem universitário que almeja seguir carreira na área de ciência de dados deseja se aprofundar na parte técnica. Ele realizou o curso básico de Python e está buscando mais materiais:

- Nome: Vinicius
- Curso realizado: Programação para todos (Primeiros passos no Python)
- ID: 'python'

O perfil do usuário será representado por **'python'**, o ID do curso realizado, que é também uma das entradas do KNN.

Para o valor de **k = 6**, os resultados representados por "Título (relevância)":

Porque você avaliou Programming for Everybody (Getting Started with Python):

1. Python Data Structures (5)
2. Using Python to Access Web Data (0)
3. Using Databases with Python (4)
4. Machine Learning (3)
5. Neural Networks and Deep Learning (1)

Precisão: 4/5 = 80% **NDCG**: 0,9438

Ordem ideal determinada: 1, 3, 4, 5 e 2

Discussão dos resultados:

O primeiro curso recomendado é de nível básico e não necessita de conhecimento prévio. Ele trata da base do Python dando ênfase em estruturas de dados e operações iniciais com a linguagem. Por isso é um curso muito interessante para iniciantes.

O segundo curso é bem direcionado a *web*, passando por *scraping* e trabalhando com JSON e XML, então é algo bem específico e fora do escopo do que o aluno procura.

O terceiro é um curso focado em bancos de dados utilizando Python, trazendo conceitos de CRUD, SQL, orientação a objetos, entre outros assuntos para uma base sólida, bem importante para ciência de dados.

O quarto é muito interessante, já que ML e *ata science* andam lado a lado na área de dados. O curso traz conceitos utilizados amplamente pelos profissionais de ciência de dados.

O último apenas foi considerado não relevante por se tratar de uma área mais específica de IA e ML, algo que foge um pouco do escopo desejado. Seria um curso mais interessante para um aluno de inteligência artificial, por exemplo.

CASO 2:

O novo pai que procura mais formas de reunir conhecimento sobre a vida diferente e cheia de desafios que o espera:

- Nome: Bruno
- Curso realizado: Everyday Parenting (Paternidade cotidiana): The ABCs of Child Rearing (O ABC da criação de filhos)
- ID: everyday-parenting'

O perfil do usuário será representado por **'everyday-parenting'**, o ID do curso realizado, que é também uma das entradas do KNN.

Para o valor de **k = 6**, os resultados representados por "Título (relevância)":

Porque você avaliou Everyday Parenting (Paternidade cotidiana): The ABCs of Child Rearing (O ABC da criação de filhos):

1. The Science of Well-Being (4)
2. Child Nutrition and Cooking (5)
3. Learning How to Learn: Powerful mental tools to help you master tough subjects (0)
4. Positive Psychology: Resilience Skills (3)
5. Psychological First Aid (3)

Precisão: 4/5 = 80% **NDCG**: 0,8574

Ordem ideal determinada: 2, 1, 4, 5 e 3

Discussão dos resultados:

O primeiro resultado é um curso que traz diversos desafios para a finalidade de criar hábitos mais saudáveis e produtivos, contribuindo para que a pessoa seja mais feliz. Cursos sobre psicologia de hábitos, rotina e saúde são ótimas sugestões para novos pais.

O segundo curso foca totalmente em nutrição infantil, levantando os mais diversos pontos das causas da má saúde nas crianças, como obesidade infantil, e ensina diversas práticas e habilidades para garantir uma dieta saudável para as crianças. O corpo é o bem mais precioso, e hábitos criados na infância podem durar pela vida inteira. Excelente recomendação.

O terceiro é um curso que ensina técnicas e métodos diferentes para aprender temas difíceis, combater procrastinação, técnicas de memorização, etc. É uma recomendação interessante para todos que estejam estudando para um concurso público, ou ao menos precisam de ajuda para estudar para uma prova muito difícil, ou têm pouco tempo para aprender um tema mais complexo. Para o caso de um pai que deseja um conteúdo mais leve, que seja apenas uma suplementação, não é tão relevante.

O quarto é relevante por ensinar aos alunos a desenvolver resiliência, que é uma virtude importante para todos, especialmente pais que precisam ser fortes em todos os âmbitos para elevar sempre a qualidade da criação dos filhos.

O quinto curso, apesar de ter como objetivo tratar de amparo psicológico emergencial em situações graves, é também muito útil no ambiente familiar. Pode ser de grande valia para lidar com problemas que a criança pode vir a apresentar futuramente, quando tiver mais idade.

CASO 3:

O fã e entusiasta de música eletrônica que foi inspirado pelos seus artistas favoritos a começar a produzir sua própria música:

- Nome: Paulo
- Curso realizado: A arte da produção musical
- ID: ‘producing-music’

O perfil do usuário será representado por **‘producing-music’**, o ID do curso realizado, que é também uma das entradas do KNN.

Para o valor de **k = 6**, os resultados representados por ”Título (relevância)”:

Porque você avaliou The Art of Music Production:

1. The Technology of Music Production (5)
2. Developing Your Musicianship (5)
3. Music Business Foundations (3)
4. Songwriting: Writing the Lyrics (4)
5. Machine Learning (0)

Precisão: 4/5 = 80% **NDCG**: 0,9909

Ordem ideal determinada: 1, 2, 4, 3 e 5

Discussão dos resultados:

A melhor recomendação está no primeiro resultado. Apesar de ter um nível mais básico do que o curso inicial, ele é composto de aulas sobre o processo completo da produção musical, inclusive aulas teóricas sobre o som em si.

O segundo curso é direcionado totalmente à música, passando por escalas, tonalidades, harmonia e assinaturas diferentes. A habilidade de tocar instrumentos e entender sobre toda a composição de uma música é um tema muito procurado e que contribui muito na carreira de produtores musicais.

O terceiro é focado no mercado da música, ensinando conceitos como direitos autorais, o papel dos agentes, contratos e outras entidades que fazem parte da área comercial da música. Apesar de relevante, é um tema geralmente procurado por artistas e produtores mais avançados, não os que estão ainda começando.

O curso de composição de letras de música é de grande valia a qualquer artista, um aliado a qualquer um que queira aprender a colocar suas ideias, emoções e mensagens que quer passar em suas músicas, principalmente novos e inexperientes artistas.

O último é o curso de ML que já apareceu nos resultados do primeiro caso, e é um dos mais avaliados da Coursera. Pode ser esse o motivo de ele aparecer aqui, assim como, talvez, a falta de mais cursos focados em produção musical na mesma intensidade dos anteriores.

7.3. Outras métricas de distância

Ao utilizar outras métricas de distância além do cosseno, os resultados não foram satisfatórios. Mesmo utilizando todas as distâncias disponíveis pelo sklearn, como euclidiana, Manhattan, Minkowski, entre outras, e aplicando normalização nos dados, para todos os três cenários e até outros testes avulsos, os resultados foram parecidos, trazendo recomendações com pouca ou nenhuma relação ao curso de entrada. Há algumas explicações para o motivo disso acontecer, como a distância de cosseno lidar melhor

com alta esparsidade, que o caso, também por ser a mais utilizada em modelos de filtragem colaborativa, já que cosseno foca na orientação dos vetores e não na magnitude. Um usuário pode dar apenas notas altas, e outro apenas notas baixas, mas mesmo assim ambos podem ter perfis semelhantes. Por isso pode-se usufruir melhor do que a distância de cosseno proporciona para esse tipo de filtragem.

8. Sistema de recomendação: filtragem baseada em conteúdo

Esta seção aborda o segundo sistema de recomendação na ordem do funcionamento do sistema proposto, com filtragem baseada em conteúdo, e todas as suas etapas. Cada subseção detalha e explica uma das etapas em todo o processo de geração das recomendações.

8.1. Extração de dados

Os maiores desafios da implementação da solução proposta já estão evidenciados desde o início. O *dataset* é, provavelmente, o recurso mais difícil de se obter, e também é o mais importante do projeto. É necessária uma base de dados que compreenda diversos tipos de alunos de cursos, com os mais variados gostos, padrões de aprendizado, objetivos, entre outros aspectos que serão relevantes para as recomendações. Muito diferente do dataset com dados dos cursos *online*, que são de fácil acesso e existem muitos disponíveis nas plataformas mais famosas de bases de dados. Uma outra barreira notável é o próprio desenvolvimento do sistema de recomendação, uma vertente muito famosa do ML e *data mining*, porém, a princípio, com poucos exemplos práticos e didáticos disponíveis em inglês ou português.

Durante a fase de desenvolvimento, foi determinado que o início se daria utilizando a abordagem da filtragem baseada em conteúdo para, primeiramente, obter um algoritmo de recomendação simples para servir como protótipo, utilizando como base de dados um *dataset* obtido no Kaggle, contendo muitos cursos da plataforma Udemy de *e-learning*.

Esse *dataset* [11], que continha mais de 200 mil registros sobre cursos dos mais variados temas e em todas as linguagens disponíveis na plataforma, recebeu um tratamento prévio e sofreu uma redução. As colunas consideradas relevantes para a recomendação foram *title* e *headline*, que são o título do curso e sua descrição, respectivamente.

O *dataframe* utilizado em todo o processo de recomendação é criado unindo as duas colunas e criando uma nova coluna *text* contendo todo o texto relevante para as *queries*.

Há também uma filtragem no *dataframe* pelos idiomas disponíveis, por meio da coluna *language* e mantendo apenas cursos em inglês, português e espanhol. Esse foi um dos primeiros problemas encontrados pelas recomendações de teste iniciais, pois os cursos levantados eram predominantemente cursos de idiomas asiáticos, como japonês e chinês, provavelmente por uma questão de ordenação de caracteres na criação do dicionário, já que esses idiomas utilizam alfabetos diferentes do romano.

[11] https://www.kaggle.com/datasets/hossaingh/udemy-courses?select=Course_info.csv

8.2. Pré-processamento dos dados

Utilizando os recursos das bibliotecas NLTK[12] e Scikit Learn[13] do Python, a coluna *text* passa por um pré-processamento com remoção de *stopwords*, stemização com Porter Stemmer e outros tratamentos, que consistem em filtragens por expressões regulares para remover caracteres especiais, *tags* HTML e outros tipos de dígitos, assim como técnicas PLN para que se possa trabalhar com o texto em sua forma pura, e por fim, o texto todo é *tokenizado*.

8.3. Vetorização

Inicialmente, o desenvolvimento do algoritmo foi feito dentro da plataforma Google Colab[14], utilizando Jupyter Notebook. Para a vetorização e criação do vocabulário que terá a similaridade medida, foi definido a ser utilizado o TF-IDF[15], junto com CountVectorizer[16], na coluna *courses['text']*.

A implementação consistia no carregamento por completo do *dataset* em memória, previamente à aplicação dos algoritmos. Por consequência de alguma mudança de política na plataforma, esse carregamento de dados foi alterado para apenas parte do *dataset*, e isso era um fator crucial para o funcionamento do sistema de recomendação em um computador convencional.

Para contornar as limitações de carregamento da plataforma, o código passou a ser desenvolvido e executado localmente, porém isso gerou uma nova situação, a execução de todas as funções que faziam parte da geração de recomendações se tornou pesada demais para a memória do computador. O *dataset* foi carregado por completo novamente, e isso impactou diretamente na etapa do TF-IDF.

O interpretador sempre lançava um erro exigindo quase 50GB de memória RAM para conseguir processar um vocabulário a partir de uma quantidade tão grande de dados, e isso fez com que a abordagem fosse deixada de lado.

Para contornar a situação, o Word2Vec[17], disponível pela biblioteca Gensim, mostrou-se de grande valia para realizar a vetorização de todos os tokens gerados nas etapas anteriores, fornecendo tudo que era necessário para os testes iniciais das primeiras recomendações, utilizando a filtragem baseada em conteúdo, como já apontado, os títulos e descrição dos cursos, comparando com termos de entrada do usuário.

Com os *tokens* gerados na etapa anterior, é criado o modelo de vetores com o Word2Vec para cada um dos cursos disponíveis no *dataframe*, utilizando novamente a coluna *courses['text']*.

8.4. Obtenção do perfil

Na geração de recomendações feitas para o caso da recomendação baseada em conteúdo, é usado um sistema que recebe uma *query* com alguns termos que servem como base

[12]https://www.nltk.org/
[13]https://scikit-learn.org/
[14]https://colab.research.google.com/
[15]https://pt.wikipedia.org/wiki/Tf-idf
[16]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html
[17]https://radimrehurek.com/gensim/models/word2vec.html

para recomendar novos cursos. A *query* é composta pela **área de interesse** e **nível de conhecimento** desejados.

A primeira parte da *query* diz respeito ao tipo de curso procurado pelo aluno. Caso a pessoa esteja procurando um curso sobre a linguagem de programação Python, essa parte terá palavras-chave que se refiram ao tema. Já na segunda parte, o aluno deve escrever termos que se refiram ao nível de conhecimento desejado. Nesse caso, o aluno quer um curso *starter*, ou seja, iniciante no assunto.

Esses são os dados de perfil coletados do usuário. Como já explicado, pela natureza do experimento não ser implementado em um sistema real, e pelos indivíduos que participam dos testes serem definidos pelo autor, o perfil de usuário será totalmente representado pela **consulta** composta pelas duas palavras-chave escolhidas, introduzidas à mão no *notebook* onde o código está disponível.

A consulta então é dividida também em *tokens*:

['python', 'starter']

Após isso, os *tokens* são utilizados para criar os vetores das palavras-chave, e parte dessas estruturas pode ser representada da seguinte forma:

[array([-1.316816 , -0.00765565, -1.5253502 , -0.4065882 , 1.1508659 , 0.69462216,
2.6009755 , 1.1592892 , -0.18154877, 0.10172073, ...)]

8.5. Similaridade

Em sequência da **vetorização**, baseado no modelo gerado pelo Word2Vec em cima da coluna *text*, um array de similaridades vazio é criado, e em um laço de repetição é calculada a similaridade para cada curso presente em *courses['text']*, que compreende o *ataframe* inteiro de cursos, e cada similaridade é introduzida ao *array* de similaridades. Uma nova coluna *courses['similarity']* é criada no *dataframe* que recebe o *array*.

8.6. Recomendações

Após a criação da coluna *similarity* dentro do *dataframe*, um segundo *dataframe* chamado **recommended_courses** é criado, ordenando o anterior pela coluna *similarity*.

A partir disso, são levantados os **cinco primeiros cursos** com a **maior similaridade** e retornados ao usuário, que nesse caso são as primeiras linhas do dataframe novo, recommended_courses.

8.7. Experimentos

O capítulo conduz, exemplifica e discute alguns cenários de testes para os experimentos realizados em cima das bases de dados utilizadas, e do sistema de recomendação proposto.

A primeira seção explica o contexto criado em torno dos experimentos, evidenciando diferentes cenários e situações do uso do sistema. Em seguida, de onde os dados são coletados para delinear os testes realizados. Após isso, as métricas utilizadas e os motivos são descritos. Por fim, a condução dos testes com o detalhamento completo e, ao final, uma discussão sobre os resultados obtidos.

8.7.1. Contexto e cenários

Como avaliação da filtragem baseada em conteúdo, três cenários baseados em uso real de uma plataforma por três indivíduos foram levantados. Os indivíduos do experimento contam com um contexto de vida profissional e pessoal e gostos pessoais diferentes, cada um em busca de um tipo específico de curso online para suprir seus interesses. Implementando o sistema proposto, foram definidos:

1. Um aluno do curso de Sistemas de Informação, em seu primeiro ano, em busca de um curso focado em uma linguagem de programação ou uma *stack* de desenvolvimento de *software* para iniciar sua carreira com um primeiro estágio na área;
2. Um profissional da área de design buscando um curso para ingressar na área de tecnologia como UX/UI;
3. Um entusiasta em busca de um curso para aprender um instrumento novo.

8.7.2. Origem dos dados

Para todos os usuários, a avaliação consiste na captação do perfil dele e na transformação em textos para servirem de termos de uma consulta para retornar cursos que satisfaçam aquela pesquisa com o motor de recomendação. Os textos são separados em *tokens* que participam do processo de vetorização. É criado um *array* de similaridades vazio que é populado com a similaridade calculada entre cada *token* do *dataframe* (coluna *courses['text']*) com cada token da consulta. Ao final, uma coluna *courses['similarity']* é criada, **ordenada por similaridade**, e os **cinco mais altos** no ranking são os **recomendados**.

Neste caso, vamos considerar que já foi realizada a extração e foram levantados os termos levando em conta:

- A área de interesse do curso procurado;
- O nível do curso desejado pelo usuário, se precisa ser para um iniciante ou um usuário avançado, por exemplo.

8.7.3. Métricas dos experimentos

Como métricas de avaliação, foram utilizadas: precisão, para medir a proporção de itens relevantes entre os recomendados e NDCG (*Normalized Discounted Cumulative Gain*), para o caso de a ordem das recomendações estar errada e itens relevantes aparecerem antes de itens relevantes. Nesse último caso, foi atribuído por análise própria do autor como 1 para o item relevante e zero para o item não relevante, para compor os cálculos do NDCG. A ordem ideal considerada para o cálculo do NDCG foi a própria ordem sugerida na seção de discussão de cada caso do experimento.

Foi escolhido manter as métricas do outro experimento para fins de comparação ao final do capítulo e conseguir esclarecer e evidenciar vantagens e desvantagens de um tipo de filtragem em relação ao outro.

8.7.4. Condução dos experimentos

CASO 1:

O aluno está em busca de um curso de programação para ingressar no mercado. Ele possui noções básicas de JavaScript, CSS e HTML e está querendo aprender um *framework* para desenvolvimento *web* e ouviu falar do VueJS.:

- Nome: Augusto
- Área de interesse: VueJS
- Nível de conhecimento: Iniciante

O perfil do usuário será representado por **'vue starter'**, sendo vue a área de interesse e starter (iniciante) o nível do público alvo do curso.

Tokens da query:

['vue', 'starter']

Utilizando como entrada **'vue starter'**, com a finalidade de encontrar um curso que ensine o básico do *framework* VueJS, teve como retorno o seguinte resultado:

Cursos recomendados:

Título: Complete Vuejs Course: Vue.js + Nuxt.js + PHP + Express.js

Descrição: VueJS, Command Line, Babel, NPM, Webpack, Vue JS CLI, Vue.js Router, VueX, Axios, iView, Express.js, Nuxt.js

Similaridade: 0.9106954336166382

Relevância: 1

Título: Webpack 4: Beyond the Basics

Descrição: Quick, code-driven, follow-along Javascript tutorials of Webpack, Babel, React, Angular, Vue, Redux, SSR, Typescript

Similaridade: 0.907133162021637

Relevância: 0

Título: React JS, Angular & Vue JS - Quickstart & Comparison

Descrição: Angular (Angular 2+), React or Vue? Get a Crash Course on each of them and a detailed comparison!

Similaridade: 0.9065245985984802

Relevância: 1

Título: Complete Vue.js 3 (Inc. Composition API, Vue Router, Vuex)

Descrição: (RE-RECORDED April 2021) Vue.js 3 is here! Learn from "Hello, Vue!" to building large apps with Vuex and Vue Router.

Similaridade: 0.9039604663848877

Relevância: 1

Título: React, NextJS and NestJS: A Rapid Guide - Advanced

Descrição: React with Typescript, Next.js, Redux, NestJS, Docker, Redis, Stripe, Frontend & Backend Filtering

Similaridade: 0.9017014503479004

Relevância: 0

Precisão: 3/5 = 60%

NDCG: 0,9060

Ordem ideal determinada: 1, 4, 3, 2 e 5

Discussão dos resultados:

O primeiro curso recomendado realmente é direcionado para iniciantes, com a apresentação do básico de uma *stack* inteira, não apenas o VueJS, para criar um projeto completo do zero, o que provavelmente é uma das tarefas desse curso.

O segundo curso já não é tão interessante por ser focado em Webpack, e o Vue é apenas uma parte dele, o que foge do objetivo inicial da procura.

O terceiro, apesar de não ser realmente o que o usuário está buscando, é um curso que apresenta comparativos entre alguns dos maiores *frameworks* de JS do mercado, o que pode dar ao usuário um quadro geral e uma visão mais completa sobre cada um, para que assim se decida melhor sobre qual deles escolher para utilizar em seus projetos.

O quarto é outra ótima recomendação, porque parece ser uma atualização do primeiro, dessa vez com a terceira versão do Vue, assim como algumas tecnologias de auxílio para o desenvolvimento utilizando essa ferramenta, como Vuex e Composition API.

O último é um curso de ReactJS, então não é exatamente o que o usuário está procurando, mas ainda está no mesmo grupo de tecnologias auxiliares para desenvolvimento web com JS.

CASO 2:

O profissional de design em busca de iniciar sua carreira na área de tecnologia como UX/UI. Ele possui conhecimento básico sobre Photoshop e Figma e quer aprender o básico necessário para uma vaga de entrada como UX:

- Nome: Pedro
- Área de interesse: Design
- Nível de conhecimento: Básico

O perfil do usuário será representado por **'ux basics'**, sendo ux a área de interesse e basics (básicos) o nível do público alvo do curso.

Tokens da query:

['ux', 'basics']

Utilizando como entrada **‘ux basics’**, com a finalidade de encontrar um curso que ensine o básico sobre UX, teve como retorno o seguinte resultado:

Cursos recomendados:

Título: Wireframing For UI / UX Designers

Descrição: Simplify your UI / UX process with paper and digital wireframes

Similaridade: 0.828489363193512

Relevância: 0

Título: Lean UX: Faster, Better UX (User Experience) Design

Descrição: Design better and faster user experiences (UX) with Lean UX

Similaridade: 0.8075733780860901

Relevância: 0

Título: Adobe XD - UX UI design For beginners

Descrição: Adobe XD - UX UI Basics

Similaridade: 0.7924297451972961

Relevância: 1

Título: Adobe XD for Web Design: Essential Principles for UI UX

Descrição: Learn Responsive Web Design, UI UX Principles, Prepare The Design For Coding, Increasing Sales via A Great Design

Similaridade: 0.7726107835769653

Relevância: 1

Título: Adobe Illustrator CC Graphic Web Design: UI Logo Design

Descrição: Adobe Illustrator CC Graphic Design: Web Design, UI Design, Logo Design, Sketch UX Design For Designers Developers

Similaridade: 0.7722707986831665

Relevância: 1

Precisão: 3/5 = 60%

NDCG: 0,6183

Ordem ideal determinada: 3, 4, 5, 1 e 2

Discussão dos resultados:

O primeiro curso foca em *wireframing*, que é um dos processos principais de UX, mas não é abrangente para todos os conceitos básicos, o que pode ser menos interessante

para o usuário.

O segundo curso se encaixaria mais em conhecimentos avançados, para melhorar e otimizar o trabalho do profissional de UX.

O terceiro já apresenta um conteúdo focado em iniciantes, apresentando os básicos do Adobe XD.

O quarto curso é bem parecido com o terceiro, mesmos princípios.

O último é um curso sobre Illustrator CC, com vários conceitos e técnicas de web design utilizando a ferramenta, mas nada nele diz que é um curso iniciante, por isso é menos relevante.

CASO 3:

O entusiasta médio deseja aprender um instrumento novo, nesse caso o piano, e como é um instrumento mais complexo, resolveu pesquisar por algum curso que cubra os conhecimentos básicos para começar a tocar pequenas peças o mais rápido possível:

- Nome: Julio
- Área de interesse: Instrumentos
- Nível de conhecimento: Iniciante

O perfil do usuário será representado por **'piano iniciante'**, sendo piano a área de interesse e iniciante o nível do público alvo do curso.

Tokens da query:

['piano', 'iniciante']

Utilizando como entrada **'piano iniciante'**, com a finalidade de encontrar um curso com ensinamentos para principiantes no piano, teve como retorno o seguinte resultado:

Cursos recomendados:

Título: We ArtPlay Piano

Descrição: Composing and Improvising on the Piano

Similaridade: 0.977283239364624

Relevância: 0

Título: Piano With Willie: Piano Chords Vol. 1

Descrição: Learn to play chords on the piano

Similaridade: 0.9478051066398621

Relevância: 1

Título: Piano With Willie: Piano Chords Vol. 2

Descrição: Learn to play chords on the piano

Similaridade: 0.9478051066398621

Relevância: 1

Título: Guitar, Ukulele Piano for Praise Worship

Descrição: Playing for Worship

Similaridade: 0.9309934377670288

Relevância: 0

Título: O Hanon do Piano Popular

Descrição: Rítmica, Coordenação motora, Independência, Grooves

Similaridade: 0.9289443492889404

Relevância: 0

Precisão: 2/5 = 40%

NDCG: 0,5307

Ordem ideal determinada: 2, 3, 5, 4 e 1

Discussão dos resultados:

O primeiro resultado traz conhecimentos para pianistas mais avançados, como composição e improvisação, algo além do que busca um iniciante.

O segundo curso já conta com ensinamentos iniciais de acordes, ótimos para quem está começando.

O terceiro é como o segundo, mas é o próximo volume, podendo ser considerado um passo dois para o aprendizado de acordes.

O quarto resultado está trazendo uma gama de instrumentos, não apenas o piano, voltados para músicas de adoração, o que não é interessante para quem quer apenas aprender piano.

O último curso é o único em português, que foi o idioma utilizado nos termos de entrada do perfil do usuário, mas o conteúdo é para instrumentistas avançados, como grooves, aprendizado sobre ritmo, etc.

9. Conclusão

9.1. Considerações Finais

Depois de analisado o estado-da-arte, algumas implementações foram selecionadas dentre as que mais se destacaram, trazendo os resultados mais satisfatórios e mais recentes. Com destaque para a solução apresentada por [Vita and Cioffi 2022], com a filtragem híbrida, que conta com os dois maiores tipos de filtragem, um trazendo interações de usuários e o outro utilizando a PLN para reduzir descrições e textos em tokens e enriquecer as recomendações. Solução essa que resolve o problema de *cold start*, visto que quando o usuário é novo no sistema e não possui dados de interações, ainda há a filtragem por conteúdo para garantir que as recomendações sejam geradas.

As fontes de dados obtidas para desenvolver o trabalho foram satisfatórias, na medida do possível, mas poderiam ter sido melhor direcionadas e mais versáteis, para realizar mais do que o que foi feito. O primeiro dataset, de cursos da Udemy, é completo para realizar um trabalho de recomendação por conteúdo, mas não há disponível um segundo com dados de interações de usuário para realizar a recomendação colaborativa. Já o segundo dataset encontrado, da Coursera, que na verdade é um conjunto de dados de cursos e dados de avaliações, fornece muitos dados de interações entre usuários e cursos, mas não possui dados textuais para aplicar PLN na recomendação por conteúdo. Assim, cada um desempenhou muito bem seu papel, sendo cada uma das filtragens propostas, mas nenhum conseguiria realizar ambas, para que se pudesse até mesmo abordar a filtragem híbrida.

Na parte da PLN, como já foi explicado no parágrafo anterior, devido à natureza dos dados obtidos e à forma como funcionam as duas formas de filtragem estudadas e trabalhadas, as técnicas de PLN se fizeram relevantes apenas na filtragem baseada em conteúdo, pelo *dataset* utilizado conter os dados textuais que serviram de entrada para a geração das recomendações. Já no caso filtragem colaborativa, além de o *dataset* utilizado não conter colunas com dados textuais relevantes para o modelo, a abordagem escolhida elegeu apenas a coluna com os dados de avaliações numéricas como a entrada principal para o funcionamento do modelo, e também a única realmente relevante.

Com os experimentos realizados, a conclusão foi que os dois SR mostraram um resultado satisfatório, porém, com uma superioridade considerável para o colaborativo. Além do fato de que não há nenhuma garantia de que os cursos recomendados pelo SR de conteúdo são bons e bem produzidos, mas para o colaborativo há sim, já que são bem avaliados por usuários de perfis semelhantes. E com isso em mente, pode-se realizar um exercício de raciocínio. Quando se procura um curso *online*, as pessoas sempre priorizam a qualidade. Primeiro é feita a busca por um assunto, e depois de encontrar os cursos daquele tema, procura-se sempre o curso de melhor qualidade e mais bem avaliado. Dessa forma, a filtragem pelo assunto ocorre primeiro, e após isso é que entram as notas. E o tema que interessa um aluno geralmente é sempre o mesmo. De acordo com [Agarwal 2019], 75% das pessoas que estudam online estão buscando emprego ou mudança de carreira, por esse motivo faz sentido pensar que essas pessoas focam em uma área apenas, então provavelmente avaliam cursos sempre do mesmo tema. O que corrobora com o sucesso do SR colaborativo.

9.2. Trabalhos Futuros

Ao longo do desenvolvimento desta tese, diversos outros trabalhos relacionados se tornaram aparentes. Alguns dos trabalhos que julgamos mais interessantes encontram-se a seguir:

9.2.1. Sistema de recomendação completo

Ao levar em conta o primeiro SR baseado em conteúdo, um possível trabalho futuro, considerando os devidos contextos em que ele é inserido, seja uma plataforma *online*, ou um *app*, por exemplo, para que que ele deixe de ser apenas um "buscador" e passe a realizar o trabalho completo de um SR do mercado, a base de dados deve ter, além das informações

textuais de cada curso, os dados dos perfis de usuário para que sejam extraídos. Com isso, deixa de ser uma necessidade utilizar os dados anteriormente explicados, e se passa a ter disponível os perfis para gerar as recomendações.

9.2.2. Funcionamento do sistema híbrido

Para a ideia dos dois SR se complementando, ou seja, o sistema híbrido, um possível trabalho futuro é, ao invés de utilizar PLN e dados textuais para fazer o refinamento das recomendações, utilizar esses dados para realizar uma amostragem inicial, retornando cursos apenas daquele tema, e com esses cursos filtrados, aplicar a filtragem colaborativa para priorizar apenas os mais avaliados e de maior qualidade, seguindo o mesmo princípio de primeiro escolher um tema e depois manter apenas os melhores cursos.

9.2.3. Experimentos

Em futuros experimentos, deve-se também priorizar uma forma mais tradicional para a realização dos testes com os modelos, retirando do conjunto de treinamento cursos bem avaliados para algumas áreas e assuntos diferentes e usá-los como conjunto de testes.

9.2.4. Transcrição de conteúdo de cursos

Algumas plataformas de cursos disponibilizam o conteúdo das aulas transcritos para os alunos. Uma ideia interessante para um futuro trabalho seria obter esses arquivos de texto transcritos e utilizá-los para treinar o modelo baseado em conteúdo, o que seria de grande valia para a aplicação das técnicas de PLN.

References

Aamir, M. and Bhusry, M. (2015). Recommendation system: state of the art approach. *International Journal of Computer Applications*, 120(12).

Agarwal, P. (2019). Why is no one talking about employability in online education?

Balush, I., Vysotska, V., and Albota, S. (2021). Recommendation system development based on intelligent search, nlp and machine learning methods. In *MoMLeT+ DS*, pages 584–617.

Brown, S. (2021). Machine learning, explained.

Education, I. C. (2020). What is natural language processing?

Kanade, V. (2022). What is machine learning? definition, types, applications, and trends for 2022.

Kramer, O. (2013). *K-Nearest Neighbors*, pages 13–23. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kumar, S. (2021). Fundamental of matrix factorization for recommender system.

Lu, J., Wu, D., Mao, M., Wang, W., and Zhang, G. (2015). Recommender system application developments: a survey. *Decision Support Systems*, 74:12–32.

Murel, J. and Kavlakoglu, E. (2024). What is content-based filtering?

Naren, J., Banu, M. Z., and Lohavani, S. (2020). Recommendation system for students' course selection. In *Smart Systems and IoT: Innovations in Computing*, pages 825–834. Springer.

Raman, K. (2024). Matrix factorization explained — what is matrix factorization?

Vita, D. L. and Cioffi, G. (2022). Hybrid movie recommender system.

Wang, Z., Huang, H., Cui, L., Chen, J., An, J., Duan, H., Ge, H., Deng, N., et al. (2020). Using natural language processing techniques to provide personalized educational materials for chronic disease patients in china: development and assessment of a knowledge-based health recommender system. *JMIR medical informatics*, 8(4):e17642.

Wolff, R. (2020). Top 10 nlp tools and services in 2022.