



CIÊNCIAS EMPRESARIAIS

ESCOLA SUPERIOR
POLITÉCNICO SETÚBAL

PEDRO HENRIQUE
MELLO PEREIRA

Utilização de técnicas de Processamento de Linguagem Natural para construção de um sistema de recomendação baseado na similaridade de vetores de representação textual;

Relatório de projeto de investigação do Mestrado
em Ciência de Dados Para Empresas

ORIENTADOR

(Grau, David Alexandre Mendes Silva Simões)

Índice

Resumo	4
Abstract	5
Introdução	6
Contexto e Motivação	6
Objetivos	7
Metodologia	8
Estrutura do Relatório	8
1 Revisão de Literatura	9
1.1 Dos sistemas de recomendação	9
1.1.1 Tipos de Sistemas de Recomendação	11
1.2 Das técnicas de processamento de linguagem natural aplicadas ao caso concreto	14
1.2.1 Problemas passíveis de serem solucionados com uso do PLN	15
1.2.2 Pré-Processamento de dados textuais	15
1.2.2.1 Tokenização:	16
1.2.2.2 Remoção de stopwords:	16
1.2.2.3 Normalização de texto:	16
1.2.2.4 Lematização e stemming:	17

1.2.3	Representação textual em formato vetorial: Da frequência simples de palavras à semântica contextual.	17
1.2.3.1	CountVectorizer:	18
1.2.3.2	TF-IDF (<i>Term Frequency - Inverse Document Frequency</i>):	19
1.2.3.3	Word2Vec:	22
1.2.3.4	Transformers e <i>self-attention</i> :	23
1.2.3.4.1	BERT e SBERT: A espinha dorsal do sistema de recomendação baseado em Transformers	24
1.2.3.4.2	Mecanismo de funcionamento do SBERT:	24
2	Objetivos e Metodologia	26
2.1	Objetivos	26
2.2	Metodologia	28
2.2.1	Extração dos dados e Datasets utilizados	29
2.2.2	Pré-processamento	33
2.2.3	Compilação dos dados	34
2.2.4	Medida de Semelhança: A similaridade do Cosseno e porquê utilizá-la.	34
2.2.4.1	Outras medidas de similaridade	36
2.2.5	Construção das engines de recomendação: Desenvolvimento das Classes TfIdfRecommender e SBERTRecommender	37
2.2.6	Metodologia de avaliação das recomendações	47
3	Apresentação dos Resultados	48
4	USE CASE:	49
5	Conclusão e Investigação Futura	50
	Bibliografia	51
	Lista de Figuras	
1	Exemplo de Sistema de Recomendação Baseado em Conteúdo.	11

2	Exemplo de Sistema de Recomendação Baseado em Filtragem Colaborativa.	12
3	Fluxo de Pré-Processamento de Dados Textuais.	16
4	Classificação do Processamento de Linguagem Natural no campo da Inteligência Artificial.	17
5	Funcionamento do SBERT.	24
6	Fluxograma da metodologia utilizada	28
7	Exemplo de função para extração da API Google Books com uso de Python	32
8	Heatmap das 10 primeiras entradas da matriz de similaridade do cosseno, que possui dimensão total de 807×807 para o dataset de cursos.	36
9	Construtor da Classe TfIdfRecommender	38
10	Demonstração da função que extrai as recomendações	41
11	Construtor da Classe SBERTRecommender	42
12	Função responsável pela extração das recomendações	46

Lista de Tabelas

1	Resultado da Vectorização com CountVectorizer	19
2	Resultado da Vectorização com TFIDF	21
3	Amostra de matriz esparsa TFIDF aplicada ao Dataset de Cursos. Cada linha representa um documento(curso) do conjunto de dados, e cada coluna diz respeito a uma palavra do vocabulário.	21
4	Amostra de 10 linhas e 5 colunas da tabela de cursos	29
5	Amostra de 10 linhas da tabela de livros	33
6	Aplicação dos estágios de pré-processamento sobre a frase exemplificativa.	39



Resumo

qishfuiqfvuvfhuiqfhvuiqhdfvuihdfuivhduifvhdvufvhiuw hfdiuvhwifd



Abstract

sjkcbhquihvuiqehvuirhqeipvhdfiuvhidfhvkjsdnfvkjsdnfvkjnvkjdfvkjsdfvkjlf

Introdução

Esta secção tem por escopo apresentar de maneira estruturada o contexto e as motivações sob as quais repousa esta produção académica, de modo a expressar aos leitores e avaliadores os objetivos da investigação, sejam eles mediatos e/ou imediatos, a metodologia utilizada e a estrutura do relatório.

Contexto e Motivação

É público e notório que estamos a viver a era do Big Data. Todos os dias uma quantidade massiva de dados é produzida e despejada na internet pelos mais diversos dispositivos eletrônicos existentes. Ressalta-se que a maior parte destes dados enquadram-se nos chamados “dados não estruturados”, que são aqueles não possuem forma ou esquema pré-definidos e não constituem necessariamente vetores de características numéricas e/ou categóricas, como por exemplo textos, imagens, áudios e vídeos.


Por conseguinte, uma das consequências mais evidentes deste novo tempo diz respeito à mudança de paradigma do comércio de mercadorias, bens e serviços. Se antes da adoção da internet pela maior parte da população mundial o canal de vendas principal da maioria das empresas era o presencial, com a massificação do acesso à internet e o crescimento exponencial do comércio eletrónico, o ambiente digital assumiu um papel central nas estratégias de vendas. Logo, haja vista que a presença online tem se mostrado cada vez mais determinante para o sucesso do negócio, as empresas passaram a investir significativamente em suas plataformas na internet, de modo a permitir aos consumidores a aquisição de mercadorias, bens e serviços a partir de qualquer lugar e a qualquer momento.

Ocorre que, além das inúmeras novas oportunidades criadas para as empresas, este novo paradigma trouxe também importantes desafios, sendo o aumento da concorrência um dos mais proeminentes. A partir deste momento a concorrência tornou-se global e trouxe consigo a necessidade de inovação constante e personalização de ofertas para atender a um público cada vez mais exigente e diverso.

Ademais, ao imaginarmos um e-commerce com centenas ou até milhares de produtos, pode ser extremamente complexo para o cliente encontrar o item que está à procura, o que pode tornar a experiência de compra frustrante e desmotivadora e levá-lo à desistência.

Neste contexto é que surgem os sistemas de recomendação, os quais permitem a oferta personalizada de sugestões aos consumidores baseadas na similaridade entre os itens comprados e/ou pesquisados anteriormente, o que pode facilitar a experiência de compra online e impulsionar as vendas da empresa.

Há aqui entretanto uma questão: Sistemas de recomendação tradicionais podem exigir investimentos volumosos para recolha, tratamento e armazenamento de grandes quantidades de dados quanto às preferências



dos clientes, o que pode inviabilizar a implementação de tais ferramentas por empresas de menor porte ou que estejam a iniciar suas atividades com poucos recursos.

Desta feita, este trabalho tem por motivação demonstrar que o desenvolvimento e aplicação de sistemas de recomendação baseados nos dados gerados diariamente no ambiente comercial pode apresentar-se como uma grande vantagem competitiva, sobretudo ao utilizar-se dos dados textuais (não estruturados), que são abundantes e ainda sub-utilizados no que diz respeito à geração de conhecimento e inteligência empresarial. Com isto, pretende-se demonstrar uma maneira de desenvolver SRs que possam ser implementados por organizações que possuem poucos recursos para investir na aquisição de dados, diminuindo portanto a barreira de entrada para a adoção desta tecnologia, tornando-a mais acessível.

Objetivos

Considerando-se o contexto retromencionado, o presente estudo tem por objetivos:

- Investigar os avanços no campo do Processamento de Linguagem Natural, sobretudo no que diz respeito à aplicação destas técnicas à criação de sistemas de recomendação não supervisionados eficientes e baseados inteiramente na similaridade entre os itens;
- Desenvolver ao menos 2 (dois) protótipos de sistemas de recomendação baseados nas características textuais que descrevem os itens, utilizando-se para tanto de técnicas de PLN e similaridade do cosseno entre os vetores de representação textual gerados por cada um dos sistemas. Esses protótipos devem ser projetados para utilizar uma quantidade mínima de dados, recorrendo apenas às descrições e demais dados textuais relevantes sobre os itens, de modo que sejam especialmente adequados para empresas em estágio inicial ou com recursos limitados, vez que, diferentemente dos sistemas de recomendação tradicionais, que frequentemente dependem de investimentos significativos para coleta e armazenamento de dados comportamentais dos usuários, essa abordagem servirá para reduzir a barreira de entrada, oferecendo uma alternativa eficiente e acessível para recomendar itens de forma eficaz;
- Perceber a diferença entre vetores de representação textual(embeddings) contextuais e não contextuais e como isso afeta o resultado final por meio da extração e comparação de recomendações;

Metodologia

A primeira etapa da elaboração do projeto diz respeito à revisão da literatura para expansão do conhecimento sobre as particularidades dos sistemas de recomendação e as técnicas de PLN a serem empregadas. De seguida, decorreu a fase de levantamento dos requisitos e necessidades para preparar o ambiente de desenvolvimento da maneira mais adequada à prototipação dos sistemas de recomendação. Ainda nesta fase também foi realizada a extração e tratamento dos dados que foram utilizados no estudo, os quais foram obtidos de fontes públicas consideradas como seguras e confiáveis. Por fim, procedeu-se à elaboração dos protótipos dos sistemas de recomendação, os quais diferem-se por meio das técnicas utilizadas para extração dos vetores de representação textual, bem como à avaliação e comparação dos resultados para retirada das conclusões e entendimento quanto aos próximos passos.

Estrutura do Relatório

O primeiro capítulo deste relatório de projeto traz em seu cerne um estudo a respeito da literatura especializada sobre o tema, com vistas a clarificar conceitos basilares, tais como os tipos de sistema de recomendação, as etapas do processo de mineração textual com ênfase na evolução dos vetores de representação textual e como a similaridade do cosseno aplica-se ao caso em tela, e estabelecer uma linha de raciocínio clara e concisa de modo a guiar o leitor ao entendimento das etapas a seguir. Neste ponto apresentou-se em apertada síntese a evolução das técnicas de Processamento de Linguagem Natural durante as últimas décadas e o estágio em que se encontra atualmente, sendo este um dos campos da Inteligência Artificial que apresentou os maiores avanços na história recente.

No capítulo a seguir foram definidos com maior riqueza de detalhes os objetivos perseguidos no projeto, bem como justificou-se a metodologia utilizada na prototipação dos sistemas de recomendação e qual foi o método de avaliação de resultados escolhido.

O capítulo 3 diz respeito à apresentação dos resultados obtidos por meio da aplicação dos algoritmos desenvolvidos tanto à base de dados de cursos da plataforma EVG quanto à base dos mais de 29.000 (vinte e nove mil) livros oriunda da API do Google Books, de modo a avaliar o quão ajustadas e assertivas foram estas recomendações no que diz respeito à similaridade com os inputs apresentados a cada um dos sistemas.

Por último, apresentam-se no capítulo 4 as conclusões obtidas no estudo, bem como os desafios encontrados e os limites dos sistemas. Aqui encontram-se também as propostas de investigação futura e possíveis caminhos para implementação do protótipo desenvolvido que apresentou a melhor performance.

1 Revisão de Literatura

Após a introdução do projeto e exposição do contexto e motivação pelas quais fora produzido, a presente secção objetiva apresentar com maior profundidade as definições existentes na literatura especializada, tanto sobre os sistemas de recomendação, quanto sobre o processamento de linguagem natural enquanto sub-campo da inteligência artificial. Aqui serão apresentados os conceitos essenciais a respeito dos tipos de SRs existentes, as técnicas de vetorização e representação textual sob o prisma do PLN, a evolução do campo do PLN ao longo das últimas décadas, a intersecção entre as áreas do estudo e, finalmente, a fundamentação da métrica de similaridade escolhida para ser aplicada aos vetores de representação textual de modo a obter os resultados esperados.

1.1 Dos sistemas de recomendação

Segundo Ricci et al. (2011) Os Sistemas de Recomendação (SR) são ferramentas e técnicas de software que fornecem sugestões de itens que podem ser úteis para um utilizador. Tais sugestões tem como objetivo principal apoiar os utilizadores no processo de tomada de decisão, como, por exemplo auxiliando-os sobre qual livro comprar numa biblioteca virtual, qual curso obter para expandir o conhecimento em determinada área ou quais notícias mais o apeteçam em um sítio de notícias online.


Este tipo de sistema baseia-se principalmente nas interações entre:

- **Usuário:** Trata-se do sujeito que recebe as recomendações de itens por parte do fornecedor;
- **Fornecedor:** Aquele que recomenda os itens ao usuário em sua plataforma;
- **Item:** Aquilo que está a ser recomendado ao usuário;

Os SR podem desempenhar diferentes funções. Sob a ótica do fornecedor, um SR pode ser usado por exemplo para ampliar as vendas ou fidelizar os utilizadores da sua plataforma. Do ponto de vista do utilizador, o objetivo principal é encontrar itens relevantes, úteis e adequados às suas necessidades, sem precisar despendar muito tempo nesta tarefa.

Por conseguinte, de acordo com Herlocker et al. (2004), os SR podem ser sub-divididos quanto às suas funções mais populares em 9(nove) tarefas principais, dentre as quais destacamos 3(três), a saber:

- **Recomendação de alguns itens bons:** O SR recomendará uma lista classificada de itens que possuem maior chance de satisfazer as necessidades do usuário, com ou sem previsão explícita de avaliação.

- 
-
- **Anotação em contexto:** O sistema destaca alguns itens de uma lista com base nas preferências de longo prazo do utilizador.
 - **Recomendar uma sequência:** O sistema recomenda uma sequência de itens que seja agradável como um todo.

Ressalta-se que, apesar de as duas primeira hipóteses elencadas anteriormente serem as abordagens mais populares em termos de aplicação de um SR, há de se ter em consideração que estas necessitam de um enorme esforço e investimento na recolha, armazenamento e tratamento de dados de reviews e feedbacks dos usuários, o que pode traduzir-se em custos significativos para a organização que está a implementar este recurso em sua plataforma comercial. Tal abordagem é conhecida no jargão técnico como Filtragem Colaborativa, justamente por apoiar-se no cruzamento de grandes massas de dados dos usuários para encontrar usuários com preferências semelhantes.

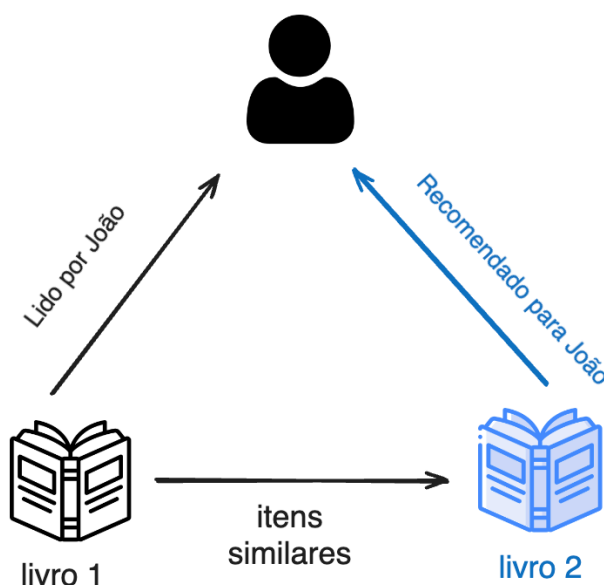
Desta feita, **este estudo concentra-se na hipótese de recomendação de sequência**, em que recomenda-se ao usuário os itens mais similares ao(s) último(s) item consumido ou pesquisado por ele. Desta maneira, a um usuário que consumiu um livro de estatística para machine learning deve se recomendado, por exemplo, um livro da linguagem R para Data Science. A esta abordagem denomina-se “Filtragem baseada no conteúdo”, a qual será abordada no tópico a seguir.

1.1.1 Tipos de Sistemas de Recomendação

Quando estamos a falar em sistemas de recomendação, existem 3(três) tipos de técnicas que podem ser utilizadas em sua concepção, as quais destacam-se a seguir:

a) *Content-based*:

Figura 1: **Exemplo de Sistema de Recomendação Baseado em Conteúdo.**



Fonte: Elaborado pelo autor.

Esta técnica baseia-se na recomendação dos itens mais similares àqueles adquiridos e/ou pesquisados pelo usuário anteriormente. Neste tipo de SR o que importa é a similaridade entre as características dos itens, de modo a utilizar unicamente o último e/ou histórico dos últimos itens consumidos pelo utilizador para recomendar itens semelhantes, sem considerar contudo as variáveis relativas ao próprio utilizador.

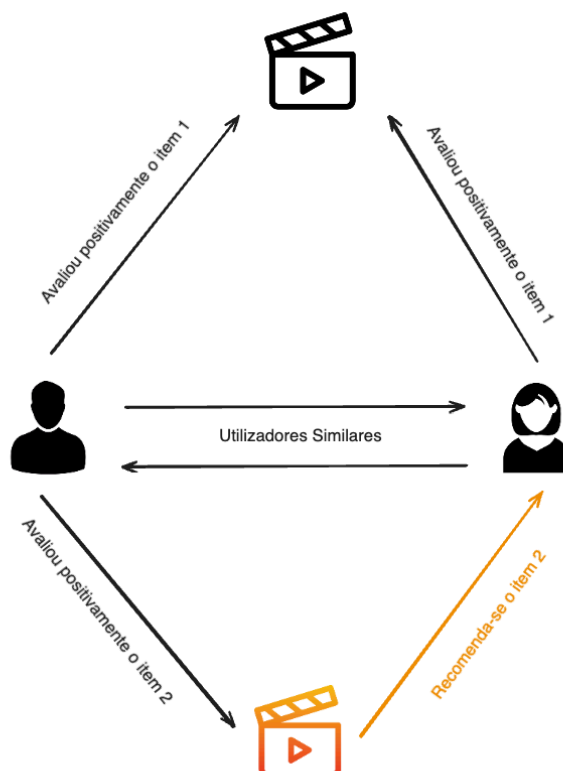
Conforme demonstra a Figura 1, imagine que o usuário João leu o livro “Os Segredos da Mente Milionária”. Com base nessa interação, o sistema analisaria as características do livro, como gênero (desenvolvimento pessoal), temas abordados (educação financeira e mentalidade de riqueza) e descrição textual da obra e recomendaria livros como “Pai Rico, Pai Pobre” de Robert Kiyosaki ou “A Mente Acima do Dinheiro” de Ted Klontz, que compartilham características similares e tendem a atrair o mesmo público-alvo.

Como principais vantagens deste tipo de sistema é possível citar a sua simplicidade e menor necessidade de dados, haja vista não ser necessário coletar grandes massas de dados dos usuários, bem como sua escalabilidade e elevada precisão em determinados tipos de domínio como por exemplo na recomendação de livros, cursos e filmes.

De outro giro, sua principal desvantagem diz respeito à limitação da descoberta de novas preferências pelos utilizadores, uma vez que ao receber apenas recomendações de itens similares dificilmente ocorreria uma exposição a itens diversos poderiam expandir seus interesses. Além disso, há uma elevada dependência da qualidade e granularidade das características que descrevem os itens para que se tenham boas recomendações.

b) *Collaborative-filtering*:

Figura 2: Exemplo de Sistema de Recomendação Baseado em Filtragem Colaborativa.



Fonte: Elaborado pelo autor.

Se na técnica baseada em conteúdo os dados relativos ao usuário não eram considerados, há aqui uma completa inversão desta lógica: A filtragem colaborativa utiliza os dados recolhidos de todos os utilizadores para recomendar a um determinado usuário conteúdos que pessoas com o perfil semelhante ao dele

consumiram anteriormente. Schafer et al. (2001) refere-se à filtragem colaborativa como “people-to-people correlation”, vez que a similaridade entre os gostos de dois usuários é calculada com base no quão similares são as avaliações atribuídas a itens compartilhados.

A Figura 2 traz um exemplo do funcionamento desta abordagem. Imagine que o utilizador Pedro assistiu e avaliou positivamente o filme “A Origem”. O sistema detecta que outros usuários que também gostaram desse filme, como Maria, assistiram a “Blade Runner 2049” e o avaliaram positivamente. Com base nessa correlação de gostos, o sistema recomenda “Blade Runner 2049” para Maria, considerando que seus interesses são semelhantes aos de Pedro.

A filtragem colaborativa possui diversas vantagens. Este tipo de SR é capaz de reconhecer padrões complexos nas preferências dos usuários sem necessitar de informações detalhadas a respeito dos itens. Sua versatilidade permite aplicação em vários domínios, o que facilita a sugestão de novos itens que não compartilham características explícitas, de modo a ampliar o escopo das recomendações. Além disso, promove descobertas inesperadas ao basear-se nas interações entre usuários, pois expõe os utilizadores a opções que podem não estar diretamente ligadas às suas escolhas anteriores, mas que se alinham com as preferências de usuários semelhantes.


Contudo, a filtragem colaborativa também enfrenta desafios importantes. Um deles é o problema do “cold start”, onde a falta de dados históricos dificulta as recomendações iniciais para novos usuários. A eficácia do sistema também depende sobremaneira da quantidade e qualidade dos dados existentes, especialmente em cenários com poucas interações registradas. Ademais, o processamento de grandes volumes de dados para calcular similaridades pode ser computacionalmente custoso, sobretudo para pequenas organizações. Por fim, é salutar ressaltar o risco de um viés de popularidade, onde itens muito avaliados apresentam tendência a serem mais recomendados, enquanto opções menos conhecidas podem ser negligenciadas.

c) Sistema Híbrido

Ricci et al. (2011) define sistemas híbridos como aqueles que combinam as técnicas mencionadas anteriormente. Segundo os autores, esses sistemas buscam aproveitar os pontos fortes de uma abordagem (A) para mitigar as desvantagens da outra (B).

Conforme ressaltado anteriormente, sistemas de recomendação baseados em filtragem colaborativa encontram problemas quando novos itens entram na base, vez que nunca foram avaliados por nenhum usuário e por isso podem acabar por não ser recomendados a ninguém por insuficiência de dados. De outro giro, isto não ocorre aos sistemas content-based, vez que mesmo que um novo item seja inserido geralmente as características que o descrevem estarão disponíveis e servirão para elaborar recomendações.

Tratar de maneira eficiente do problema do *cold start*, conforme abordado no parágrafo anterior, consiste em uma das principais vantagens deste sistema. Além disso, este tipo de SR é o que tende a apresentar as



recomendações mais personalizadas ao gosto do usuário, pois utiliza-se tanto de dados e padrões extraídos das interações coletivas, quanto dos dados que caracterizam cada item.

Ocorre que, apesar de seu funcionamento aparentemente perfeito, este sistema traz consigo uma importante desvantagem que pode tornar-se ainda mais desafiadora à medida em que as organizações que visam aplicá-lo possuem recursos limitados: Sistemas de Recomendação Híbridos possuem grande dificuldade de implementação e manutenção, haja vista a multiplicidade de técnicas aplicadas e a grande quantidade de dados que devem ser processados. Assim, torna-se custoso computacionalmente colocá-los em produção, sobretudo no contexto de empresas menores e com maiores limitações de budget.


1.2 Das técnicas de processamento de linguagem natural aplicadas ao caso concreto

Kamath & Kanakaraj (2015) propuseram um sistema de recomendação de notícias eletrônicas (e-news) que utiliza técnicas de Processamento de Linguagem Natural (PNL) para melhorar a experiência do utilizador em mecanismos de pesquisa. O sistema aborda a limitação das técnicas tradicionais baseadas em bag-of-words, que se baseiam na correspondência exata de palavras-chave, o que pode resultar na omissão de documentos potencialmente relevantes que não contêm as palavras-chave exatas da consulta do utilizador.

O algoritmo proposto pelos autores utiliza-se de uma abordagem semântica bag-of-words aprimorada, incorporando sinónimos de WordNet (Miller et al. (1990)) para aumentar a cobertura da pesquisa e capturar artigos de notícias potencialmente relevantes. Ao considerar sinónimos, o sistema expande o alcance da correspondência de palavras-chave, aumentando a probabilidade de recuperar documentos relevantes que podem usar termos diferentes para descrever o mesmo conceito, o que confere maior flexibilidade e alcance ao algoritmo. Por conseguinte, o sistema realiza agrupamento de documentos com base em vetores TF-IDF, os quais terão a essência de seu funcionamento detalhada adiante neste trabalho.

O estudo destaca em seu cerne o potencial das técnicas de PNL para melhorar os sistemas de recomendação de notícias eletrônicas, fornecendo aos usuários resultados de pesquisa mais completos e precisos.

Apesar de ainda serem importantes e relevantes até os dias de hoje, técnicas como o TF-IDF não captam contexto quando da vetorização textual e tem dado lugar a abordagens mais modernas, baseadas sobretudo em redes neurais e arquitetura de transformers para captação avançada de contexto por meio de mecanismos de atenção.



Esta subsecção abordará as nuances do Processamento de Linguagem Natural de modo a demonstrar a relevância deste ramo da Inteligência Artificial no desenvolvimento dos sistemas de recomendação não supervisionados baseados em conteúdo ora propostos, preparando o leitor para compreender a metodologia utilizada, a qual será apresentada na próxima secção.

1.2.1 Problemas passíveis de serem solucionados com uso do PLN

A primeira etapa de qualquer projeto que utilize PLN diz respeito à identificação do problema, ou seja, definir o tipo de tarefa a ser desenvolvida. Ressalta-se que cada uma dessas tarefas exige abordagens específicas de pré-processamento e vetorização, com vistas a garantir que os modelos consigam extrair significado dos textos de forma eficiente para a consecução do objetivo.

No caso em tela, o foco está na similaridade de sentenças, essencial para a construção de sistemas de recomendação baseados em conteúdo. Esse tipo de sistema compara textos e sugere itens com base na semelhança semântica entre seus textos de apresentação.

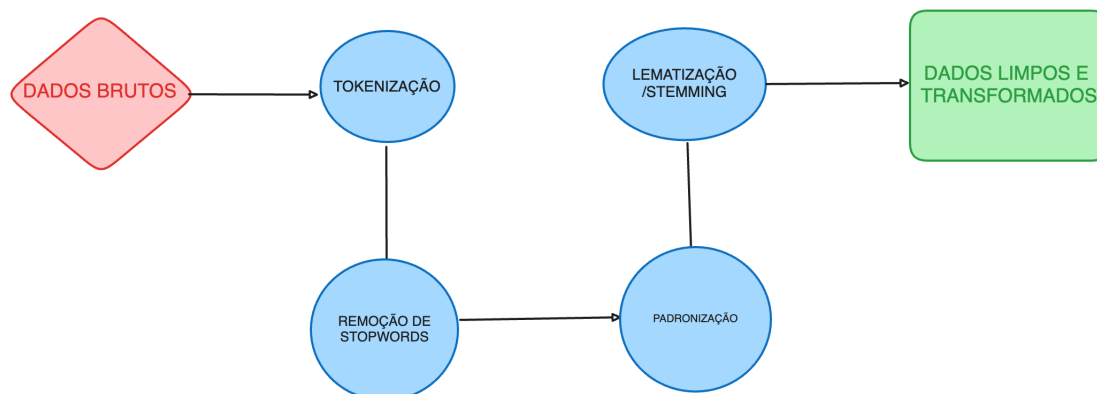
Alguns exemplos de tarefas em PLN e suas aplicações são:

- **Classificação Textual:** Categorização automatizada de textos de acordo com as suas áreas temáticas;
- **Reconhecimento de Entidades Nomeadas (NER):** Identificação de nomes de pessoas, locais, empresas dentre outros em um texto, útil sobretudo para extração automatizada de informações em notícias e documentos.
- **Busca Semântica:** Recuperação de arquivos ou documentos com base no significado das consultas;
- **Análise de Sentimentos:** Classificação automatizada de determinado texto quanto a sua polaridade;

1.2.2 Pré-Processamento de dados textuais

Após a identificação do problema que se deseja resolver com o uso do PLN, a próxima etapa que deve ser considerada diz respeito ao pré-processamento, que também pode ser percebido como o estágio em que se realiza a limpeza e transformação dos dados textuais. Esta etapa é essencial para transformar o texto bruto em um formato adequado para análise e modelagem, visando garantir que ruídos e inconsistências não comprometam o desempenho do sistema.

Figura 3: Fluxo de Pré-Processamento de Dados Textuais.



Fonte: Elaborado pelo autor.

A Figura 3 demonstra um fluxo com algumas das técnicas de pré-processamento mais proeminentes no âmbito da mineração textual, técnicas estas que serão expostas a seguir, com o objetivo de demonstrar os passos que devem ser seguidos no desenvolvimento de soluções eficazes no âmbito do PLN.

1.2.2.1 Tokenização:

Trata-se do processo de divisão do texto em unidades menores, conhecidas como **Tokens**, as quais podem ser palavras, símbolos, números ou outros tipos de caracteres.

1.2.2.2 Remoção de stopwords:

Stopwords são palavras comuns e pouco informativas que aparecem diversas vezes ao longo de um texto, como os pronomes, artigos e preposições. A depender da tarefa pretendida, estas palavras devem ser removidas pois não contribuem de maneira significativa para o enriquecimento do significado textual e podem gerar ruídos que atrapalham a performance dos modelos de PLN.

1.2.2.3 Normalização de texto:

Este estágio consiste em padronizar as palavras do texto de modo a trazer uniformidade ao vocabulário. Neste ponto todo o texto é convertido para letras minúsculas ou maiúsculas (a depender do caso) pondendo incluir-se ainda a remoção de caracteres especiais, sinais de pontuação e acentuação, para que variações desnecessárias sejam eliminadas.

Isto é especialmente importante quando da utilização de técnicas como o CountVectorizer ou TF-IDF, vez que evita que palavras com pequenas variações sejam tratadas como distintas, reduzindo a dimensionalidade do vocabulário e tornando a representação dos textos mais eficiente, além de gerar vetores mais consistentes.

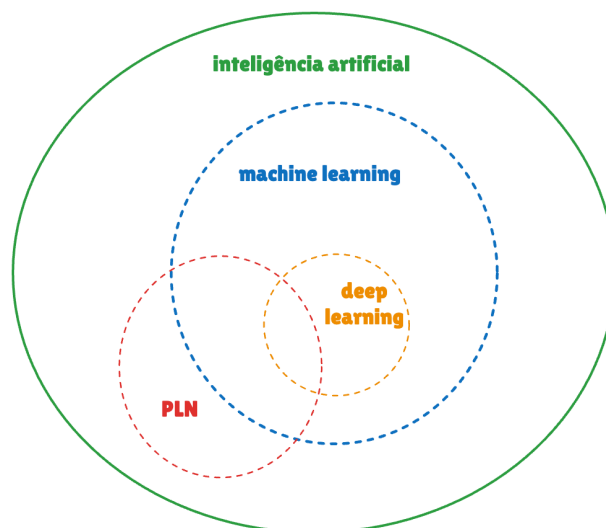
1.2.2.4 Lematização e stemming:

A lematização consiste em reduzir as palavras à sua forma base (lema), levando em conta o contexto e a gramática. Esta técnica de pré-processamento usa dicionários linguísticos para garantir que a palavra resultante seja válida. Por exemplo, as palavras “estudando”, “estudou” e “estudam” são todas reduzidas a “estudar”, pois essa é a **forma canônica do verbo**.


O stemming, por sua vez, é uma técnica mais simples, que elimina sufixos para reduzir palavras à sua forma raiz, sem contudo considerar a gramática. Ele usa regras fixas para truncar as palavras, o que pode gerar formas sem sentido. No exemplo anterior, “estudando”, “estudou” e “estudam” seriam reduzidos a “estud”. Embora o stemming reduza o vocabulário de forma eficiente, sua falta de precisão pode ser um problema em algumas aplicações.

1.2.3 Representação textual em formato vetorial: Da frequência simples de palavras à semântica contextual.

Figura 4: Classificação do Processamento de Linguagem Natural no campo da Inteligência Artificial.



Fonte: Elaborado pelo autor.



Muito tem se falado a respeito da Inteligência artificial generativa, sobretudo após o advento dos mecanismos de atenção (Vaswani (2017)), entretanto o que nem todos estão cientes é que a IA é um grande domínio do qual fazem parte a aprendizagem de máquina, o aprendizado profundo e o Processamento de Linguagem Natural, sub-divisão de extrema importância na concepção das IAs Generativas.

O PLN engloba um conjunto de métodos computacionais que permitem que as máquinas compreendam, interpretem e manipulem a linguagem humana. Entretanto, a compreensão textual pelas máquinas não ocorre da mesma maneira que ocorre nos humanos, mas através de métodos computacionais que transformam as palavras e/ou sentenças em números, aos quais denomina-se vetores de representação textual, o que possibilita que os algoritmos interpretem e extraiam importantes insights dos textos escritos em linguagem natural.

Desta forma, a vetorização textual desempenha um papel central no Processamento de Linguagem Natural, sendo responsável por transformar palavras e sentenças em representações numéricas, chamadas vetores de representação textual. Essa etapa é essencial para que os algoritmos possam interpretar e extrair insights de textos escritos em linguagem natural. Conforme demonstrado anteriormente, o processo inicia-se na definição do problema, passando então ao pré-processamento, no qual o texto é dividido em unidades menores chamadas tokens, que passam por limpeza, padronização, e pela remoção de palavras com baixo valor semântico (stopwords). Após essa preparação, são utilizados métodos que variam em complexidade e eficácia para geração dos vetores. Tais métodos evoluíram significativamente ao longo do tempo, a começar por abordagens simples, baseadas na contagem de palavras e vocabulário, e culminando em modelos sofisticados baseados em mecanismos de atenção, que capturam nuances semânticas profundas e contextuais. A evolução destes métodos será detalhada a seguir.

1.2.3.1 CountVectorizer:

Trata-se de um dos primeiros métodos utilizados no campo do PLN para transformar textos em representações numéricas passíveis de serem percebidas pelas máquinas. O CountVectorizer consiste em criar uma espécie de vocabulário com todas as palavras únicas existentes em um texto e, após isso, realiza-se a contagem da frequência de cada uma daquelas palavras ao longo do texto formando-se uma matriz onde cada linha representa um documento e cada coluna representa uma palavra do vocabulário formado.

A seguir um exemplo deste tipo de representação:

- a) **“Eu Gosto de PLN”**
- b) **“PLN é incrível”**

O Count-Vectorizer representaria as sentenças como:

Tabela 1: Resultado da Vectorização com CountVectorizer

	Eu	gosto	de	PLN	é	incrível
Documento a	1	1	1	1	0	0
Documento b	0	0	0	1	1	1

Apesar de ser simples e de fácil implementação, as desvantagens desta abordagem são muitas, dentre as quais destacam-se: Alta sensibilidade a ruídos e ortografia, não captação de contexto, ordem das frases e relação entre palavras, alta dimensionalidade dos dados a depender do tamanho do vocabulário, geração de matrizes esparsas e a importância exacerbada dada às palavras mais frequentes em detrimento das menos frequentes, problema este que ensejou o desenvolvimento da próxima técnica a ser citada neste trabalho;

1.2.3.2 TF-IDF (Term Frequency - Inverse Document Frequency):

Assim como o *CountVectorizer*, o TF-IDF também é um método de transformação de textos em valores numéricos com o intuito de representar a linguagem natural junto às máquinas. Ocorre que, diferentemente da abordagem anteriormente mencionada segundo a qual apenas a frequência absoluta das palavras importava para a formação dos vetores, o TF-IDF tem por objetivo reduzir a importância de termos muito comuns e destacar aqueles que podem ser mais relevantes.

Aqui, as palavras com maior número de ocorrências ao longo do documento tendem a ser tratadas como menos relevantes e têm sua importância diluída, de modo a conferir maior destaque àquelas palavras com menor ocorrência ao longo do texto que está a ser vetorizado. Para tanto, utilizam-se as seguintes fórmulas:

Term Frequency (TF)

A frequência de termo (TF) é a frequência relativa do termo (t) dentro do documento (d):

$$tf(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

onde,

- ($f_{t,d}$) é o número de vezes que o termo (t) aparece no documento (d),
- ($\sum_{t' \in d} f_{t',d}$) é o número total de termos no documento (d).

Inverse Document Frequency (IDF)

A frequência inversa de documento (IDF) mede a importância de um termo em uma coleção de documentos, e é calculada como:

$$idf(t) = \log \left(\frac{N}{1 + |\{d \in D : t \in d\}|} \right)$$

onde,

- (N) é o número total de documentos,
- ($|\{d \in D : t \in d\}|$) é o número de documentos que contêm o termo (t),
- O número 1 é adicionado ao denominador para evitar divisão por zero no caso de um termo não aparecer em nenhum documento.

Por fim, a métrica TF-IDF é então o produto da frequência do termo pela frequência inversa de documento:

$$tfidf(t, d) = tf(t, d) \times idf(t)$$

A seguir uma tabela exemplificativa do resultado da aplicação desta técnica em três documentos¹ diferentes, quais sejam:

- “1. Este jogador é muito rápido e habilidoso.”
- “2. Este jogador não é rápido mas é habilidoso.”
- “3. Este jogador é talentoso e não é rápido.”

Tabela 2: Resultado da Vectorização com TFIDF

	este	habilidoso	jogador	mas	muito	não	rápido	talentoso
doc1	0.364544	0.469417	0.364544	0.000000	0.617227	0.000000	0.364544	0.000000
doc2	0.329995	0.424929	0.329995	0.558731	0.000000	0.424929	0.329995	0.000000
doc3	0.364544	0.000000	0.364544	0.000000	0.000000	0.469417	0.364544	0.617227

Por ser uma das técnicas mais utilizadas nos campos da busca e classificação textual, decidiu-se testar esta técnica no desenvolvimento de um dos sistemas de recomendação deste trabalho, o qual será apresentado na secção da Metodologia.

Entretanto, apesar de ser uma das técnicas de vetorização textual mais importantes para o PLN ao longo do tempo e de apresentar uma notável evolução no que diz respeito à captação da importância dos termos para o sentido do documento, vez que reduz drasticamente a influência de termos corriqueiros como os artigos e preposições, o TF-IDF ainda possui desvantagens, dentre as quais destacam-se:

- **Não captação de contexto;**
- Ainda possui elevada sensibilidade a variações ortográficas;
- **Geração de matrizes esparsas, conforme exemplo da Tabela 3, as quais exigem elevado poder computacional para processamento;**
- A ideia de que os termos mais frequentes ao longo dos documentos são os menos importantes nem sempre será verdadeira.

Tabela 3: Amostra de matriz esparsa TFIDF aplicada ao Dataset de Cursos. Cada linha representa um documento(curso) do conjunto de dados, e cada coluna diz respeito a uma palavra do vocabulário.

	anexo	animados	animais	ano	anomalias	anormalidade
0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0

Continued on next page


¹ Documento neste contexto diz respeito à porção total de texto presente em determinada linha do dataframe

	anexo	animados	animais	ano	anomalias	anormalidade
2	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0
...
802	0.0	0.0	0.0	0.0	0.0	0.0
803	0.0	0.0	0.0	0.0	0.0	0.0
804	0.0	0.0	0.0	0.0	0.0	0.0
805	0.0	0.0	0.0	0.0	0.0	0.0
806	0.0	0.0	0.0	0.0	0.173172	0.0

1.2.3.3 Word2Vec:

Nas palavras de Mikolov et al. (2013) “Many current NLP systems and techniques treat words as atomic units – there is no notion of similarity between words, as these are represented as indices in a vocabulary”.

Conforme dito pelos criadores do Word2Vec, as abordagens de vetorização textual apresentadas até então possuíam a desvantagem inerente de criação de vetores esparsos e demasiado grandes, além de não capturar nenhum tipo de relação de proximidade de significado entre as palavras do vocabulário. Desta forma, o Word2Vec foi proposto como uma alternativa revolucionária por Mikolov et al. no ano de 2013. Este método introduziu a ideia de representações distribuídas de palavras, onde cada termo é mapeado em um espaço vetorial contínuo, permitindo a captação de nuances semânticas e sintáticas entre termos relacionados.



Isto tornou-se possível por meio da utilização de um modelo de linguagem baseado em uma rede neural probabilística do tipo feed forward para treinamento dos word embeddings do word2vec, com vistas a aprender representações densas de palavras diretamente a partir de grandes volumes de texto de maneira eficiente.

Esta técnica baseia-se em duas arquiteturas principais, conhecidas como Continuous Bag-of-Words (CBOW) e Skip-gram. Enquanto o CBOW prevê a palavra atual com base no contexto circundante, o Skip-gram tenta prever as palavras ao redor com base em uma única palavra central. Ambos os modelos utilizam camadas de projeção compartilhadas para mapear palavras em vetores de dimensão fixa, o que resulta em representações compactas e informativas.

Como vantagens desse modelo de geração de embeddings é possível citar a sua alta eficiência do ponto de vista computacional, sobretudo se comparado às abordagens anteriores, e a sua capacidade de captar relações semânticas entre as palavras do corpus.

Apesar de ser considerado como um marco importante no campo da vetorização textual, o word2vec também enfrenta limitações, dentre as quais destacam-se a necessidade de dados em demasia para treinamento, pois não apresenta bons resultados se treinado com poucos dados e o desafio de lidar com a polissemia e termos raros/novos que não foram inseridos no modelo durante a etapa de treinamento.

1.2.3.4 Transformers e self-attention:

Introduzida no ano de 2017 por Vaswani et al. (2017), a arquitetura de transformers é considerada como um dos grandes marcos da história do Processamento de Linguagem Natural. Esta arquitetura superou as limitações de modelos anteriores, como RNNs e LSTMs, ao utilizar mecanismos de auto-atenção para capturar relações e dependências globais entre os termos em uma frase. A partir desse avanço, os Transformers passaram a permitir a formação de vetores de representação textual dinâmicos e contextualizados, resolvendo um dos maiores desafios do PLN até então: a captação eficiente de contexto.

Esta técnica de vetorização textual possibilita o processamento paralelo completo de sentenças, de modo a captar o contexto bidirecional. Ao contrário de modelos como Word2Vec, que geram representações estáticas e unidirecionais (da esquerda para a direita), a arquitetura de Transformers considera tanto o contexto à esquerda quanto à direita simultaneamente, gerando representações dinâmicas e contextualizadas.

Os mecanismos de autoatenção em modelos como os Transformers permitem que cada palavra em uma frase “preste atenção” em todas as outras palavras, incluindo ela mesma, para capturar dependências e relações contextuais. Tal técnica baseia-se nos conceitos de query, key e value. Em apertada síntese, cada palavra gera uma query, que é comparada com as keys de todas as outras palavras para calcular pesos de relevância. Esses pesos são então aplicados às values correspondentes, resultando em uma representação contextualizada da palavra.

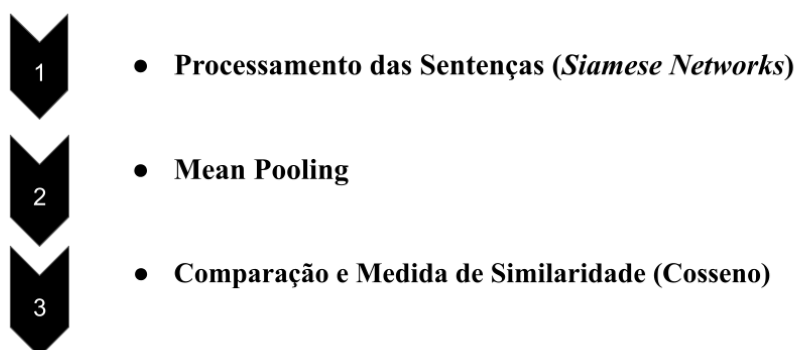
Além das vantagens citadas anteriormente, este novo paradigma também mostra-se muito relevante pela sua flexibilidade, vez que além da aplicação no desenvolvimento de modelos de **Large Language Models** como GPT e **BERT**, esta arquitetura pode ser adaptada a diversas tarefas no campo do PLN como tradução, busca semântica, classificação textual e, sobretudo, à tarefa de similaridade de sentenças a qual constitui o núcleo dos sistemas de recomendação desenvolvidos neste trabalho.

1.2.3.4.1 BERT e SBERT: A espinha dorsal do sistema de recomendação baseado em Transformers


Introduzido por Devlin et al. (2019), o BERT(*Bidirectional Encoder Representations from Transformers*) é um modelo de linguagem baseado na arquitetura de transformers. Este importante modelo foi pré-treinado em grandes corpora de texto usando duas tarefas principais: Masked Language Modeling (MLM), onde palavras são ocultadas e o modelo deve prevê-las com base no contexto, e Next Sentence Prediction (NSP), que ensina o modelo a entender relações entre pares de sentenças.

Ocorre que o BERT, apesar de suas inúmeras vantagens, possui algumas limitações que o tornam menos eficiente para tarefas específicas, como o cálculo de similaridade entre sentenças. Primeiramente, o BERT exige um grande poder de processamento devido à sua arquitetura complexa e ao mecanismo de self-attention, que tem custo computacional quadrático em relação ao comprimento da sequência. Além disso, o BERT não foi concebido originalmente para a tarefa de similaridade entre sentenças, uma vez que sua finalidade precípua é a previsão de palavras ocultas (*Masked Language Modeling*) e a compreensão de relações entre pares de sentenças (*Next Sentence Prediction*). Como resultado, para comparar duas sentenças, o BERT precisa processá-las em conjunto, o que é computacionalmente ineficiente e pode representar um risco para aplicações em larga escala, como sistemas de recomendação.

Figura 5: Funcionamento do SBERT.



Fonte: Elaborado pelo autor



1.2.3.4.2 Mecanismo de funcionamento do SBERT: Com o objetivo de adaptar o BERT para a realização de tarefas relativas à similaridade entre sentenças e busca semântica, o SBERT (Sentence-BERT) foi desenvolvido. Apresentado em agosto de 2019 no artigo *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks* Reimers & Gurevych (2019), o SBERT é uma variação do BERT projetada especificamente para tarefas de comparação de sentenças. Diferente do BERT, que processa pares de sentenças em conjunto, o SBERT gera embeddings independentes e semanticamente densos para cada sentença, que podem ser comparados de forma eficiente por meio de métricas como a similaridade do cosseno. Conforme explanação constante à Figura 5, essa abordagem é viabilizada por uma arquitetura siamesa, onde o BERT é ajustado para produzir representações vetoriais contextualizadas. Soma-se a isso uma camada de mean pooling que auxilia na criação de um único vetor de embeddings que representa toda a sentença.

No âmbito deste projeto, o SBERT é especialmente vantajoso, pois possui a capacidade de calcular a similaridade entre descrições de textos de maneira rápida e precisa, tornando-o interessante para aplicação em sistemas de recomendação baseados em conteúdo. Desta feita, justifica-se sua utilização que será demonstrada na secção de metodologia deste trabalho, comparando-o a um sistema com arquitetura mais simples baseado em TF-IDF.

2 Objetivos e Metodologia

Nesta secção, apresentam-se com detalhes os objetivos deste projeto e as estratégias metodológicas adotadas para sua realização. O trabalho tem como propósito desenvolver dois sistemas de recomendação de cursos utilizando técnicas de Processamento de Linguagem Natural (PLN), buscando identificar com precisão a similaridade entre os itens. Para isso, são empregadas duas bases de dados: uma composta por cursos da plataforma Escola Virtual de Governo (EVG) com 612 entradas e outra contendo livros em língua portuguesa extraídos da API Google Books, contendo 29.815 entradas.

Para isso, detalha-se a partir deste momento o processo de coleta, limpeza e estruturação dos dados, além dos métodos utilizados para a geração das recomendações. O estudo explora duas abordagens distintas para a vetorização textual: o TF-IDF, uma técnica mais tradicional, e o SBERT, baseado em redes neurais do tipo transformers. O objetivo é comparar o desempenho de ambas as metodologias e avaliar qual delas oferece recomendações mais precisas e relevantes.

Por conseguinte, apresenta-se o arcabouço técnico adotado, sobretudo no que concerne às ferramentas utilizadas para processamento dos dados e extração dos resultados. Por fim, são explicitados os critérios para avaliação do desempenho dos modelos e o prisma sob o qual os resultados são analisados.

2.1 Objetivos

- a) Investigar os avanços no campo do PLN, com ênfase na aplicação criativa destas técnicas para criação de sistemas de recomendação não supervisionados e totalmente baseados na similaridade textual entre os itens. O cerne neste ponto é analisar o quão eficaz pode ser essa abordagem, sem a necessidade de recolha de dados comportamentais ou feedback explícito pelos usuários, de modo a comprovar a sua relevância para empresas que possuem orçamentos modestos e/ou limitados para investimentos em infraestrutura de coleta, armazenamento e processamento de dados;
- b) Desenvolver um sistema de recomendação baseado na técnica TF-IDF para extração de vetores de representação textual esparsos, onde seja inputado como dado de entrada um determinado item e sejam emitidas ao menos duas recomendações de itens similares, tanto para a base de cursos da EVG quanto para a base de livros;
- c) Desenvolver um sistema de recomendação baseado em arquitetura de Transformers para extração de vetores de embeddings densos, onde seja inputado como dado de entrada um determinado item e sejam emitidas ao menos duas recomendações de itens similares, tanto para a base de cursos da EVG quanto para a base de livros;

✦

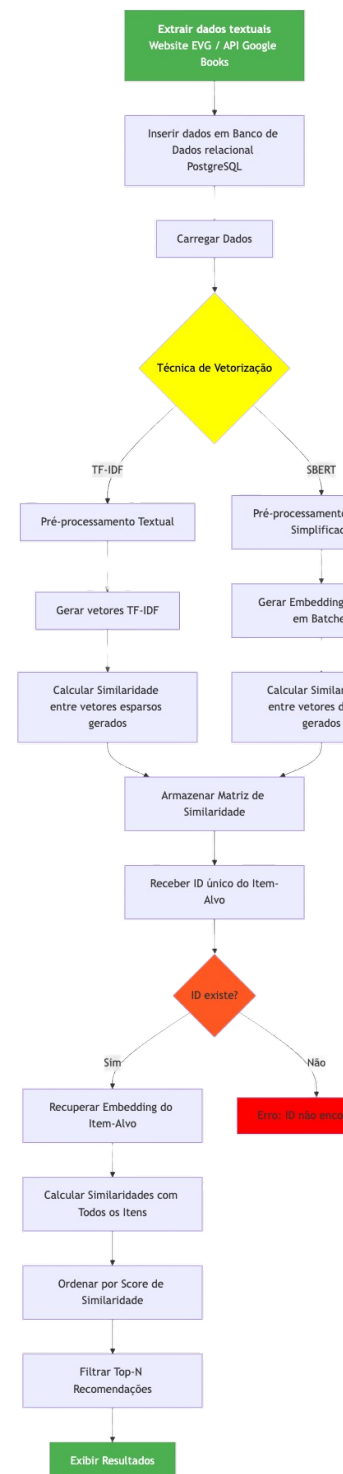
d) Comparar o desempenho de ambos os sistemas no que diz respeito a:

- Qualidade das recomendações emitidas;
- Capacidade de percepção de contexto e nuances semânticas. O que interessa ao projeto neste ponto é perceber a diferença entre vetores de representação textual contextuais(SBERT) e não contextuais(TF-IDF) e como isso afeta o resultado final no que tange à qualidade das recomendações exaradas;
- Capacidade de generalização em diferentes domínios;
- Capacidade de recomendação de sequências naturais de itens (Por exemplo, se lhe é inputado como dado de entrada o item “Harry Potter e a Câmara Secreta” o sistema deve ser capaz de recomendar naturalmente o próximo item da sequência, como “Harry Potter e o Prisioneiro de Azkaban”, além de recomendar itens semelhantes, como outros livros de fantasia similares à série do exemplo);
- Tempo necessário para processamento dos dados e emissão das recomendações tanto na base menor(cursos) quanto na base maior(livros);

e) Finalizar o projeto tendo avançado significativamente no entendimento do funcionamento e aplicação das principais técnicas de PLN

2.2 Metodologia

Figura 6: Fluxograma da metodologia utilizada



A Figura 6 demonstra o fluxo metodológico utilizado ao longo deste projeto. Os passos serão detalhados a seguir em pontos específicos.

2.2.1 Extração dos dados e Datasets utilizados

A etapa inicial do estudo envolve a extração e o armazenamento dos dados textuais necessários para o desenvolvimento e avaliação dos sistemas de recomendação. Para testar a eficácia dos modelos em diferentes cenários, foram selecionados dois conjuntos de dados distintos: o primeiro, de menor volume, é composto pela matriz de cursos disponíveis no website da Escola Virtual de Governo (EVG); o segundo, significativamente maior, contém livros em língua portuguesa extraídos da API do Google Books. A escolha desses datasets permite avaliar o desempenho dos sistemas tanto em bases reduzidas quanto em conjuntos de dados mais extensos e diversificados.

a) Dataset “Cursos”

Tabela 4: Amostra de 10 linhas e 5 colunas da tabela de cursos

id curso	nome curso	apresentacao	conteudo programatico
1328	Processamento de Linguag...	Descubra como o Processa...	Módulo 1 - Análise de te...
1329	Treinamento Censo Superior	O Censo da Educação Supe...	Módulo 1 - O Inep e os C...
1276	Armamento Institucional	Nesse curso você conhece...	Módulo 1 - Normas Gerais...
1231	Elaboração de editais em...	O curso capacita profiss...	Módulo Introdução ;Módu...
936	GTFS, GPS e bilhetagem e...	Neste curso, você conhe...	Módulo 1: Gestão de dado...
1030	Instrumentos de Desenvol...	Descubra quais são as at...	Módulo 1 - Fazendo as co...
1138	Primeiros passos com o M...	Neste curso, você conhe...	Como criar um documento ...
860	Compras sustentáveis e a...	Este curso tratará dos p...	Módulo 1: Sustentabilida...
676	Etiquetado Ambiental Tip...	El curso presenta el eti...	Módulo 1: Contextualizac...
840	Aplicação do Power BI pa...	Neste curso, você irá co...	Módulo 1: Introdução ao ...

A Escola Virtual de Governo (EVG) é uma plataforma de ensino a distância concebida e gerenciada pela Escola Nacional de Administração Pública (ENAP), vinculada ao Governo Federal brasileiro. Seu

objetivo principal é oferecer capacitação gratuita e de qualidade para servidores públicos e cidadãos em geral, promovendo o desenvolvimento de competências relacionadas à gestão pública, políticas sociais, tecnologia e outras áreas do conhecimento estratégicas para o setor público.


Isto posto, o dataset da EVG foi escolhido devido à riqueza de informações textuais (strings) em suas descrições e à quantidade relativamente baixa de entradas, o que possibilitou uma análise mais controlada e detalhada das recomendações, tornando-o adequado para desenvolvimento dos protótipos dos sistemas. Esse conjunto de dados foi extraído em 05/02/2025, diretamente do website da [EVG](#), no formato .csv. Ele contém 808 linhas e 13 colunas. A seguir, apresenta-se o dicionário completo de variáveis deste conjunto de dados:

- `id_curso`: Inteiro, Identificador único do curso na plataforma;
- `nome_curso`: *String*, Título da capacitação;
- `eixos_tematicos`: *String*, Sub-divisão do eixo ao qual a capacitação pertence;
- `competencias`: *String*, Competencias profissionais abrangidas pela capacitação;
- `certificador`: *String*, Ente responsável pela plaicação do curso;
- `conteudista`: *String*, Ente repsonsável pela elaboração dos conteúdos do curso;
- `carga_horaria`: Inteiro, Medida em horas. Carga do curso;
- `disponibilidade_dias`: Inteiro, Em quanto o tempo o curso pode ser feito. Medido em dias;
- `tipo_oferta`: *String*, Quem pode se inscrever na capacitação;
- `apresentacao`: *String*, Texto de apresentação do curso na plataforma;
- `publico_alvo`: *String*, Público-Alvo do curso;
- `conteudo_programatico`: *String*, Módulos e/ou capítulos que compõem a capacitação;
- `data_lancamento`: *Datetime*, Data em que a capacitação foi lançada no site;

É importante destacar que, das 13 features do conjunto de dados, apenas quatro foram utilizadas no desenvolvimento dos sistemas de recomendação, conforme apresentado na Tabela 4. A escolha dessas colunas se deve à necessidade de maximizar o uso de informações textuais disponíveis sobre cada curso, ampliando a base de conhecimento para a recomendação.

Dado que o sistema de recomendação é inteiramente baseado em dados textuais (strings), colunas numéricas como carga horária, disponibilidade do curso e data de lançamento foram descartadas, pois não agregam valor semântico relevante. Além disso, seu uso poderia introduzir ruídos desnecessários no modelo, prejudicando a qualidade das recomendações.

Entre as features textuais, foram selecionadas apenas aquelas que fornecem informações extensas e únicas sobre os cursos: `nome`, `apresentação` e `conteúdo programático`. Essas colunas contêm descrições detalhadas sobre os temas abordados, facilitando a identificação de palavras-chave e relações semânticas. A coluna `nome_curso`, por sua vez, foi utilizada apenas para identificar cada capacitação em conjunto com seu respectivo ID.



Por fim, visando aproximar-se ao máximo de um ambiente de produção real em que os dados de input utilizados na emissão das recomendações estarão armazenados em algum tipo de *database*, as informações dos cursos foram armazenadas em um banco de dados relacional PostgreSQL, tendo sido utilizada a linguagem python e as bibliotecas psycopg2-binary e pandas para conexão com o banco e ingestão dos dados.

b) Dataset “Livros”

O segundo e mais volumoso dataset utilizado para desenvolver e testar os sistemas de recomendação traz informações sobre mais de 29.000 obras literárias em Português-BR e seus campos foram extraídos por meio de consumo da API [Google Books](#) com uso da linguagem *Python* e das bibliotecas *Requests* e *Pandas*. Assim como o antecessor, este conjunto de dados traz consigo um campo textual em específico que possui grande relevância para o desenvolvimento deste projeto: A coluna de descrição do livro.

- Processo de obtenção dos dados:

Figura 7: Exemplo de função para extração da API Google Books com uso de Python

```
## Assim funciona a requisição da API do google books:
import requests

isbn = '9780545010221'
url = 'https://www.googleapis.com/books/v1/volumes?q=isbn:'+isbn
r = requests.get(url)
data = r.json()

#verificar se há itens retornados na resposta
if 'items' in data:
    # verificar se existem dados sobre o ISBN apontado
    if 'volumeInfo' in data['items'][0]:
        title = data['items'][0]['volumeInfo']['title']
        authors = data['items'][0]['volumeInfo']['authors']
        description = data['items'][0]['volumeInfo'].get('description', 'No description available')
        print("Title:", title)
        print("Authors:", authors)
        print("Description:", description)
    else:
        print("No volume information available")
else:
    print("No items found in the response")
```

Python

Title: Harry Potter and the Deathly Hallows
Authors: ['J. K. Rowling']
Description: "The final adventure in J.K. Rowling's phenomenal, best-selling Harry Potter book series"—Provided by publisher.

O processo para extração destes dados, no entanto, não fora tão simples e direto como aquele percorrido para o dataset anterior. A API do google books tem suas requisições baseadas no código ISBN que identifica cada livro, conforme demonstrado na Figura 7. Logo, foi necessário um trabalho preliminar de pesquisa para localizar uma lista com todos os ISBNs disponíveis, a qual fora encontrada e extraída em formato .parquet na plataforma [Hugging Face](#), para só então adaptar a função em python para iterar sobre esta lista e construir o dataframe com os dados consolidados.

Ocorre que mostrou-se inviável encaminhar requisições à API Google Books para todos os ISBNs listados, pois excederia o limite de requisições definida na documentação da ferramenta, além de gerar custos de processamento e armazenamento desnecessários, pois a maior parte daqueles dados não interessam ao projeto. Soma-se a isso o fato de que, conforme evidenciado no output da Figura 7, muitos dos identificadores presentes na lista retromencionada diziam respeito a obras em outras línguas que não a língua portuguesa, o que não era desejável uma vez que o cerne deste projeto é desenvolver sistemas de recomendação adaptados à língua

portuguesa.

Isto posto, a estratégia utilizada para solucionar os problemas apontados anteriormente foi utilizar a biblioteca *langdetect*, desenvolvida pelo Google originalmente em java e adaptada em sua integralidade para python, a qual detecta automaticamente, baseando-se no título textual do livro associado ao seu código ISBN, a linguagem a qual aquela obra dizia respeito. Após a sua aplicação, a lista de ISBNs que anteriormente possuía 28 milhões de entradas, passou a conter pouco mais de 100.000 entradas.

Tabela 5: Amostra de 10 linhas da tabela de livros

isbn	title	authors	description
9788500001710	177 maneiras de enlouq...	Margot Saint-Loup	177 dicas de Margot Sa...
9788500002533	Da cabeça aos pés	Marilda Castanha	'Da cabeça aos pés' é ...
9788500002540	Fada fofa, onça-fada!	Sylvia Orthof	Esta é mais uma divert...
9788500003042	Homem mais rico da Bab...	George Samuel Clason	Traz orientações atrav...
9788500005626	Pai, Me Compre Um Amigo?	Pedro Bloch	Esta é a história de B...
9788500005763	Memórias de um sargent...	Manuel António de Almeida	A linguagem popular e ...
9788500006593	No Final Do Seculo	Nathan P. Gardels	Uma coleção de ensaios...
9788500007385	Memórias de Ramses o g...	CLAIRE LALOUETTE	'As Memórias de Ramsés...
9788500007453	Viagens de Marco Polo, As	Carlos Heitor Cony	O veneziano Marco Polo...
9788500007606	Que cara tem o Brasil?	Mônica Pimenta Velloso	Por que será que o ver...

Por conseguinte, os dados foram efetivamente extraídos do google books apenas para os ISBNs de interesse tendo sido transformados e limpos com uso da linguagem SQL e inseridos em uma nova tabela denominada “livros”, armazenada no mesmo database onde já estava a tabela de cursos, de modo a centralizar a fonte de dados e padronizar o uso do método “carrega_dados” para ambas as tabelas na classe Recommender, a qual será analisada ao fim desta secção. A Tabela 5 traz uma amostra da composição final da tabela de livros.

2.2.2 Pré-processamento

De seguida, procedeu-se à limpeza e transformação dos dados textuais, preparando-os para a vetorização e subsequente inserção nos modelos de recomendação. As minúcias técnicas utilizadas nesta etapa serão expostas na secção que versa sobre a construção das engines de recomendação. No entanto, de forma geral, a metodologia aplicada na limpeza e transformação da coluna `compilado_textual` incluiu diversas etapas

essenciais para garantir a qualidade e consistência dos dados antes da etapa de vetorização, dentre as quais destacam-se:

- Remoção de stopwords e extensão da lista pré-definida pela biblioteca nltk para abarcar casos específicos identificados durante o processo de exploração dos dados;
- Normalização para que todas as palavras estivessem em letras minúsculas;
- Remoção de acentuação gráfica;
- Remoção de sinais de pontuação e caracteres especiais;

Quanto aos demais métodos de pré-processamento comumente aplicados no âmbito do PLN, como a lematização e o stemming, é importante destacar que, durante os testes exploratórios, tais abordagens não se mostraram eficazes para os conjuntos de dados utilizados. Por esse motivo, optou-se por não incorporá-las nas funções de limpeza dos dados aplicadas às versões finais dos sistemas de recomendação desenvolvidos.

2.2.3 Compilação dos dados

Após a coleta e processamento dos conjuntos de dados, o próximo passo envolve os processos de extração e engenharia de features. No contexto deste projeto, isso significa selecionar as variáveis mais relevantes para o desenvolvimento dos sistemas de recomendação, além de criar uma nova variável, chamada “compilado_textual”, que reunirá as informações textuais mais importantes sobre cada item em ambos os conjuntos de dados.

A criação da feature “compilado_textual”, que centraliza as informações textuais essenciais para os sistemas de recomendação, envolveu a agregação das seguintes colunas² para cada conjunto de dados:

- **Cursos:** nome, apresentação e conteúdo programático;
- **Livros:** título e descrição;

2.2.4 Medida de Semelhança: A similaridade do Cosseno e porquê utilizá-la.

Optou-se por adotar no âmbito do presente projeto a similaridade do cosseno como métrica objetiva de aferição da semelhança entre os vetores dos itens, haja vista sua ampla utilização no campo do PLN por ser indiferente à magnitude dos vetores, com foco na comparação da direção destes.

Este parâmetro de similaridade mede o ângulo entre dois vetores em um espaço multidimensional, indicando o grau de similaridade entre eles, ou seja, o quão próximos eles são semanticamente. Esta comparação é fundamental para que, dado determinado item, determine-se quais são os mais próximos a ele e, portanto, mais relevantes, os quais deverão ser recomendados ao usuário.

²A descrição de cada coluna consta no dicionário de variáveis da secção Extração dos dados e datasets utilizados

Considerando os vetores associados às palavras z e w , a similaridade do cosseno entre elas pode ser percebida como:

$$\text{cos-similarity}(z, w) = \frac{z \cdot w}{\|z\| \|w\|} = \cos(\alpha)$$

onde,

- α é o ângulo entre as palavras z e w .

Ressalta-se ainda que o valor da similaridade do cosseno para o caso em tela varia entre 0 e 1, onde 1 indica identidade e 0 indica que os vetores são ortogonais(nenhuma relação de similaridade). Destaca-se ainda a natureza não-negativa dos vetores gerados no caso do TFIDF, por exemplo, onde são representadas frequências de palavras ponderadas, o que impede que a similaridade do cosseno seja menor que 0.

No que tange ao modelo de Sentence Embeddings(SBERT), apesar de ser possível que os vetores de embeddings assumam valores negativos, eles são projetados para captar apenas relações semânticas positivas. Aqui o cálculo da similaridade do cosseno entre esses embeddings é feito em um espaço onde a semântica é representada por proximidade e não por oposição. Ademais, nesta técnica de geração de embeddings os vetores extraídos também são normalizados, a exemplo do que ocorre com o TFIDF, para garantir que a magnitude dos vetores não terá influência sobre a similaridade.

Por conseguinte, a etapa de normalização dos vetores durante a extração destes também tem por escopo garantir que o ângulo entre eles nunca seja superior a 90° , de modo a manter o cosseno restrito ao intervalo entre $[0,1]$.

Figura 8: Heatmap das 10 primeiras entradas da matriz de similaridade do cosseno, que possui dimensão total de 807×807 para o dataset de cursos.



A Figura 8 traz um exemplo claro do que fora explicitado até aqui: Nela representa-se uma amostra de 10 entradas da matriz de similaridade do cosseno gerada ao comparar-se os vetores de todos os itens entre si, matriz esta que constitui o ponto central dos sistemas de recomendação ora propostos. É possível observar que determinado item possui identidade (similaridade cosseno = 1) consigo mesmo, e que os valores das similaridade são menores à medida que os comparamos com os demais itens da matriz.

2.2.4.1 Outras medidas de similaridade

Embora a similaridade do cosseno tenha sido a métrica adotada no âmbito deste trabalho, outras medidas de distância, como a distância Euclidiana e a distância de Manhattan, são frequentemente utilizadas em tarefas de análise de dados. No entanto, essas métricas podem apresentar limitações significativas no presente contexto de sistemas de recomendação baseados em **vetores de alta dimensionalidade**.

No caso da matriz TFIDF gerada pelo primeiro sistema de recomendação, estamos a lidar com uma matriz esparsa, conforme ilustrado na amostra da Tabela 3. Para o conjunto de dados menor (cursos), a matriz possui cerca de 7.000 dimensões, enquanto para o conjunto de dados maior (livros), ela ultrapassa as 100.000 dimensões. Isto ocorre porque cada palavra única no vocabulário se transforma numa coluna na

matriz. Já para as recomendações baseadas em vectores de embeddings densos gerados pelo modelo SBERT (paraphrase-multilingual-MiniLM-L12-v2), o espaço vetorial é restrito a 384 dimensões. Embora isto seja consideravelmente menor que o espaço TFIDF, ainda pode ser excessivo para métricas que consideram a distância absoluta entre os pontos, em vez de se focarem no ângulo e na direcção entre os vectores.

Isto posto, resta claro que um dos principais desafios no presente contexto diz respeito ao enfrentamento da alta dimensionalidade do espaço vetorial, fenómeno que afeta métricas sensíveis à magnitude dos vectores, como a Euclidiana, mas tende a apresentar impacto reduzido na similaridade do cosseno, que se concentra na orientação relativa dos vectores, conforme discutido por Aggarwal et al. (2001).

Por fim, embora a similaridade do cosseno tenha sido escolhida para extracção dos resultados deste trabalho, serão apresentados, na secção seguinte (“Apresentação dos resultados”), testes comparativos que extraem recomendações para o mesmo item base, utilizando a similaridade do cosseno, a distância Euclidiana e a distância de Manhattan. O objectivo é analisar as principais diferenças nos resultados gerados por estas diferentes abordagens.

2.2.5 Construção das engines de recomendação: Desenvolvimento das Classes `TfidfRecommender` e `SBERTRecommender`

Após a obtenção e transformação dos dados, passou-se à construção efetiva dos sistemas de recomendação. Optou-se mais uma vez pelo uso da linguagem de programação python e do paradigma de orientação a objetos para facilitar a manutenção, organização e reutilização dos códigos fonte dos sistemas de recomendação. Foram implementadas duas classes distintas – `TfidfRecommender` e `SBERTRecommender` –, cada uma encapsulando o fluxo completo de seu respectivo algoritmo.

A partir deste momento, cada sistema de recomendação será analisado separadamente, dado que a natureza dos métodos utilizados para a extração e representação dos vectores difere significativamente entre eles. Inicialmente, será apresentada a implementação do modelo baseado em TF-IDF, uma abordagem mais tradicional e rudimentar para recomendação textual. Em seguida, será detalhada a construção do modelo baseado em SBERT e *transformers*, que utiliza representações semânticas mais avançadas para capturar significados contextuais nos textos.

- **`TfidfRecommender`**

Figura 9: Construtor da Classe TfidfRecommender

```
import pandas as pd
import re
import nltk
from sqlalchemy.engine import create_engine
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
from tqdm import tqdm

class TfidfRecommender:
    def __init__(self, DB_USER,
                 DB_PASS,
                 DB_NAME,
                 DB_HOST, emb_cols, id_col: str, item_name_col: str, stopwords_extra=None):
        self.DB_USER = DB_USER
        self.DB_PASS = DB_PASS
        self.DB_HOST = DB_HOST
        self.DB_NAME = DB_NAME
        self.engine = create_engine(f'postgresql://{self.DB_USER}:{self.DB_PASS}@{self.DB_HOST}:5432/{self.DB_NAME}')

        #cols que serao usadas p/ extrair texto p/ vetores de embeddings
        self.emb_cols = emb_cols
        self.id_col = id_col
        self.item_name_col = item_name_col
        self.stop_words = set(stopwords.words('portuguese'))

        # add stopwords personalizadas se fornecidas como argumento
        if stopwords_extra is not None:
            self.stop_words.update(stopwords_extra)

        #Inicialização do Tfidf
        self.tfidf_vectorizer = TfidfVectorizer()

        # Dados e manipulação dos vetores
        self.df_text = None
        self.embeddings = None
        self.cosine_sim = None
        self.indice = None
        self.tf_matrix = None
```

- Bibliotecas e versão do Python utilizadas:

- **Python 3.12.3**

- Pandas: Manipulação e estruturação dos dados em DataFrames.
 - re: Expressões regulares para limpeza e pré-processamento de texto.
 - nltk: Utilizado para processamento de linguagem natural, especialmente remoção de stopwords no caso em tela.
 - Sqlalchemy: Criação de conexão com banco de dados PostgreSQL para consulta dos dados.
 - Sklearn: Inicialização do tfidf, conversão de textos em vetores numéricos e cálculo da similaridade do cosseno via função linear_kernel.

- Parâmetros ao instanciar a classe:

- **DB_USER, DB_PASS, DB_NAME, DB_HOST:** Dados relativos ao banco de dados onde as informações relevantes para desenvolvimento do modelo encontram-se.
 - **emb_cols:** Colunas textuais que serão utilizadas como fonte de informação para o modelo. Este é o parâmetro mais importante, vez que aqui o operador deve apontar aquelas features de interesse no dataset que serão compiladas e servirão de base para extração dos vetores de representação textual.
 - **id_col:** Coluna na base de dados que contém a informação a respeito do identificador único de

cada item a ser processado e recomendado.

- **item_name_col:** Coluna que contém o título/nome do item a ser processado e recomendado adiante. Importante pois seu texto será adicionado às informações das colunas de emb_cols com vistas a maximizar os dados textuais sobre cada um dos itens.
- **stopwords_extra:** Faculta ao operador do sistema de recomendação a opção de adicionar ainda mais stopwords à etapa de limpeza textual que façam sentido para aquele domínio específico, as quais serão adicionadas às stopwords previamente definidas pela biblioteca nltk e aplicadas à etapa de limpeza textual.

- Método “carrega_dados”:

- Argumento(s) necessários: Nome da tabela no banco de dados que será utilizada como fonte de extração de texto para o modelo;
- Este método é responsável pela abertura do banco de dados relacional e carregamento das colunas textuais de interesse para o modelo. Utiliza-se para tanto os dados das features inseridos pelo operador ao instanciar a classe TfIdfRecommender(emb_cols + item_name_col).

- Método “_limpar_e_compilar_texto”:

Considerando o fragmento original do texto descritivo do livro “As Mil e Uma Noites”, observa-se um exemplo da aplicação do método “_limpar_e_compilar_texto” aos dados ora em comento:

“O livro apresenta as fábulas das ‘Mil e uma Noites’, com seu colorido oriental”

Tabela 6: Aplicação dos estágios de pré-processamento sobre a frase exemplificativa.

Etapa de Pré-Processamento	Resultado
Frase Original	O livro apresenta as fábulas das 'Mil e uma Noites', com seu colorido oriental
Minúsculas	o livro apresenta as fábulas das 'mil e uma noites', com seu colorido oriental
Remoção de Acentos	o livro apresenta as fabulas das 'mil e uma noites', com seu colorido oriental
Remoção de Stopwords	livro apresenta fabulas 'mil e uma noites' colorido oriental
Remoção de Pontuação Extra	livro apresenta fabulas mil e uma noites colorido oriental

- Executa a etapa de limpeza e preparação dos dados textuais das colunas que serão vetorizadas. Por

se tratar do sistema de recomendação baseado em tfidf, o qual requer minuciosa etapa de limpeza dos dados para evitar-se redundâncias e ruídos que sobrecarreguem o sistema ao estender em demasia o vocabulário (bag-of-words*), aplicam-se aos dados as seguintes transformações: Remoção de acentos, sinais de pontuação e caracteres especiais; Alteração para que todas as palavras estejam em letras minúsculas; Remoção das stopwords pré-definidas pela biblioteca nltk + stopwords definidas pelo operador em stopwords_extra, se houverem; Junção do texto de todas as colunas textuais apontadas pelo operador em uma coluna chamada “compilado textual”, a qual trará o texto do nome do item + o texto das demais colunas referentes ao texto de apresentação, eventual conteúdo programático e descrição. Esta coluna será utilizada como base para extração dos vetores que descrevem aquele item na matriz tfidf. Ressalta-se ainda que a nova variável criada prescinde de qualquer tratamento por já ter sido formada pela agregação de colunas limpas e transformadas.

- Método “gerar_vetores”:
- Recebe os dados da coluna “compilado_textual” limpos e compilados e aplica sobre eles o método tfidf_vectorizer da biblioteca scikit-learn, de modo a transformá-los nos vetores de representação textual numéricos que serão utilizados como base para a etapa de aplicação da similaridade do cosseno e posterior extração das recomendações;
- Calcula a matriz TF-IDF, onde cada linha representa um item (livro ou curso) e cada coluna representa um termo do vocabulário, atribuindo pesos conforme a importância relativa do termo no conjunto de dados. Isso permite que textos com palavras frequentes, mas pouco discriminativas, tenham menos peso, enquanto termos mais raros e relevantes recebam maior influência na representação vetorial.
- Utiliza a técnica de linear kernel para calcular a similaridade do cosseno entre os vetores da matriz TF-IDF. Esse cálculo mede o grau de proximidade entre os itens, atribuindo valores entre 0 e 1, onde 1 indica máxima similaridade e 0 indica ausência de relação.
- Armazena a matriz de similaridade do cosseno para que ela possa ser utilizada posteriormente no método de recomendação, garantindo que as consultas sejam feitas de forma eficiente sem necessidade de recalculá-las a cada requisição.
- Cria um índice (self.índice), que mapeia os identificadores únicos dos itens (id_col) aos seus respectivos índices na matriz de similaridade, permitindo consultas rápidas para recomendações futuras.
- Exibe informações sobre a geração dos embeddings, incluindo o shape da matriz TF-IDF e amostras das matrizes TF-IDF e de similaridade do cosseno, para fins de depuração e análise da qualidade das representações vetoriais.
- Método “recomendar”:

Figura 10: Demonstração da função que extrai as recomendações

```
def recomendar(self, id_item, top_n=3):
    """Recomenda itens similares com base no ID do item de entrada."""
    if self.cosine_sim is None:
        raise ValueError("A similaridade ainda não foi calculada. Certifique-se de que os embeddings foram gerados.")

    idx = self.indice[id_item]
    item_name = self.df_text.loc[self.df_text[self.id_col] == id_item, self.item_name_col].values[0].title()

    # Calcular scores de similaridade
    sim_score = list(enumerate(self.cosine_sim[idx]))
    sim_score = sorted(sim_score, key=lambda x: x[1], reverse=True)[1:top_n+1]
    sim_index = [i[0] for i in sim_score]

    recomendacoes = pd.DataFrame({
        'Item Recomendado': self.df_text[self.item_name_col].iloc[sim_index],
        'Similaridade': [score[1] for score in sim_score]
    }).reset_index(drop=True)

    print(f"As recomendações mais similares ao item '{item_name}' são:\n")
    return print(recomendacoes)
```

- Responsável por exibir as recomendações de itens mais similares ao item de entrada(input), utilizando a matriz de similaridade do cosseno previamente calculada. Ele recebe um ID de item e retorna os N itens mais similares, com base na proximidade vetorial entre os textos embutidos via TF-IDF.
- Em primeira medida, recebe um ID de item e consulta a matriz de similaridade do cosseno previamente gerada para encontrar os itens mais similares ao item de entrada.
- A seguir, a partir do ID do item fornecido pelo usuário, o método busca seu índice na matriz TF-IDF utilizando `self.indice[id_item]`. O nome do item original é recuperado da base de dados (`self.df_text`) para ser exibido na saída, facilitando a interpretação do usuário.
- Obtém a linha correspondente ao item de entrada na matriz de similaridade do cosseno, recuperando os valores de similaridade em relação a todos os outros itens.
- Ordena os itens de maneira decrescente quanto à similaridade, garantindo que os mais relevantes apareçam primeiro(remove a própria entrada da lista para evitar que o item recomende a si mesmo devido a similaridade máxima).
- Seleciona os

N

itens mais similares, conforme definido pelo usuário, garantindo que a recomendação seja ajustável às necessidades do sistema.

- Retorna um DataFrame contendo os IDs, nomes e scores de similaridade dos itens recomendados, de modo a facilitar a análise da qualidade daquilo que foi recomendado como sendo similar ao item de entrada.
- Exibe as recomendações de forma estruturada, apresentando o nome do item original e os itens sugeridos com seus respectivos scores.
- **SBERTRecommender**

Figura 11: Construtor da Classe SBERTRecommender

```
import pandas as pd
from sqlalchemy.engine import create_engine
from sentence_transformers import SentenceTransformer, util
import torch
import nltk
from nltk.corpus import stopwords
from tqdm import tqdm
import os
import warnings
from dotenv import load_dotenv

warnings.filterwarnings("ignore")
load_dotenv()

class SBERTRecommender:
    def __init__(self, DB_USER,
                  DB_PASS,
                  DB_NAME,
                  DB_HOST,
                  model_name: str,
                  emb_cols,
                  id_col: str,
                  item_name_col: str):
        self.DB_USER = DB_USER
        self.DB_PASS = DB_PASS
        self.DB_HOST = DB_HOST
        self.DB_NAME = DB_NAME
        self.engine = create_engine(f'postgresql://{self.DB_USER}:{self.DB_PASS}@{self.DB_HOST}:5432/{self.DB_NAME}')
        ##chama o modelo
        self.model = SentenceTransformer(model_name)
        ##chama cols que serao usadas p/ extrair texto p/ vetores de embeddings e stopwords em PT
        self.emb_cols = emb_cols
        self.id_col = id_col
        self.item_name_col = item_name_col
        self.stop_words = set(stopwords.words('portuguese'))
        self.embeddings = None
        self.df_text = None
```

- Bibliotecas e versão do Python utilizadas:
 - **Python 3.12.3**
 - Pandas: Manipulação e estruturação dos dados em DataFrames.
 - Sentence-Transformers: Geração de embeddings a partir de textos, utilizando modelos pré-treinados como SBERT.
 - nltk: Utilizado para processamento de linguagem natural, especialmente remoção de stopwords no caso em tela.
 - Sqlalchemy: Criação de conexão com banco de dados PostgreSQL para consulta dos dados.
 - Torch: Suporte ao processamento vetorial e operações matriciais com embeddings.
 - tqdm: Exibição de barra de progresso para acompanhar o processamento dos embeddings.
 - dotenv: Carregamento seguro de variáveis de ambiente sensíveis, como credenciais do banco de dados.
 - warnings: Supressão de mensagens de aviso durante a execução do código.
 - os: Manipulação de variáveis de ambiente e configurações do sistema.
- Parâmetros ao instanciar a classe:
 - **DB_USER, DB_PASS, DB_NAME, DB_HOST:** Dados relativos ao banco de dados onde as informações relevantes para desenvolvimento do modelo encontram-se.
 - **emb_cols:** Colunas textuais que serão utilizadas como fonte de informação para o modelo. Aqui

o operador deve apontar as features de interesse no dataset que serão compiladas e servirão de base para extração dos vetores de representação textual.

- **id_col:** Coluna na base de dados que contém a informação a respeito do identificador único de cada item a ser processado e recomendado.
- **item_name_col:** Coluna que contém o título/nome do item a ser processado e recomendado adiante. Importante pois seu texto será adicionado às informações das colunas de emb_cols com vistas a maximizar os dados textuais sobre cada um dos itens.
- **model_name:** Atalho para o modelo a ser utilizado para geração dos embeddings textuais. Permite flexibilidade na escolha de diferentes versões do modelo, a depender do domínio da aplicação. **No caso em tela utilizou-se o modelo “paraphrase-multilingual-MiniLM-L12-v2” da biblioteca Sentence Transformers**, entretanto outros modelos poderiam ter sido aplicados.

- **Método “carrega_dados”:**

- **Argumento(s) necessários:** Nome da tabela no banco de dados que será utilizada como fonte de extração de texto para o modelo.
- **Descrição:** Este método é responsável por estabelecer a conexão com o banco de dados PostgreSQL e carregar as colunas textuais de interesse definidas pelo operador ao instanciar a classe. Ele executa uma query SQL para extrair os dados das colunas especificadas em emb_cols e item_name_col, e define id_col como o identificador único de cada item.

- **Fluxo de execução:**

1. Conecta-se ao banco de dados utilizando as credenciais informadas na inicialização da classe.
2. Executa uma consulta SQL para selecionar as colunas de interesse na tabela especificada.
3. Armazena os dados extraídos em um DataFrame Pandas (df_text).
4. Remove valores nulos, para que apenas registros completos sejam utilizados na geração dos embeddings.

- **Método “limpa_dados”:**

- **Argumento(s) necessários:** DataFrame contendo as colunas textuais extraídas do banco de dados.

- **Descrição:** Método responsável pela limpeza e pré-processamento dos textos antes da geração dos embeddings densos pelo modelo. Aplica técnicas padronizadas de tratamento textual, removendo ruídos e normalizando os dados para melhorar a qualidade das representações geradas. Destaca-se o fato de ser uma limpeza menos complexa que aquela realizada no sistema de recomendação, haja vista a menor sensibilidade às nuances de padronização de vocabulário neste sistema.

- **Fluxo de execução:**

1. Converte todos os textos para letras minúsculas para garantir uniformidade.
2. Elimina espaços em demasia.
3. Tokeniza os textos.
4. Remove stopwords padrão da biblioteca NLTK; Diferente do sistema anterior, aqui não serão definidas stopwords além das pré-definidas pela biblioteca em comento;
5. Agrupa o texto de todas as colunas textuais apontadas pelo operador em uma coluna chamada “compilado textual”, a qual trará o texto do nome do item + o texto das demais colunas que serão utilizadas para extração dos vetores textuais(emb_cols). Ressalta-se ainda que a nova variável criada prescinde de qualquer tratamento por já ter sido formada pela agregação de colunas limpas e transformadas.

- **Método “embeddings_extract”:**

- **Argumento(s) necessários:**

- * `batch_size` (padrão: 64): Define o lote de sentenças a serem processadas simultaneamente para otimização do uso de memória.

- **Descrição:**

- * Este método é essencial para o funcionamento do sistema, pois é responsável pela geração dos embeddings (vetores numéricos densos) a partir dos textos processados na etapa de anterior. Utiliza-se um modelo de deep learning para converter os textos em embeddings densos, de modo a permitir cálculos de similaridade eficientes.

– **Fluxo de execução:**

1. Inicialização da coluna 'embeddings' no DataFrame, preparando-a para armazenar os vetores.
2. Divide os textos em lotes menores (`batch_size`) para evitar sobrecarga de memória e melhora a eficiência do processamento.
3. Para cada lote:
 - * Extrai os textos da coluna 'compilado_textual', garantindo que existam dados para processar.
 - * Usa o modelo de embeddings apontado pelo usuário ao instanciar a classe para efetivamente transformar os textos em vetores numéricos.
 - * Converte os embeddings gerados para listas e os armazena na coluna apropriada.
4. Empilha os vetores de embeddings resultantes em uma matriz 2D usando `torch.stack`, consolidando-os em um formato adequado para cálculos de similaridade.

– **Tratamento de erros:**

- * Se um erro ocorrer durante a geração dos embeddings, o processamento é interrompido e uma mensagem de erro é exibida.
- * Caso todos os embeddings sejam `None`, uma exceção é levantada para alertar sobre possíveis problemas nos dados de entrada.

– **Método “recomendar_itens”:**

Figura 12: Função responsável pela extração das recomendações

```
def recomendar_itens(self, id_item, top_n: int = 3):
    """Recomenda itens similares com base no ID do item de entrada."""
    id_item = str(id_item)

    if id_item not in self.df_text[self.id_col].values:
        print(f"ID {id_item} não encontrado na base de dados.")
        return None

    df_reset = self.df_text.reset_index()
    indice = pd.Series(df_reset.index, index=df_reset[self.id_col].astype(str))
    idx = indice[id_item]

    sim_score = util.cos_sim(self.embeddings[idx], self.embeddings).squeeze()
    sim_score = sim_score.tolist()

    sim_score = sorted(enumerate(sim_score), key=lambda x: x[1], reverse=True)[1:top_n+1]
    sim_index = [i[0] for i in sim_score]

    recomendacao = pd.DataFrame({
        'Item Recomendado': self.df_text[self.item_name_col].iloc[sim_index],
        'Sim_Coss': [score[1] for score in sim_score]
    }).reset_index(drop=True)

    if recomendacao.empty:
        print("Nenhuma recomendação encontrada.")
    else:
        print(f"As recomendações mais similares ao item '{self.df_text.loc[idx, self.original_nome_item]}' são:\n")

    return recomendacao
```

– **Argumento(s) necessários:**


- * `id_item`: Identificador único do item para o qual as recomendações serão geradas. Aqui é atribuído manualmente, mas em um contexto de produção diz respeito, por exemplo, ao último item adquirido ou pesquisado pelo usuário.
- * `top_n` (padrão: 3): Número de recomendações mais similares a serem retornadas.

– **Descrição:**

- * Este método realiza efetivamente a recomendação de itens relevantes com base na matriz de similaridade do cosseno entre os embeddings previamente gerados. Trata-se da etapa final do fluxo de recomendação baseado em vetores densos.

– **Fluxo de execução:**

1. Verifica se `id_item` existe na base de dados. Caso contrário, exibe uma mensagem e interrompe a execução.
2. Calcula a similaridade do cosseno entre o embedding do item selecionado e todos os outros embeddings dos demais itens da base.
3. Ordena os itens pela maior similaridade e seleciona os `top_n` mais similares, excluindo-se



da relação o próprio item consultado, haja vista sua similaridade máxima(identidade).

4. Cria um DataFrame para exibição dos itens recomendados. Caso nenhuma recomendação seja encontrada, exibe uma mensagem apropriada.

– **Tratamento de erros:**

- * Se o `id_item` não for encontrado na base de dados relacional, a função interrompe a execução e exibe uma mensagem de erro.

2.2.6 Metodologia de avaliação das recomendações

3 Apresentação dos Resultados

INTRODUZIR COM UMA MSCLA ENTRE: “This work adopts an unsupervised approach to recommendation, as it does not require labeled user-item interactions or explicit relevance judgments. Instead, recommendations are generated by comparing item embeddings derived from textual descriptions, leveraging semantic similarity as the primary signal.”

E: “While traditional metrics like precision and recall require labeled datasets, this work evaluates recommendation quality through interpretative analysis, examining semantic coherence and contextual relevance. Case studies (Table 3) demonstrate that embeddings capture nuanced relationships (e.g., thematic parallels between historical novels and biographies), though limitations arise from polysemy (Section 4.2).”

INSERIR UMA PRIMEIRA PARTE COM TESTES DE DIFERENTES METRICAS DE SIMILARIDADE. DIZER QUE MANHATTAN NAO SE MOSTROU ADEQUADA E EUCLIDEAN, APESAR DE RESULTADOS PROXIMOS EM AMBOS OS RECOMENDERS, TORNA A INTERPRETAÇÃO E A EXPLICACAO DOS RESULTADOS MUITO MAIS COMPLEXA SE COMPARADA A SIMILARIDADE DO COSSENO. (LEMBRAR QUE VETORES TFIDF SAO NORMALIZADOS E POR ISSO EUCLIDEAN E COSINE PODEM TER SIDO TAO SEMELHANTES)

DIVIDIR EM 2: PRIMEIRO, RESULTADOS APÓS APLICAÇÃO AO DATASET DE CURSOS (MENOR VOLUME); SEGUNDO, RESULTADOS APOS APLICAÇÃO AO DATASET DE LIVROS(MAIS VOLUMOSO E COMPLEXO); DETALHAR MINUCIOSAMENTE A DIFERENÇA NA CASPTAÇÃO DE CONTEXTO, TEMPO DE PROCESSAMENTO PARA AMBAS AS ABORDAGENS ETC.



4 USE CASE:

CITAR RAPIDAMENTE APLICAÇÃO NA EXTRAÇÃO DE RECOMENDAÇÕES PARA PNDP.
RESSALTAR ECONOMIA DE RECURSOS E GANHO DE EFICIENCIA.

INSERIR CAPTURA DE TELA DO APP STREAMLIT DESENVOLVIDO E RESSALTAR A
FUNCIONALIDADE DE O GESTOR PODER ALTERAR A SIMILARIDADE DO COSSENO PARA
AUMENTAR OU DIMINUIR O RIGOR NA HORA DE RECOMENDAR



5 Conclusão e Investigação Futura

DIZER SOBRE USO DE VECTOR DATABASES PARA ARMAZENAR EMBEDDINGS E GANHAR TEMPO; INSERÇÃO DE TRESHOLD MINIMO DE SIMILARIDADE DO COSSENO

Bibliografia

- Aggarwal, C. C., Hinneburg, A., & Keim, D. A. (2001). On the Surprising Behavior of Distance Metrics in High Dimensional Space. *Proceedings of the 8th International Conference on Database Theory (ICDT)*, 420–434. https://doi.org/10.1007/3-540-44503-X_27
- Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. <https://arxiv.org/abs/1810.04805>
- Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1), 5–53. <https://doi.org/10.1145/963770.963772>
- Kamath, S. S., & Kanakaraj, M. (2015). Natural language processing-based e-news recommender system using information extraction and domain clustering. *International Journal of Image Mining*, 1(1), 112. <https://doi.org/10.1504/ijim.2015.070031>
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. <https://arxiv.org/abs/1301.3781>
- Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. J. (1990). Introduction to WordNet: An on-line lexical database. *International journal of lexicography*, 3(4), 235–244.
- Reimers, N., & Gurevych, I. (2019, novembro). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. <http://arxiv.org/abs/1908.10084>
- Ricci, F., Rokach, L., & Shapira, B. (2011). *Introduction to Recommender Systems Handbook* (F. Ricci, L. Rokach, & B. Shapira, Eds.). Springer. <https://doi.org/10.1007/978-0-387-85820-3>
- Schafer, J. B., Konstan, J. A., & Riedl, J. (2001). *Data Mining and Knowledge Discovery*, 5(1/2), 115–153. <https://doi.org/10.1023/a:1009804230409>
- Vaswani, A. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.