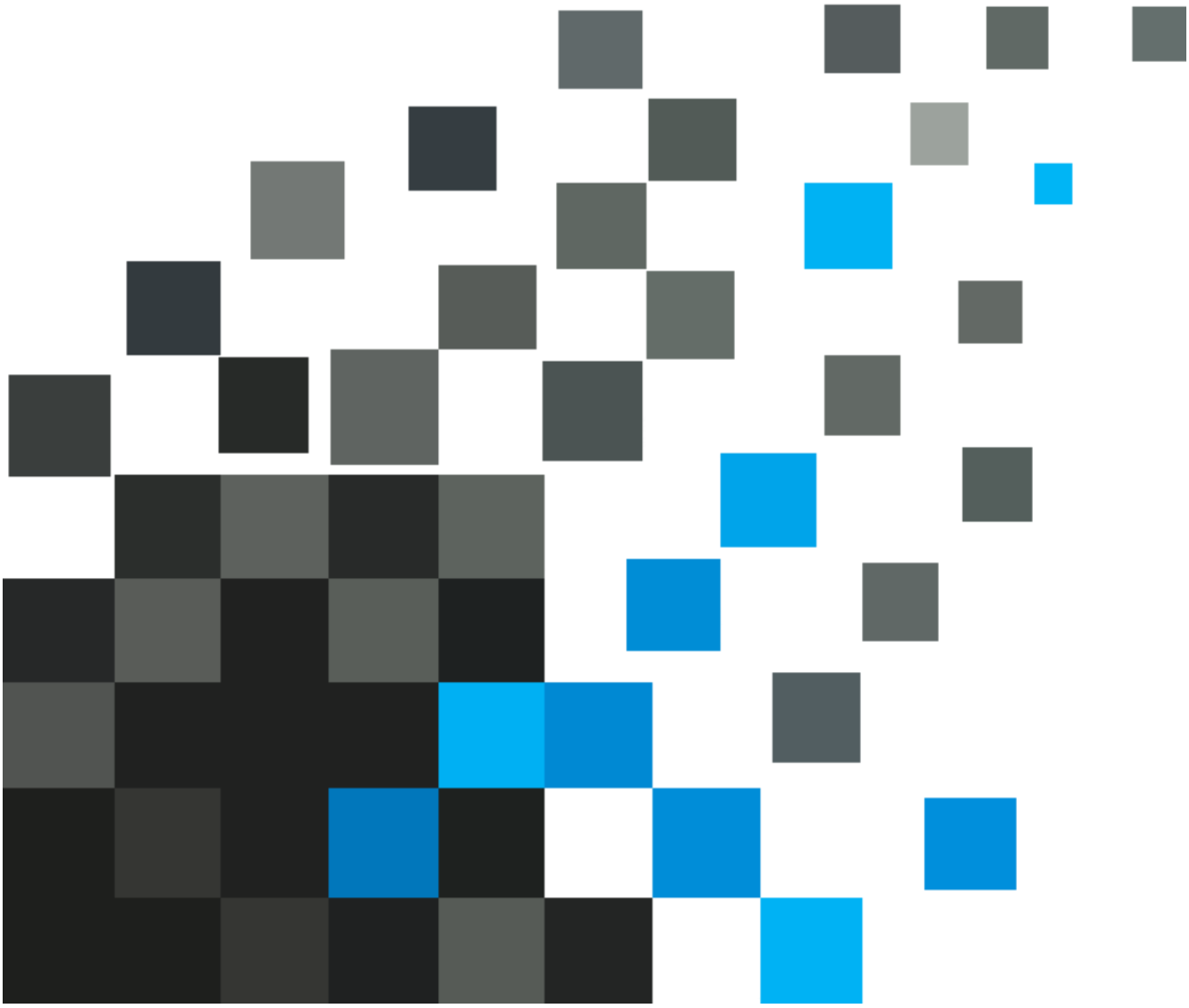
unesp

UNIVERSIDADE ESTADUAL PAULISTA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO



Re-ranqueamento contextual não supervisionado para recuperação de texto

Luis Felipe Antunes Venezian

INSTITUTO DE GEOCIÊNCIAS E CIÊNCIAS EXATAS

RIO CLARO

2026

UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"

Instituto de Geociências e Ciências Exatas
Câmpus de Rio Claro

LUIS FELIPE ANTUNES VENEZIAN

Re-ranqueamento contextual não supervisionado para recuperação de texto

Dissertação de Mestrado apresentada ao Instituto de Geociências e Ciências Exatas do Câmpus de Rio Claro, da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Daniel Carlos Guimarães Pedronette

Rio Claro - SP
2026

V458r

Venezian, Luis Felipe Antunes
Re-ranqueamento contextual não supervisionado para recuperação de texto / Luis Felipe Antunes Venezian. -- Rio Claro, 2026
114 p. : il., tabs.

Dissertação (mestrado) - Universidade Estadual Paulista (UNESP), Instituto de Geociências e Ciências Exatas, Rio Claro
Orientador: Daniel Carlos Guimarães Pedronette

1. Recuperação da Informação. 2. Recuperação da Informação de Texto. 3. Recuperação Contextual. 4. Similaridade de Texto para Recuperação. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Dados fornecidos pelo autor(a).

IMPACTO POTENCIAL DESTA PESQUISA

Esta pesquisa avalia métodos não supervisionados para melhorar buscas textuais sem exigir novos dados rotulados ou treinamento. Seu potencial científico e técnico está em apoiar sistemas de acesso à informação mais eficazes e reprodutíveis, alinhados ao Objetivo de Desenvolvimento Sustentável 9 (ODS 9): Indústria, Inovação e Infraestrutura.

POTENTIAL IMPACT OF THIS RESEARCH

This research evaluates unsupervised methods to improve text search without requiring new labeled data or training. Its scientific and technical potential lies in supporting more effective and reproducible information access systems, aligned with Sustainable Development Goal 9 (SDG 9): Industry, Innovation and Infrastructure.

UNIVERSIDADE ESTADUAL PAULISTA
"Júlio de Mesquita Filho"

Instituto de Geociências e Ciências Exatas
Câmpus de Rio Claro

LUIS FELIPE ANTUNES VENEZIAN

Re-ranqueamento contextual não supervisionado para recuperação de texto

Dissertação de Mestrado apresentada ao Instituto de Geociências e Ciências Exatas do Câmpus de Rio Claro, da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos para obtenção do título de Mestre em Ciência da Computação.

Comissão Examinadora

Prof. Dr. Daniel Carlos Guimarães Pedronette
IGCE / UNESP / Rio Claro (SP)

Prof. Dr. Lucas Pascotti Valem
ICMC / USP / São Carlos (SP)

Prof. Dr. Bruno Elias Penteado
FC / UNESP / Bauru (SP)

Conceito: Aprovado.

Rio Claro (SP), 2 de setembro de 2026.

Fruto dos privilégios que tenho: teto, saúde, comida à mesa, todo o apoio para que eu pudesse otimizar meu tempo, incontáveis finais de semana e feriados dedicados a esta jornada e um sacrifício compartilhado com meus pais, Rosemara e Márcio, meu irmão, Davi, e minha namorada, Lara.

Agradecimentos

Agradeço ao Programa de Pós-Graduação em Ciência da Computação da UNESP, aos excelentes professores que cruzaram meu caminho e à flexibilidade que distingue este programa.

Por fim, agradeço especialmente ao meu orientador, Professor Doutor Daniel Carlos Guimarães Pedronette, que, de maneira proativa, me inspirou a iniciar esta jornada. A vida e suas surpresas.

"Esquecer é uma necessidade. A vida é uma lousa, em que o destino, para escrever um novo caso, precisa de apagar o caso escrito."
(Machado de Assis)

Ainda que a vida me apague um pouco daqui, compete-me recuperar, sempre que o desejo surgir, o que ficou nestes capítulos. Em alguns transistores, em alguma nuvem.

Resumo

A recuperação de informação (RI) é um campo fundamental da ciência da computação que possibilita a busca eficiente por conhecimento em grandes coleções de documentos. Por meio de uma revisão aprofundada da literatura, são discutidos modelos tradicionais de recuperação, técnicas de avaliação e a evolução das representações textuais desde abordagens esparsas como TF-IDF e BM25 até representações densas baseadas em aprendizado profundo. Apesar desses avanços, o ranqueamento dos resultados ainda é tipicamente calculado de forma independente para cada documento, desconsiderando as relações de vizinhança existentes entre os itens recuperados. Como contribuição central, este estudo avalia técnicas de re-ranqueamento não supervisionado que exploram justamente essas relações contextuais entre documentos, uma abordagem com histórico de bons resultados em *benchmarks* de recuperação de imagens, mas ainda pouco investigada para texto. Para examinar sua transferência ao domínio textual, foi conduzido um experimento com nove conjuntos de dados públicos, quatro modelos de recuperação e três métodos de re-ranqueamento baseados em aprendizado de variedades, configurados em nove valores do parâmetro de vizinhança, totalizando 972 configurações executadas. Entre as 36 combinações de conjunto e recuperador, a melhor configuração selecionada aumentou a Precisão@20 em 35 casos, com ganho máximo de 19,8%; no caso restante, a variação foi inferior a 0,01%. Os resultados fornecem evidência de que o sinal contextual baseado em informações de ranqueamento pode complementar recuperadores esparsos e densos no domínio textual, sustentando a hipótese de transferência no escopo avaliado.

Palavras-chave: Recuperação de Informação; Re-ranqueamento Não Supervisionado; Aprendizado de Variedades; Recuperação Textual.

Abstract

Information Retrieval (IR) is a fundamental field in computer science that enables efficient knowledge discovery within large document collections. Through a comprehensive literature review, traditional retrieval models, evaluation techniques, and the evolution of textual representations are discussed, from sparse approaches such as TF-IDF and BM25 to dense representations based on deep learning. Despite these advances, result ranking is still typically computed independently for each document, disregarding the neighbourhood relationships among the retrieved items. As its central contribution, this study explores unsupervised re-ranking techniques that exploit precisely these contextual relationships among documents, an approach well established in image retrieval but still largely unexplored for text. To examine their transfer to the textual domain, an experiment was conducted with nine public datasets, four retrieval models, and three manifold-learning-based re-ranking methods, configured across nine neighbourhood parameter values, totalling 972 executed configurations. Among the 36 dataset–retriever combinations, the selected best configuration increased Precision@20 in 35 cases, with a maximum gain of 19.8%; in the remaining case, the variation was below 0.01%. The results provide evidence that rank-based contextual information can complement sparse and dense retrievers in the textual domain, supporting the transfer hypothesis within the evaluated scope.

Keywords: Information Retrieval; Unsupervised Re-ranking; Manifold Learning; Text Retrieval.

Title: Unsupervised contextual re-ranking for text retrieval.

Lista de ilustrações

Figura 1 – Conjuntos de recuperação da informação. 22
Figura 2 – Curva de precisão e revocação para Q1. 24
Figura 3 – Principais componentes de um sistema de recuperação da informação. . 30
Figura 4 – Modelo vetorial em três dimensões. 31
Figura 5 – Componentes de aprendizado de máquina em RI. 35
Figura 6 – Funcionamento dos modelos CBOW e SkipGram. 37
Figura 7 – Arquitetura biLSTM. 39
Figura 8 – Tarefas de MLM e NSP. 39
Figura 9 – Agregação de embeddings por *average pooling*. 42
Figura 10 – Realimentação de relevância e pseudo-relevância. 47
Figura 11 – Famílias *pointwise*, *pairwise* e *listwise* de Learning-to-Rank. 48
Figura 12 – Comparação entre *bi-encoder* e *cross-encoder*. 49
Figura 13 – Funcionamento da *Reciprocal Rank Fusion*. 50
Figura 14 – Interface de comparação entre listas ranqueadas e combinação de embeddings. 51
Figura 15 – Intuição do aprendizado de variedades. 52
Figura 16 – Estrutura comum dos métodos de re-ranqueamento contextual. 53
Figura 17 – Base fictícia e listas ranqueadas de exemplo. 54
Figura 18 – Núcleo do CPRR sobre a vizinhança de F1. 55
Figura 19 – Acúmulo de afinidades no CPRR. 55
Figura 20 – Núcleo do BFS-Tree sobre a base de exemplo. 57
Figura 21 – Núcleo do RDPAC e difusão na matriz de afinidades. 58
Figura 22 – Visão geral de uma execução experimental. 61
Figura 23 – Desenho combinatório da avaliação experimental. 62
Figura 24 – Etapa 1: indexação dos documentos. 63
Figura 25 – Etapa 2: consultas e listas ranqueadas iniciais. 64
Figura 26 – Etapa 3: conversão para o formato UDLF. 65
Figura 27 – Etapa 4: re-ranqueamento contextual. 66
Figura 28 – Etapa 5: qrels e cálculo das métricas. 66
Figura 29 – Arquitetura do *UDLF Text Espresso*. 68
Figura 30 – Desempenho global dos métodos em função de K. 84
Figura 31 – Relação entre densidade de relevância e melhor K. 85
Figura 32 – Projeção UMAP do Mental Health antes e depois do CPRR. 88
Figura 33 – Projeção UMAP do Huff Post antes e depois do CPRR. 89
Figura 34 – Re-ranqueamento contextual de uma consulta do BBC News. 91
Figura 35 – Re-ranqueamento contextual de uma consulta do WOS. 92

Lista de tabelas

Tabela 1 – Precisão, precisão interpolada e revocação para Q1. 24
Tabela 2 – Representação dos conjuntos para um termo de busca. 32
Tabela 3 – Visão geral dos modelos de embeddings. 40
Tabela 4 – Resumo dos conjuntos de dados dos experimentos. 70
Tabela 5 – Distribuição de documentos relevantes por consulta. 71
Tabela 6 – Resultados de re-ranqueamento para ArguAna. 76
Tabela 7 – Resultados de re-ranqueamento para SciFact. 76
Tabela 8 – Resultados de re-ranqueamento para NFCorpus. 77
Tabela 9 – Resultados de re-ranqueamento para SciDocs. 77
Tabela 10 – Resultados de re-ranqueamento para FiQA. 78
Tabela 11 – Resultados de re-ranqueamento para WOS. 78
Tabela 12 – Resultados de re-ranqueamento para BBC-News. 79
Tabela 13 – Resultados de re-ranqueamento para HuffPost. 79
Tabela 14 – Resultados de re-ranqueamento para Mental-Health. 80
Tabela 15 – Melhor configuração de re-ranqueamento no corte @20. 81
Tabela 16 – Sensibilidade dos métodos ao parâmetro K. 84
Tabela 17 – Comparação entre BM25, melhor modelo denso, *cross-encoder* e UDLF. 86
Tabela 18 – Comparação entre RRF e UDLF no SciDocs. 86

Lista de abreviaturas e siglas

ANN	*Artificial Neural Network*
AP	*Average Precision*
AQE	*Automatic Query Expansion*
BFS-Tree	*Breadth-First Search - Tree*
BEIR	*Benchmarking Information Retrieval*
BERT	*Bidirectional Encoder Representations for Transformers*
BGE	*BAAI General Embedding*
BiLM	*Bidirectional Language Model*
CBOW	*Continuous Bag of Words*
CPRR	*Cartesian Product of Ranking References*
CSLM	*Continuous Space Language Models*
DESM	*Dual Embedding Space Model*
DPR	*Dense Passage Retrieval*
ELMo	*Embeddings From Language Model*
FAISS	*Facebook AI Similarity Search*
GPT	*Generative Pre-trained Transformer*
LSA	*Latent Semantic Analysis*
LSTM	*Long Short-Term Memory*
MAP	*Mean Average Precision*
MiniLM	*Minimal Language Model*
MLM	*Masked Language Modeling*
MRR	*Mean Reciprocal Rank*
MS MARCO	*Microsoft Machine Reading Comprehension*

MTEB	*Massive Text Embedding Benchmark*
NLP	*Natural Language Proccesing*
NSP	*Next Sentence Prediction*
OOV	*Out of vocabulary*
PVDBOW	*Paragraph Vector Distributed Bag of Words*
PVDM	*Paragraph Vector Distributed Memory*
PPDB	*Paraphrase Database*
RAG	*Retrieval Augmented Generation*
RBO	*Rank-Biased Overlap*
RDPAC	*Rank-Based Diffusion Process with Assured Convergence*
RI	*Recuperação da Informação*
TREC	*Text Retrieval Conference*
UDLF	*Unsupervised Distance Learning Framework*
UMAP	*Uniform Manifold Approximation and Projection*
VSM	*Vector Space Model*

Lista de símbolos

$\|R\|$	Total de documentos relevantes no corpus.
$\|D\|$	Total de documentos recuperados pelo sistema.
$\|R \cap D\|$	Quantidade de documentos simultaneamente relevantes e recuperados.
$P(k)$	Precisão na k-ésima posição da lista de documentos recuperados.
r_k	Indicador binário que vale 1 se o documento k é relevante e 0 caso contrário.
$AP(q)$	Average Precision para uma consulta q.
P_R	Precisão R.
$RR(Q_i)$	Reciprocal Rank para uma consulta Q_i.
PR_i	Posição do primeiro documento relevante para a consulta Q_i.
$P(t\|R)$	Probabilidade de um termo t aparecer em documentos relevantes.
$P(\neg t\|\neg R)$	Probabilidade de um termo t não aparecer em documentos irrelevantes.
$\text{sim}(d, b)$	Similaridade entre um documento d e uma busca b.
N	Total de documentos no corpus.
n	Total de documentos que possuem o termo de busca.

Sumário

1 INTRODUÇÃO 17

2 FUNDAMENTAÇÃO TEÓRICA 21
2.1 Recuperação de Informações 21
2.2 Definição Formal 21
2.3 Avaliação da Recuperação 22

3 MODELOS DE REPRESENTAÇÃO DA INFORMAÇÃO TEXTUAL 29
3.1 Modelos Clássicos 29
3.1.1 Modelo Booleano 30
3.1.2 Modelo Vetorial 30
3.1.3 Modelo Probabilístico 31
3.1.4 BM25 - Best Matching 25 32
3.1.5 Abordagens de Aprendizado de Máquina 34
3.2 *Embeddings* 35
3.2.1 *Embeddings* Estáticos 36
3.2.2 *Embeddings* Contextualizados 38
3.3 Representação de Frases e Documentos 41
3.4 Combinação de Representações 44
3.5 Funções de Ranqueamento 45

4 RE-RANQUEAMENTO 46
4.1 Re-ranqueamento Clássico 46
4.2 Recuperação e Re-ranqueamento Neural 48
4.3 Fusão e Agregação de Listas Ranqueadas 49
4.4 Re-ranqueamento Contextual Não Supervisionado 51

5 METODOLOGIA PROPOSTA 60
5.1 Fluxo de Dados e Etapas 61
5.1.1 Etapa 1 - Indexação dos Documentos 62
5.1.2 Etapa 2 - Execução das Consultas 63
5.1.3 Etapa 3 - Preparação da Entrada para o UDLF 63
5.1.4 Etapa 4 - Configuração e Execução do Re-ranqueamento 64
5.1.5 Etapa 5 - Avaliação dos Resultados 65
5.2 O Framework UDLF Text Espresso 67

6 AVALIAÇÃO EXPERIMENTAL 70

6.1 Conjuntos de Dados . . . 70
6.1.1 Conjuntos de Dados BEIR . . . 70
6.1.2 Conjuntos de Dados Não-BEIR . . . 72
6.2 Configuração Experimental . . . 74
6.2.1 UDLF . . . 74
6.2.2 Definição do Parâmetro K . . . 75
6.3 Resultados Quantitativos . . . 75
6.3.1 Sensibilidade ao Parâmetro K . . . 83
6.3.2 Comparação com Cross-Encoder Supervisionado . . . 85
6.3.3 Comparação com Reciprocal Rank Fusion . . . 86
6.4 Análise Qualitativa . . . 87
6.5 Discussão e Síntese dos Resultados . . . 93
6.5.1 Síntese dos Achados . . . 93
6.5.2 Limitações da Avaliação . . . 94

7 CONCLUSÃO . . . 96
7.1 Síntese dos Resultados . . . 96
7.2 Contribuições . . . 97
7.3 Limitações . . . 97
7.4 Trabalhos Futuros . . . 98

REFERÊNCIAS . . . 100

1 Introdução

A recuperação de informação (RI) é uma área ampla da computação dedicada a permitir que os usuários encontrem documentos relevantes em uma coleção definida [76]. Em termos gerais, essa atividade compreende identificar, representar e organizar documentos para habilitar consultas e facilitar o acesso aos itens que atendem a uma necessidade de informação [68]. Embora os textos sejam seu objeto mais comum, os princípios da área se estendem a diferentes tipos de conteúdo. Das grandes coleções da Biblioteca de Alexandria aos índices dos mecanismos de busca modernos, permanece o objetivo de localizar informação relevante em meio a um conjunto documental [6].

Os sistemas de recuperação de informação executam busca com precisão em dados estruturados em conjunto com operadores lógicos, mas o principal desafio reside no conjunto de dados não estruturados, que predominam o conteúdo de informação produzida e requerem técnicas diferentes [24]. O advento da internet continua expandindo o volume de informação não estruturada, textos, imagens, vídeos e registros de servidores são exemplos de informação sem estrutura definida para busca exata, ainda nesse sentido, encontram-se os primeiros sinais de aumento de conteúdo gerado por inteligencia artificial [29, 145]. O cenário descrito constrói desafios para recuperação da informação, cuja complexidade é relativa ao tamanho e diversidade do conjunto de documentos em que se aplica.

O surgimento dos grandes modelos de linguagem LLMs também tem demandado crescente quantidade de integrações entre os sistemas de RI e *frameworks* de RAG (*Retrieval-Augmented Generation*) [25, 32, 70], permitindo que a recuperação de dados relevantes sirva de base para gerar respostas que contextualizam atividades em tempo de execução. Desse modo, os esforços nessa área são impulsionados pela oportunidade de impactar, ainda mais, a eficácia daqueles que precisam recuperar o documento correto em meio a um conjunto de informações preciosas que dividem espaço com outros itens não relevantes.

A evolução dos sistemas de recuperação textual possui uma longa trajetória. Ainda na década de 1960, foram desenvolvidos métodos para indexar documentos a partir dos termos neles contidos, entre os quais se destaca o modelo vetorial [40, 108]. Nesse modelo, documentos e consultas são representados por vetores cujos componentes indicam a presença ou a importância dos termos. A ponderação por frequência deu origem ao TF-IDF e forneceu parte da base estatística posteriormente incorporada ao BM25 [110, 101], que permanece uma referência para a comparação de abordagens modernas. Esses modelos são denominados esparsos porque a dimensionalidade de seus vetores acompanha o vocabulário do sistema.

Embora fundamentais, as representações esparsas enfrentam limitações diante de polissemia, sinonímia e variações lexicais. A representação textual evoluiu, então, de abordagens baseadas em contagem, como *Bag of Words*, para *embeddings* capazes de codificar relações semânticas e, mais recentemente, produzir representações dependentes do contexto linguístico [90]. Ao distinguir os sentidos assumidos por uma palavra em diferentes contextos, esses modelos oferecem aos sistemas de RI um sinal semântico que complementa a correspondência literal entre termos.

Nesse percurso, estudiosos buscaram aprimorar a representação vetorial de termos, passagens e documentos [77]. As redes neurais artificiais permitiram o desenvolvimento de representações densas (ou *dense embeddings*), impulsionadas pela arquitetura *Transformer* [132] e por modelos como o BERT [21]. Esses avanços alcançaram diversas tarefas de processamento de linguagem natural, entre elas a recuperação da informação.

Após uma consulta inicial, os resultados podem ser objeto de re-ranqueamento, tarefa que se concentra em refinar a ordenação dos documentos retornados por um sistema de RI, ou por mais de um sistema em casos de fusão [65]. Outras estratégias incluem a fusão de listas ranqueadas [16], que combina resultados de múltiplos sistemas, e o aprendizado de ranqueamento (*learning to rank*) [72], que treina modelos para otimizar diretamente métricas de ordenação. Dessa forma, os avanços em representações e técnicas de re-ranqueamento permitem aprimorar a recuperação em diferentes estágios de sua arquitetura.

Entre as abordagens mais proeminentes de re-ranqueamento, as técnicas baseadas em *cross-encoders* representam o estado da arte supervisionado: ao receber a consulta e o documento concatenados como entrada única de um *Transformer*, o modelo captura interações token a token[1] entre ambos, produzindo estimativas de relevância superiores [89]. Essa arquitetura, porém, impõe duas limitações práticas: como não é possível pré-computar representações independentes dos documentos, cada par consulta-documento exige uma inferência completa, tornando a avaliação exaustiva inviável em coleções de grande porte [100]; ademais, o treinamento requer exemplos rotulados de relevância, o que restringe a generalização a domínios com escassez de anotações [126]. Paralelamente, métodos não supervisionados baseados em aprendizado de variedades (*manifold learning*) e relações contextuais entre documentos têm apresentado resultados promissores em domínios visuais, como a recuperação de imagens [37, 92, 131, 130, 91], onde a estrutura local do espaço de representação é explorada para refinar listas ranqueadas sem necessidade de supervisão. Contudo, até o presente momento, tais técnicas permaneciam confinadas ao domínio visual, sem investigação sistemática de sua eficácia em recuperação textual, lacuna que o presente trabalho se propõe a preencher.

[1] Ao longo deste texto, o termo *token* pode ser lido, de forma aproximada, como equivalente a "palavra". A rigor, um *token* é a unidade mínima processada pelo modelo e nem sempre corresponde a uma palavra completa, podendo ser uma subpalavra, um sinal de pontuação ou um símbolo especial.

Nos recuperadores de primeiro estágio, a relevância de um documento é usualmente estimada por uma comparação direta com a consulta, como a similaridade de cosseno ou a distância euclidiana entre vetores, ou ainda por uma função lexical como o BM25. Embora eficazes, essas comparações tratam cada par consulta–documento de forma essencialmente independente e não incorporam explicitamente como os documentos se organizam em relação aos demais itens do corpus. Neste trabalho, o contexto não se refere apenas ao conteúdo textual de um documento, mas à sua posição na estrutura de vizinhança: cada documento é considerado também como consulta, e suas listas de vizinhos revelam relações recorrentes, vizinhanças compartilhadas e caminhos de afinidade entre documentos. Assim, uma proximidade isolada e possivelmente espúria pode ser enfraquecida, enquanto relações confirmadas repetidamente pela vizinhança do corpus são reforçadas. Os métodos CPRR, BFS-Tree e RDPAC exploram essa evidência estrutural sem exigir rótulos de relevância nem introduzir um novo estágio supervisionado de re-ranqueamento [129, 92, 37].

A hipótese central que motiva este esforço é a de que o re-ranqueamento contextual baseado em informações de ranqueamento, ao explorar relações recorrentes de vizinhança entre documentos em vez de depender somente de comparações diretas par a par, recupera sinais de relevância complementares aos modelos de primeiro estágio. Dessa forma, a abordagem pode corrigir ambiguidades locais da lista inicial sem depender de dados rotulados, sendo aplicável tanto a recuperadores esparsos quanto densos. Os resultados experimentais obtidos ao longo deste trabalho sustentam essa hipótese no escopo avaliado: uma avaliação envolvendo três métodos contextuais não supervisionados, nove bases de dados de cinco domínios e quatro modelos de recuperação, com 972 configurações executadas, identificou melhorias de Precisão@20 em 35 das 36 combinações quando selecionada a configuração mais favorável para cada cenário, alcançando ganho relativo de até 19,8%. A magnitude e a ocorrência dos ganhos variam entre métodos, métricas e coleções, mas o conjunto das evidências mostra que a estrutura local das listas ranqueadas pode oferecer informação útil e complementar na recuperação textual.

Diante do exposto, as principais contribuições deste trabalho são:

- a aplicação inédita de métodos de re-ranqueamento contextual baseados em aprendizado de variedades à recuperação textual, avaliando sua viabilidade em uma modalidade para a qual não foram originalmente projetados;
- uma comparação experimental em larga escala de três métodos não supervisionados, avaliados sobre nove conjuntos de dados de cinco domínios e quatro modelos de recuperação (um esparso e três densos), totalizando 972 experimentos de re-ranqueamento;
- evidência empírica de que o sinal contextual baseado em informações de ranqueamento pode complementar recuperadores esparsos e densos, com resultados favoráveis na

ampla maioria das combinações avaliadas sob a configuração selecionada;

- um arcabouço experimental reprodutível, de código aberto, que integra coleta de dados, recuperação esparsa e densa, re-ranqueamento e cálculo de métricas em um *pipeline* unificado, facilitando a replicação e a extensão dos experimentos.

Parte dos resultados desta dissertação foi consolidada no artigo *Context-Aware Text Re-Ranking through Unsupervised Manifold Learning*, aceito para publicação na Trilha Principal do 41º Simpósio Brasileiro de Banco de Dados (SBBD 2026) [133]. O artigo apresenta uma versão condensada da proposta e dos experimentos centrais, enquanto esta dissertação amplia esse material com a fundamentação teórica, a descrição detalhada do *UDLF Text Espresso*[2], o protocolo experimental completo, a análise de sensibilidade ao parâmetro K e comparações adicionais.

Além desta Introdução, o conteúdo deste estudo é organizado em seis capítulos:

- O Capítulo 2 discute a fundamentação teórica para este trabalho;
- O Capítulo 3 apresenta os principais modelos de representação da informação, com ênfase em embeddings e em sua aplicação à recuperação textual;
- O Capítulo 4 apresenta o re-ranqueamento e os trabalhos correlatos, abrangendo estratégias clássicas, neurais supervisionadas, de fusão de listas ranqueadas e contextuais não supervisionadas;
- O Capítulo 5 descreve a metodologia proposta e o pipeline experimental adotado;
- O Capítulo 6 apresenta a avaliação experimental e a análise dos resultados;
- O Capítulo 7 apresenta a conclusão, com contribuições, discussão e trabalhos futuros.

[2] https://github.com/unesp-text-retrieval/udlf-espresso

2 Fundamentação Teórica

Este capítulo apresenta os conceitos teóricos fundamentais para o entendimento dos métodos de recuperação de informação. A Seção 2.1 introduz a definição dos sistemas de RI, as principais aplicações da solução e seus métodos de avaliação.

2.1 Recuperação de Informações

A história dos sistemas de recuperação da informação precede a popularização da internet. Já na década de 1960, sistemas inteligentes de busca eram empregados comercialmente [112, 125], e seus mecanismos evoluíam em conjunto com a capacidade computacional de processamento e armazenamento [114]. Nesse mesmo período, foram introduzidos conceitos que permanecem relevantes, como a representação de consultas e documentos por vetores em um espaço multidimensional.

Avanços significativos nos sistemas de RI ocorreram a partir de modelos vetoriais espaciais e probabilísticos até chegarem ao aprendizado de máquina, em especial as técnicas de aprendizado profundo. Esses avanços foram complementados pelos esforços direcionados ao aprimoramento de redes neurais, que possibilitaram, por exemplo, o treinamento de modelos de ranqueamento cada vez mais inteligentes [38].

Os sistemas de Recuperação da Informação possuem uma vasta gama de aplicações, que vão desde sistemas de perguntas e respostas [55] até motores de busca populares, como o exemplo mencionado na introdução, além disso, a utilização de RI pode capacitar a coleta de documentos relevantes em diversas áreas, como no auxílio a diagnósticos médicos [116] e na aplicação adequada da legislação [18], quando empregados por juristas. Dado seu impacto, a introdução de ferramentas de RI é de grande interesse para múltiplas disciplinas. A ciência, em particular, também beneficia-se desse avanço ao utilizar buscadores específicos para encontrar artigos, livros e periódicos, impulsionando assim a pesquisa e o desenvolvimento do conhecimento.

2.2 Definição Formal

Um modelo de recuperação de informação é caracterizado como uma tupla de quatro elementos [6] $\langle D, Q, M, R(q_i, d_j)\rangle$, e pode ser definido como:

- D é o conjunto das representações lógicas dos documentos presentes na coleção, chamadas de *documentos*;

- Q é o conjunto das representações lógicas das necessidades de informação dos usuários, chamadas de consultas;
- M é um arcabouço que define a modelagem das representações dos documentos e consultas, além dos seus relacionamentos. Exemplos incluem conjuntos e relações booleanas, vetores e operações de álgebra linear, espaços amostrais e distribuições de probabilidade;
- $R(q_i, d_j)$ é uma função de ranqueamento que associa um número real à relação entre a consulta $q_i \in Q$ e o documento $d_j \in D$. Este valor define uma ordenação dos documentos em relação à consulta q_i.

Assim, o objetivo do sistema de recuperação de informação é, dado uma consulta $q_i \in Q$, retornar um subconjunto de D ordenado de acordo com a função $R(q_i, d_j)$. A Figura 1 ilustra os conjuntos definidos e a relação entre eles.

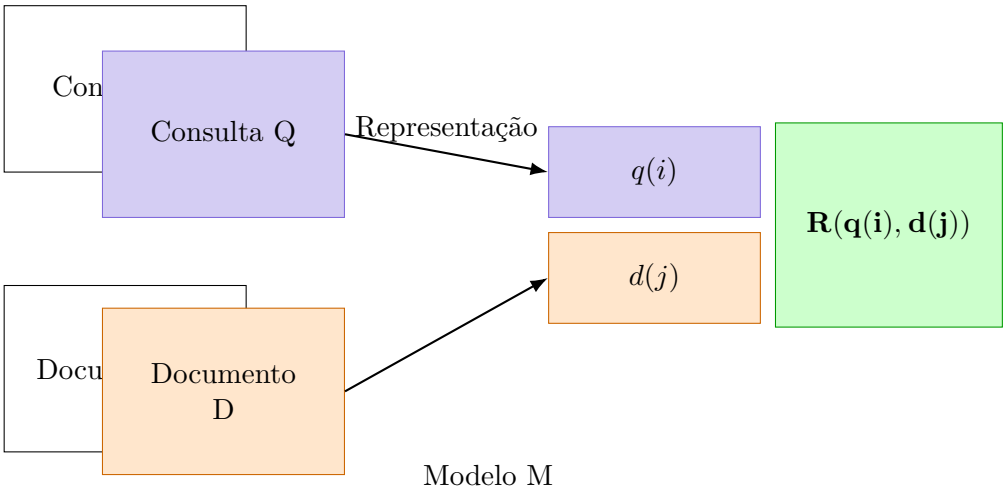


Figura 1 – Caracterização dos conjuntos presentes em sistemas de recuperação da informação.
Fonte: Elaboração própria.

2.3 Avaliação da Recuperação

Para que os sistemas de RI possam evoluir com rigor, são necessárias técnicas e protocolos para avaliar a qualidade das consultas nestes sistemas, apenas a partir de um método padrão entre duas soluções, podemos calcular uma métrica e comparar os resultados obtidos. Esta seção descreve algumas das heurísticas tradicionais para avaliar a qualidade dos sistemas de RI com base na experiência do usuário com relação a capacidade de recuperar documentos julgados relevantes para sua consulta. Existem outros fatores que podem estar relacionados com a qualidade de um sistema de RI, como a velocidade de execução, a disponibilidade e a interface de acesso, mas estes não serão apresentados.

Os conceitos de Precisão (*Precision*) e Revocação (*Recall*) consolidaram-se na avaliação de recuperação a partir dos experimentos de Cranfield, conduzidos no Reino Unido entre as décadas de 1950 e 1960. Esses estudos estabeleceram um protocolo controlado para comparar linguagens de indexação e sistemas de recuperação, tornando-se uma referência para a avaliação experimental da área [13, 135].

Precisão e Revocação

A Precisão quantifica a proporção de documentos relevantes entre os documentos recuperados pelo sistema, ou seja, reflete a capacidade do sistema em retornar apenas documentos que atendem às expectativas do usuário. Por outro lado, a Revocação avalia a capacidade do sistema em recuperar todos os documentos relevantes disponíveis, medindo a abrangência da recuperação [6].

Formalmente, essas métricas podem ser expressas através da interseção entre os conjuntos de documentos relevantes (R) e os documentos recuperados (D):

$$\text{Precisão} = \frac{|R \cap D|}{|D|} \tag{2.1}$$

onde $|R \cap D|$ denota a quantidade de documentos que são simultaneamente relevantes e recuperados, e $|D|$ é o total de documentos recuperados pelo sistema.

$$\text{Revocação} = \frac{|R \cap D|}{|R|} \tag{2.2}$$

onde $|R|$ representa o total de documentos relevantes existentes no corpus.

No contexto de um sistema de recuperação S, considere uma consulta *Q1*, para a qual há um conjunto de documentos relevantes previamente classificados, *R1 = {d2, d17, d20, d23, d37}*. Suponha que o sistema retorne a seguinte lista de resultados, onde os documentos relevantes estão destacados:

$$\{\mathbf{d2}, d5, \mathbf{d20}, d8, \mathbf{d37}, d88, \mathbf{d23}, d13, d16, \mathbf{d17}\}$$

Neste exemplo, calculamos a precisão e a revocação em diferentes posições da lista de resultados:

A interpolação da precisão é utilizada para suavizar as oscilações na série de precisão e fornecer uma representação mais clara do desempenho do sistema em diferentes pontos de revocação. O objetivo é garantir que a precisão interpolada em um determinado ponto seja sempre maior ou igual à precisão observada para qualquer nível de revocação superior.

Formalmente, a precisão interpolada em um ponto de revocação r é dada pela máxima precisão observada para qualquer revocação [76] $r' \geq r$:

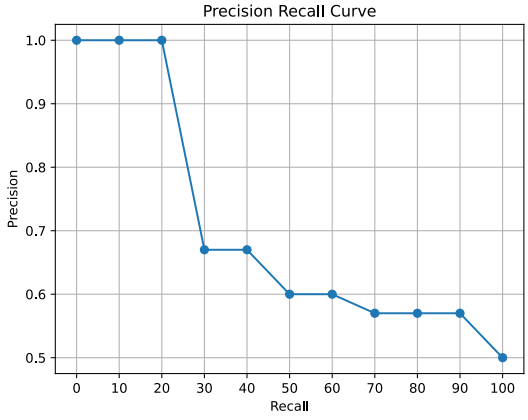
Documento	Precisão	Precisão Interpolada	Revocação
d2	1,00	1,00	20%
d5	0,50	1,00	20%
d20	0,67	0,67	40%
d8	0,50	0,67	40%
d37	0,60	0,60	60%
d88	0,50	0,60	60%
d23	0,57	0,57	80%
d13	0,50	0,57	80%
d16	0,44	0,57	80%
d17	0,50	0,50	100%

Tabela 1 – Precisão, Precisão Interpolada e Revocação para a consulta Q1.

$$\text{Precisão Interpolada}(r) = \max_{r' \geq r} \text{Precisão}(r') \tag{2.3}$$

Essa definição garante que a curva de precisão-revocação interpolada seja não crescente, facilitando comparações entre sistemas de recuperação da informação, isso significa que, em vez de projetarmos a precisão exata em cada ponto, utilizamos o valor máximo da precisão para todos os pontos seguintes. Os valores interpolados são utilizados para projetar um gráfico que mostra claramente como o sistema equilibra a precisão e a revocação à medida que mais documentos são recuperados. A Figura 2 demonstra a relação das métricas mencionadas para a consulta *Q1*.

Figura 2 – Curva de precisão e revocação da consulta *Q1*.
Fonte: Elaboração própria.



Esta apresentação considerou uma consulta única, todavia, os sistemas geralmente são avaliados a partir de várias consultas de teste. Nesse contexto, o valor médio da precisão para cada nível de revocação pode ser aplicado entre as consultas para obter uma avaliação final. O método *P@N* utiliza o valor médio para avaliar a métrica em uma posição N específica do resultado da consulta (i.e P@5, P@10, P@15), este parâmetro é ajustado de acordo cada necessidade, alguns sistemas implicam a necessidade de alta

precisão nos primeiros itens da busca, ou seja, altos valores em um N pequeno. Quando o usuário deseja ouvir uma música, ele usualmente deseja vê-la no topo dos resultados, enquanto o resto não o interessa, entretanto, uma pesquisa exploratória na área da saúde, por exemplo, exige a avaliação minuciosa em diversos documentos relevantes, de nada adiantaria a um médico buscar apenas um bom estudo de caso seguido de conteúdos irrelevantes, durante sua pesquisa o profissional precisa de mais informações relevantes para suportar uma determinada conduta.

Mean Average Precision (MAP)

A fim de obter um valor único para representar a pontuação da precisão em relação a revocação, a MAP é resultado da média da avaliação da *Average Precision (AP)* em cada consulta.

Average Precision

Para cada consulta, calcula-se a média das precisões em cada posição onde um documento relevante aparece na lista de resultados. Seja R o número total de documentos relevantes para uma consulta e $P(k)$ a precisão na k-ésima posição na lista de documentos recuperados. Se r_k é 1 quando o documento k é relevante e 0 caso contrário, a AP para uma consulta q é dada por:

$$AP(q) = \frac{1}{R} \sum_{k=1}^{n} P(k) \cdot r_k \tag{2.4}$$

Para exemplificar, considere a seguinte consulta que recupera cinco documentos, onde 3 documentos destacados são o total de documentos considerados relevantes para essa consulta, que também foram retornados na seguinte ordem:

$$\{\mathbf{d1}, d2, \mathbf{d3}, \mathbf{d4}, d5\}$$

As precisões nas posições relevantes são:

$$\{\mathbf{1.00}, 0.5, \mathbf{0.67}, \mathbf{0.75}, 0.6\}$$

A *Average Precision* considera apenas os documentos relevantes, portanto, no exemplo da consulta a equação será:

$$AP = \frac{1}{3}(1.0 + 0.67 + 0.75) = 0.81 \tag{2.5}$$

MAP

O *Mean Average Precision* é a média das *Average Precisions* calculadas para um conjunto de consultas Q. Formalmente definido como:

$$MAP = \frac{1}{|Q|} \sum_{q \in Q} AP(q) \tag{2.6}$$

Considere a consulta anterior com AP = 0.81 e uma nova consulta que resulte em um AP = 0.47, formulamos MAP da seguinte maneira:

$$MAP = \frac{1}{2}(0.81 + 0.47) = 0.64 \tag{2.7}$$

Precisão R

Assim como no MAP, o objetivo de calcular o valor da precisão R também consiste em obter um único valor para representar a lista ranqueada, contudo, neste caso o cálculo baseia-se na razão entre a interseção dos documentos recuperados D até a N-ésima posição da lista ranqueada e os documentos relevantes R com o total de documentos relevantes R [6]. Onde N é o tamanho do conjunto total de documentos relevantes para uma determinada consulta. Considere a consulta *Q1* demonstrada na seção de Precisão e Revocação, que possui 5 documentos considerados relevantes de acordo com conjunto *R1*, o valor da Precisão R para o cenário descrito se dá pela razão quantidade de documentos relevantes até a quinta posição e o total de documentos relevantes existentes no corpus, formulado como:

$$P_R = \frac{|R \cap D_{1:N}|}{|R|} = \frac{3}{5} = 0.6 \tag{2.8}$$

A Precisão R pode ser avaliada em consultas individuais, mas também pode ser sumarizada com o valor médio em um determinado conjunto de consultas, o valor da PR também é usualmente aplicado na construção de histogramas de precisão, os histogramas são calculados a partir da diferença dos valores da PR em cada uma das consultas efetivadas e podem ser utilizados em comparações entre modelos.

Mean Reciprocal Rank (MRR)

O MRR é uma métrica que favorece avaliações que tem como objetivo verificar se o sistema é capaz de fornecer resultados relevantes nas primeiras posições de cada conjunto resposta. Para calcular o *Reciprocal Rank*, considere uma consulta Q_i, um conjunto D_i com documentos recuperados para esta consulta e uma função PR que informa a primeira posição da lista ranqueada com informação relevante. Para calcular o *Reciprocal Rank* (RR), consideramos a seguinte expressão, onde PR_i é a posição do primeiro documento relevante para a consulta Q_i.

$$RR(Q_i) = \frac{1}{PR_i} \tag{2.9}$$

Esta métrica comumente utiliza um limite, k, no número de resultados considerados, o *Reciprocal Rank* é modificado da seguinte forma:

$$RR(Q_i) = \begin{cases} \frac{1}{PR_i} & \text{se } PR_i \leq k \\ 0 & \text{se } PR_i > k \end{cases} \tag{2.10}$$

Essa adaptação permite que o sistema corte para zero o *Reciprocal Rank* quando a primeira posição relevante está além da posição k, favorecendo resultados relevantes nas primeiras posições da lista ranqueada. O *Mean Reciprocal Rank* se dá pela média entre os resultados de RR aplicados as consultas de teste no sistema.

Outras Métricas de Avaliação

Além das métricas abordadas, diversos outros métodos de avaliação têm sido propostos para medir a qualidade dos sistemas de recuperação da informação sob diferentes paradigmas. Alguns desses métodos se concentram na parametrização da relação entre revocação e precisão, como a Métrica E, que combina essas duas métricas a partir de um parâmetro b que favorece a métrica para representar mais a precisão do conjunto quando $b \to 0$ e mais a revocação quando $b \to \infty$. Outras abordagens abordam limitações nas avaliações tradicionais, como é o caso do binarismo nas classificações de relevância, nesse contexto, a métrica *Discounted Cumulative Gain* (DCG), leva em consideração a posição dos documentos relevantes na lista de resultados, atribuindo maior peso a documentos recuperados nas primeiras posições [46]. Por fim, há também métodos que comparam diretamente listas ranqueadas produzidas por diferentes sistemas, como a métrica *Kendall's Tau*, que calcula o grau de concordância entre duas listas, ou o *Spearman's Rank Correlation*, que mede a correlação entre as posições dos documentos em duas listas distintas.

Essas métricas, assim como as apresentadas nesta seção, são essenciais para que os sistemas de recuperação possam ser continuamente aprimorados, garantindo resultados cada vez mais precisos e relevantes para os usuários.

Coleções de Referência

Para habilitar a avaliação consistente de um sistema de RI é necessário que existam coleções de documentos previamente mapeados para realização dos testes, essas coleções em conjunto com as métricas de avaliação mencionadas nas seções anteriores possibilitam a comparação entre um conjunto diverso de soluções propostas para recuperação de informação. Esse conjunto de documentos mapeados são conhecidos como coleções de referência, durante muito tempo a área de RI trabalhou com coleções pouco formais e de pequeno porte, em 1992 alguns órgãos do estado norte americano promoveram a conferência de recuperação de textos TREC-1 [39].

Text Retrieval Conferences (TREC)

A TREC foi fundada com intuito de estabelecer um padrão para avaliações consistentes de RI, a conferência ocorre anualmente e é dedicada a padronizar coleções que são previamente validadas por especialistas e disponibilizada os pesquisadores da área. De modo geral, o conteúdo das coleções é dividido em três partes: os documentos, os tópicos (i.e consultas, as requisições de informação solicitadas ao corpus) e um conjunto de documentos relevantes para cada tópico. As coleções contém documentos relacionados com diversas fontes de conhecimento e evoluem a cada nova conferência, atualizando-se de acordo com os desafios do campo.

Outras coleções de interesse foram desenvolvidas para casos de uso específicos, como a INEX [35], voltada à recuperação em dados XML; a coleção de notícias Reuters [69]; e a OHSUMED [43], da área médica. Mais recentemente, o BEIR reuniu coleções heterogêneas em um protocolo comum para avaliar a generalização de modelos de recuperação, com ênfase em cenários *zero-shot* [126].

3 Modelos de Representação da Informação Textual

A representação de palavras em um contexto de recuperação da informação é geralmente um objeto matemático associado a forma de vetores, por exemplo, por meio da representação *one-hot*. Inicialmente, os vetores representavam as palavras eram esparsos, deste modo, a dimensão da representação estava diretamente relacionada com o tamanho do vocabulário contido no corpus do sistema de recuperação, essas limitações motivaram a pesquisa de novos métodos para aprender representações de palavras a partir de grandes corpora em um espaço enxuto. Essa representação também é conhecida como *word embeddings* e junto com a área de NLP evoluiu para um contexto conhecido como representações densas, com dimensões fixas, tipicamente menores, treinados por algoritmos de aprendizado de máquina [8].

De modo geral, *word embeddings* podem ser gerados de inúmeras maneiras, figurar diferentes valores e representar semanticamente as palavras com melhor desempenho em um determinado contexto ou outro, de acordo com a qualidade do modelo que foi construído. Para a utilização dos *embeddings*, apesar de não trivial, é possível realizar um treinamento para construção dos próprios vetores, com base em um conjunto de dados proprietário, ou de maneira mais simplificada utilizar os modelos pré treinados que estão disponíveis na internet.

Neste capítulo, a seção 3.1 descreve os principais modelos de representação da informação e posiciona o papel do aprendizado de máquina em sistemas de RI. Em seguida, a seção 3.2 apresenta a evolução dos embeddings e o recorte adotado nesta dissertação para representação de frases e documentos, com foco nos modelos efetivamente avaliados nos experimentos.

3.1 Modelos Clássicos

Outra maneira de descrever brevemente um modelo de RI pode ser apresentá-lo como a combinação em formato de tríade entre a representação de documentos, a expressão de busca e a função de busca, que intermedeia a relação dos componentes anteriores. A eficiência do modelo está relacionada diretamente com a sinergia entre os três elementos, desde as primeiras representações de modelos de RI, as propostas que prevaleceram seguem esta mesma definição como ponto de partida [26]. A Figura 3 ilustra os principais componentes dos sistemas de RI.

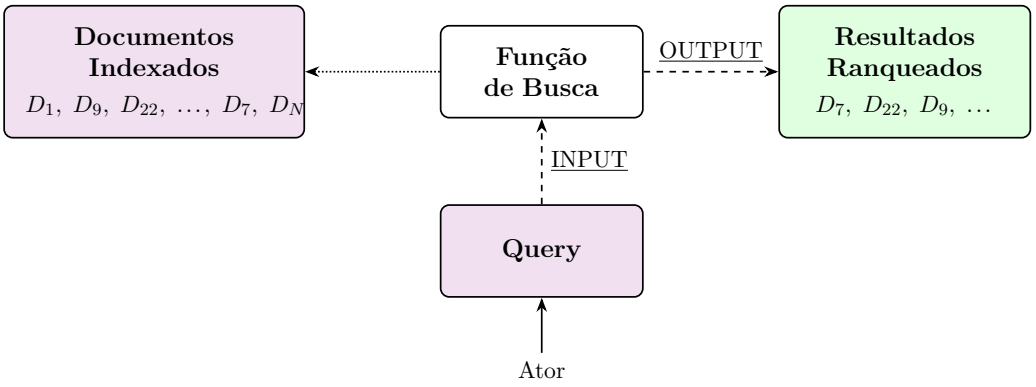


Figura 3 – Principais componentes dos sistemas de recuperação da informação.
Fonte: Elaboração própria.

3.1.1 Modelo Booleano

O modelo booleano é desenvolvido com base na Álgebra Booleana, um sistema binário, padrão utilizado em sistemas eletrônicos, com operações algébricas que resultam em 1 ou 0, *verdadeiro* ou *falso*. Este modelo representa os documentos a partir de um conjunto de termos indexados, onde as buscas são conectadas por combinações de operadores lógicos *AND*, *OR* e *NOT*, e retornam resultados que satisfazem estritamente as condições impostas na consulta, por exemplo, uma condição de busca do ***termo A AND termo B*** retornará somente documentos que contenham, ao menos, uma ocorrência de cada um termos pesquisados, esta abordagem também é conhecida como modelo da teoria dos conjuntos.

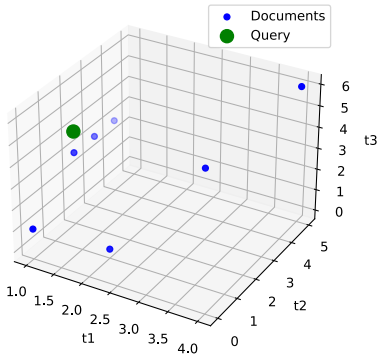
Algumas aplicações estenderam a busca com novos operadores de proximidade como *WITH*, para consulta de termos que aparecem em um mesmo parágrafo, *ADJ* e para termos que ocorrem seguidos uns aos outros [28]. Por fim, o operador *NEAR/N*, para consultar termos que estão a N termos distantes de outros termos.

3.1.2 Modelo Vetorial

O modelo Vetorial se baseia em um ambiente onde os documentos podem ser pesquisados a partir de uma consulta com termos, que podem estar esparsos e não restritos a uma declaração com operadores lógicos, a função de busca retorna um conjunto de documentos que contém os termos da consulta, cada um com um peso configurado de acordo com sua relevância, neste ambiente todo documento é representado como um vetor equilibrado com pesos, a consulta também é representada por um vetor, o sistema retorna uma lista de documentos ordenados pelos maiores valores de similaridade com a consulta do usuário [109]. Os pesos em um vetor são determinados considerando a frequência dos termos em um determinado documento ou no corpus (i.e. a coleção de todos os documentos). A Figura 4 demonstra a representação espacial do modelo vetorial projeta as dimensões *t1*, *t2*, e *t3*, onde cada uma representa, por exemplo, um termo do

corpus, cada ponto espacial de um documento possui uma distância em relação a consulta, o modelo espaço vetorial se baseia nesta distância para considerar a recuperação dos documentos.

Figura 4 – Representação gráfica do Modelo Vetorial com três dimensões.
Fonte: Elaboração própria.



Os pesos de cada termo indexado no modelo vetorial representam a sua importância, a atribuição é feita a partir da aplicação de dois métodos. O primeiro, *TF (Term Frequency)*, é o resultado do cálculo da frequência $f(t,d)$ do termo t em um determinado documento d. Quanto maior a frequência $f(t,d)$ em um documento, maior a capacidade do termo em representar aquele conteúdo. Contudo, o termo pode também aparecer com frequência em todos os demais documentos do corpus, o que deprecia sua relevância. Portanto, o segundo método, *IDF (Inverse Document Frequency)*, representa a razão entre o total de documentos N no corpus e a quantidade de documentos $n(t)$ que contêm o termo t, sendo assim, $\text{IDF}(t) = \frac{N}{n(t)}$. Para todo termo, seus pesos $w(t,d)$ são calculados a partir do produto desses métodos, ou seja, $\textit{TF-IDF}(t,d) = f(t,d) \cdot \frac{N}{n(t)} = w(t,d)$ [110].

3.1.3 Modelo Probabilístico

Neste modelo, os autores definem uma solução de RI baseada em um alicerce probabilístico, em um corpus os documentos são classificados em dois conjuntos em relação com uma determinada busca: relevantes e não relevantes [102]. Da mesma maneira, o resultado da busca é denominado como conjunto dos recuperados, que, idealmente, deve conter apenas conteúdo relevante.

Os documentos do corpus são representados no sistema por vetores binários, onde a presença ou a falta de um determinado termo de indexação é declarada respectivamente como 1 ou 0. O cálculo da similaridade entre uma expressão de busca e os documentos do corpus é realizada a partir da seguinte formulação:

$$\text{sim}(d,b) = \sum_{t \in b} \log \left(\frac{P(t|R) \cdot P(\neg t|\neg R)}{P(t|\neg R) \cdot P(\neg t|R)} \right) \tag{3.1}$$

Onde:

- $\text{sim}(d,b)$: Similaridade entre o documento d e a busca b,
- $t \in b$: Termos t presentes na busca b,
- $P(t|R)$: Probabilidade de o termo t aparecer em documentos relevantes,
- $P(\neg t|\neg R)$: Probabilidade de o termo t **não** aparecer em documentos irrelevantes,
- $P(t|\neg R)$: Probabilidade de o termo t aparecer em documentos irrelevantes,
- $P(\neg t|R)$: Probabilidade de o termo t **não** aparecer em documentos relevantes.

Em seguida, o cálculo para substituir as variáveis na função de similaridade será a contagem das ocorrências dos termos expressos na busca:

	Relevante	Não Relevante
Documento contém t	r	$n-r$
Documento não contém t	$R-r$	$N-R-n+r$

Tabela 2 – Representação dos conjuntos para um termo de busca.

Sendo **N** o total de documentos no corpus, **n** o total de documentos que possuem o termo de busca, **R** a quantidade total de documentos relevantes no corpus e **r** a quantidade de documentos relevantes que possuem o termo de busca, a formula anterior pode ser reescrita como:

$$\text{sim}(d,b) = \sum_{t \in b} \log\left(\frac{r \cdot (N-R-n+r)}{(n-r)\cdot(R-r)}\right) \tag{3.2}$$

No início, a instância de um sistema de RI probabilístico não tem, necessariamente, a informação sobre quais são os documentos relevantes do corpus; essa informação é, pouco a pouco, classificada pelo usuário do programa. Portanto, quanto não existirem documentos classificados como relevantes, o sistema utiliza uma função de similaridade simplificada, baseando-se na quantidade de ocorrências dos termos da expressão de busca:

$$\text{sim}(d,b) = \sum_{t \in b} \log\left(\frac{N-n}{n}\right) \tag{3.3}$$

3.1.4 BM25 - Best Matching 25

O BM25 [103] é um dos modelos mais conhecidos e amplamente utilizados em sistemas de recuperação da informação. Ele é baseado no modelo probabilístico de recuperação, calculando a relevância de um documento em relação a uma consulta através de uma combinação ponderada da frequência dos termos da consulta no documento e

características do corpus. Seu principal objetivo é atribuir maior relevância a documentos onde os termos da consulta aparecem de maneira mais significativa, considerando não apenas a frequência dos termos, mas também a raridade destes no corpus.

Uma das principais características do BM25 é sua capacidade de balancear a influência da frequência de termos através da função de saturação. Isso significa que um aumento na frequência de um termo em um documento não aumenta linearmente sua relevância, evitando um viés excessivo para documentos muito longos ou repetitivos. Essa saturação é controlada pelo parâmetro k, que regula a sensibilidade do modelo ao peso da frequência de termos.

Outro elemento central no BM25 é a normalização baseada no tamanho do documento. Esse ajuste é realizado pelo parâmetro b, que equilibra a penalização para documentos muito longos ou muito curtos. Valores maiores de b resultam em uma maior penalização para documentos extensos, enquanto valores menores minimizam essa influência, permitindo flexibilidade na adaptação do modelo a diferentes domínios ou coleções.

Apesar de ser um modelo simples e não depender de representações densas ou métodos de aprendizado profundo, o BM25 continua sendo uma referência na recuperação de informações devido à sua eficiência computacional e eficácia em benchmarks amplamente utilizados, como o BEIR. Ele é frequentemente usado como baseline em estudos que exploram métodos mais sofisticados, como embeddings de texto baseados em redes neurais, destacando-se por sua robustez em diversas aplicações práticas. A fórmula do modelo se dá pela seguinte expressão:

$$\text{score}(q,d) = \sum_{t \in q} \text{IDF}(t) \cdot \frac{f(t,d) \cdot (k_1 + 1)}{f(t,d) + k_1 \cdot \left(1 - b + b \cdot \frac{|d|}{\text{avgdl}}\right)} \tag{3.4}$$

Onde,

- $\text{score}(q,d)$: A pontuação de relevância entre a consulta q e o documento d, que determina o quão bem o documento corresponde à consulta.
- $\text{IDF}(t)$: *Inverse Document Frequency*
- $f(t,d)$: Frequência do termo t no documento d, ou seja, o número de vezes que t aparece no documento.
- k_1: Parâmetro de ajuste que controla a sensibilidade à frequência do termo $f(t,d)$.
  - Valores típicos de k_1 estão entre 1.2 e 2.0.
- b: Parâmetro que regula a penalização ou ajuste baseado no comprimento do documento.

  - Valores típicos de b variam entre 0.75 e 1.0.
  - Um $b = 0$ elimina a normalização pelo comprimento do documento.

- $|d|$: O comprimento do documento d, medido como o número total de palavras ou termos presentes no documento.
- avgdl: O comprimento médio dos documentos no corpus. Usado para normalizar o tamanho relativo de d em relação a outros documentos.

3.1.5 Abordagens de Aprendizado de Máquina

O aprendizado de máquina é o campo que estuda algoritmos capazes de aprender padrões a partir de dados, em vez de seguir regras explicitamente programadas. O aprendizado profundo é um subconjunto desse campo, baseado em redes neurais artificiais com múltiplas camadas, que ganhou força com o advento do *Big Data*, o aumento do poder computacional e novas técnicas de treinamento [118]. Ao empilhar camadas e unidades, uma rede profunda consegue representar funções de complexidade crescente, sendo particularmente eficaz em tarefas que mapeiam um vetor de entrada para um vetor de saída, desde que haja dados de treinamento e recursos computacionais suficientes [63]. Essas características tornaram tais técnicas especialmente úteis para dados não estruturados, como texto, imagem e fala.

Na recuperação da informação, o uso de redes neurais não é recente. Um trabalho pioneiro modelava conexões entre termos indexados e documentos, reforçando relações relevantes e inibindo as irrelevantes até a rede convergir para um padrão de recuperação satisfatório [81]; avaliações posteriores apontaram limitações, como o número fixo de ciclos de treinamento e a pouca exploração da capacidade de generalização das redes [119]. Algoritmos de aprendizado profundo são aplicados em diferentes pontos do sistema de RI: na modelagem de representações de consultas e documentos, na ordenação dos resultados e na expansão ou reformulação das consultas.

No ranqueamento, redes neurais podem ser empregadas para calibrar funções de ordenação, ajustando parâmetros como o k e o b do BM25 a partir de exemplos rotulados, em que a função de perda orienta o ajuste até que a ordenação atinja precisão satisfatória [72]. Na representação, as redes produzem os *word embeddings*, vetores em um espaço multidimensional que permitem comparar consultas e documentos por cálculo de similaridade espacial, indo além da correspondência exata de termos. Esse mecanismo pode, inclusive, ser combinado a métodos tradicionais como o TF-IDF para mitigar falsos positivos decorrentes de correlações espúrias entre termos [80].

Na expansão de consultas, a ideia é reformular a consulta original com termos relacionados ou sinônimos, ampliando a cobertura de documentos relevantes. Esse processo pode ser manual ou automático (*Automatic Query Expansion*, AQE), e soluções baseadas em

word embeddings com técnicas como *K-nearest neighbors* já superaram métodos tradicionais de sobreposição de termos [107]. Posteriormente, abordagens baseadas em modelos de linguagem reescrevem a consulta para aproximá-la da forma dos documentos, aumentando a similaridade entre ambos [141].

A Figura 5 sintetiza esses componentes, ilustrando os pontos do sistema de RI: representação, expansão de consultas e ranqueamento e re-ranqueamento em que o aprendizado de máquina pode ser empregado como trabalho adicional de otimização.

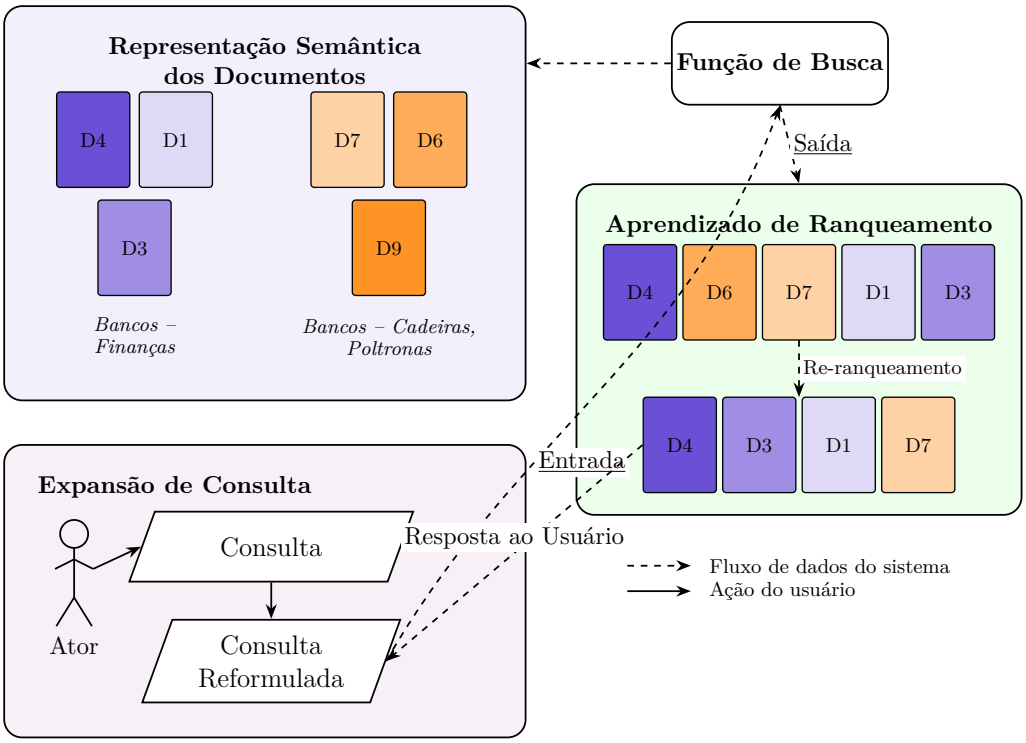


Figura 5 – Componentes de aplicação de aprendizado de máquina em sistemas de recuperação da informação.
Fonte: Elaboração própria.

Em todas essas frentes, as técnicas de aprendizado profundo integram o estado da arte da área. Entre elas, a representação textual é um componente central: é a partir dos *embeddings* que consultas e documentos passam a ser comparados por significado, fundamento tanto da recuperação quanto do re-ranqueamento investigado nesta dissertação. Por isso, a próxima seção dedica-se a detalhar como esses vetores são construídos, preparando a discussão sobre o re-ranqueamento contextual, foco deste trabalho.

3.2 *Embeddings*

No contexto de linguagem natural, embeddings são representações vetoriais que mapeiam unidades textuais para um espaço contínuo, de modo que proximidade geométrica reflita proximidade do significado também dessas palavras, frases ou documentos de texto,

esse efeito também é conhecido como similaridade semântica [52]. Em termos práticos, isso permite comparar consultas e documentos para além da coincidência literal de termos.

Para orientar a leitura, é útil separar três níveis de representação: palavra, frase e documento. Embeddings de palavra descrevem significado local de termos; embeddings de frase procuram captar uma proposição completa; embeddings de documento buscam representar o tema global e a intenção informacional de textos maiores. Em recuperação da informação, os dois últimos níveis tendem a ser mais úteis, pois consulta e documento são comparados como unidades semânticas completas.

Cabe destacar que o termo "contexto" nesta seção refere-se ao *contexto linguístico* (palavras que cercam um termo na sentença), distinto do *contexto estrutural* entre documentos explorado no re-ranqueamento contextual dos capítulos seguintes. A apresentação a seguir percorre a evolução dos embeddings em dois grandes movimentos: os modelos *estáticos* e os *contextualizados*. Ao final, sintetiza essa trajetória em uma visão geral comparativa.

3.2.1 *Embeddings* Estáticos

Os embeddings estáticos atribuem a cada palavra um único vetor, independentemente da sentença em que ela apareça. Representam as primeiras abordagens amplamente adotadas de representação semântica densa e permanecem relevantes como referência conceitual e como *baseline* de baixo custo.

***Latent Semantic Analysis* (LSA)**

Antes da utilização de redes neurais artificiais para a construção de embeddings, o LSA [22] foi a abordagem mais conhecida para reduzir a dimensionalidade dos vetores e, ao mesmo tempo, capturar a representatividade semântica das palavras. O LSA aplica a técnica de *singular value decomposition* (SVD) à matriz termo-documento, que originalmente representa a frequência de termos nos documentos. Essa decomposição reduz a dimensionalidade da matriz e revela relações latentes entre palavras e contextos, agrupando termos semanticamente relacionados. Como resultado, o LSA supera limitações dos métodos baseados apenas em correspondência exata, ajudando a identificar documentos relevantes mesmo quando diferentes termos descrevem conceitos semelhantes.

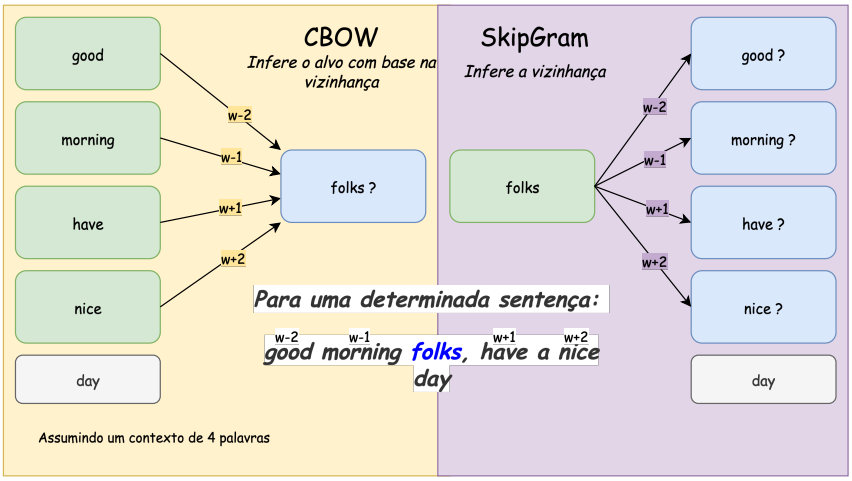
Word2Vec

Embora o LSA produza bons resultados, sua acurácia depende fortemente do volume de dados de treinamento. O Word2Vec [77] foi projetado para ser mais eficiente: utiliza redes neurais com arquiteturas simples, *Skip-gram* e *CBOW* (*Continuous Bag*

of Words), permitindo treinamento mais rápido sobre grandes corpora, inclusive com frameworks de processamento distribuído como o DistBelief [20].

Na arquitetura Skip-gram, o modelo prevê as palavras de contexto a partir de uma palavra-alvo: dada a palavra 'gato', tenta prever termos próximos, como 'animal' ou 'doméstico'. Esse método é mais lento, mas, segundo os autores funciona melhor para palavras raras. Já no CBOW, o modelo faz o inverso, usando as palavras ao redor para prever a palavra-alvo, o que o torna mais eficiente em corpora grandes. A Figura 6 ilustra as duas arquiteturas; ao final do treinamento, os pesos tornam-se os embeddings da palavra-alvo.

Figura 6 – Funcionamento do modelo CBOW e SkipGram.
Fonte: Elaboração própria.



Treinado em grandes volumes de dados, o Word2Vec mostrou-se eficaz em capturar padrões semânticos, mapeando palavras de contextos semelhantes para vetores próximos. Isso permite resolver analogias como $rei - homem + mulher = rainha$ com boa precisão.

Global Vectors (GloVe)

O GloVe [93] deriva os embeddings de uma matriz de ência de palavras em todo o corpus. Diferentemente do Word2Vec, que foca no contexto local de uma janela, o GloVe considera a coocorrência *global* entre palavras. Sua função objetivo minimiza a diferença entre o produto escalar dos embeddings e o logaritmo da contagem de coocorrência, aproximando no espaço vetorial palavras que ocorrem juntas com frequência ao longo do corpus. Por exemplo, 'cidade' e 'prefeito' podem raramente aparecer na mesma janela, mas têm alta coocorrência global por serem frequentemente mencionadas em discussões sobre governança urbana, relação que o GloVe captura mesmo sem proximidade local.

FastText

O FastText [9] foi desenvolvido para superar limitações do Word2Vec, sobretudo em idiomas com rica morfologia. Em vez de tratar cada palavra como unidade atômica, ele divide as palavras em N-gramas de caracteres, e a representação de uma palavra é a soma dos vetores de seus N-gramas. Assim, o modelo captura semelhanças entre palavras de estrutura parecida mesmo sem tê-las observado de forma idêntica. Em português, por exemplo, 'correr', 'correndo' e 'corremos' compartilham a raiz 'corr', representada por N-gramas como 'cor', 'orr' e 'rre'. Essa estratégia também lida melhor com palavras fora do vocabulário, gerando representações a partir dos N-gramas em vez de atribuir um vetor nulo. Para conter o crescimento do número de N-gramas, o FastText usa uma função de *hashing* que os distribui em um número fixo de *buckets*, reduzindo o custo de armazenamento sem prejudicar significativamente o desempenho.

Cada um desses modelos foi proposto para superar limitações do anterior, mas a literatura apresenta evidências contrastantes: há trabalhos em que o FastText supera os demais [58] e outros em que o Word2Vec se sai melhor [104]; na língua portuguesa, o GloVe já foi reportado com melhor desempenho [41]. A escolha do modelo ideal, portanto, depende do caso de uso.

3.2.2 *Embeddings* Contextualizados

Diferentemente dos estáticos, os embeddings contextualizados geram uma representação que depende da sentença inteira: a mesma palavra pode receber vetores distintos conforme o contexto linguístico, o que trata diretamente o problema da polissemia.

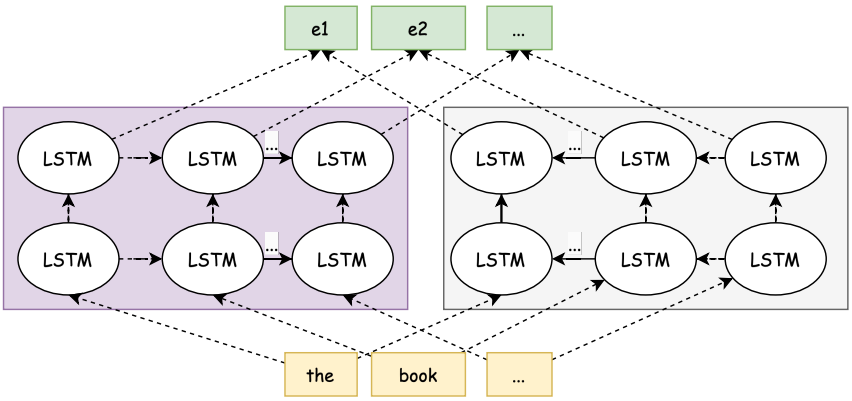
***Embeddings from Language Model* (ELMo)**

O ELMo [95] introduziu o termo *Deep Contextualized Pretraining* e baseia-se em uma rede biLM (*Bidirectional Language Model*) com camadas LSTM, também chamada biLSTM. A arquitetura, ilustrada na Figura 7, combina dois modelos de linguagem unidirecionais: um da esquerda para a direita e outro da direita para a esquerda, maximizando, respectivamente, $P(w_k \mid w_1, \ldots, w_{k-1})$ e $P(w_k \mid w_{k+1}, \ldots, w_N)$. A combinação dos dois sentidos gera representações sensíveis ao contexto: a palavra 'academia', por exemplo, recebe vetores distintos em 'academia de ginástica' e 'academia de ciências'.

***Bidirectional Encoder Representations from Transformers* (BERT)**

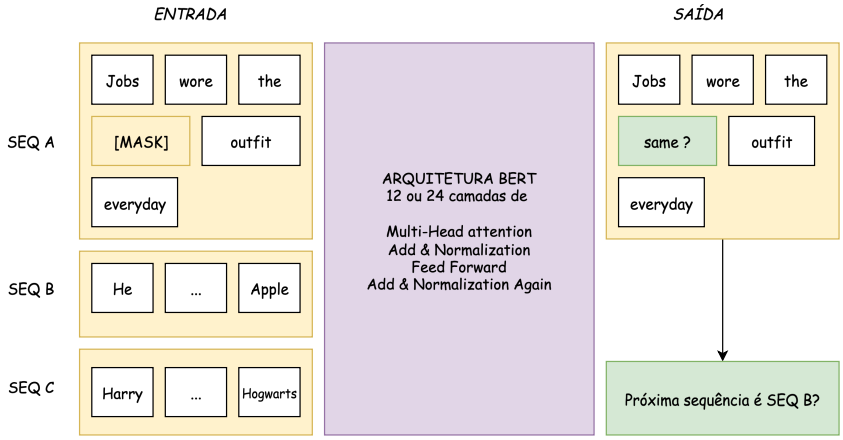
O BERT [21] foi o primeiro grande modelo a aplicar a arquitetura *Transformer* e o mecanismo de *self-attention* [132] ao processamento de linguagem natural de forma profundamente bidirecional: em vez de considerar apenas os *tokens* à esquerda do alvo, incorpora também os à direita, aproveitando o contexto completo da palavra durante o treinamento [50]. Sua arquitetura é composta apenas pelo *encoder* do *Transformer*, e o

Figura 7 – Ilustração da arquitetura biLSTM.
Fonte: Elaboração própria.



pré-treinamento combina duas tarefas: *Masked Language Modeling* (MLM), em que parte das palavras é mascarada e prevista a partir do contexto, e *Next Sentence Prediction* (NSP), que modela a relação entre sentenças. A Figura 8 ilustra ambas.

Figura 8 – Ilustração das tarefas de MLM e NSP.
Fonte: Elaboração própria.



A partir do BERT, surgiram diversas variantes adaptadas a diferentes necessidades: o RoBERTa [73] elimina o NSP e treina com mais dados; o DistilBERT [115] aplica *knowledge distillation* para produzir um modelo mais leve; e o ALBERT [59] reduz o número de parâmetros e, à época, reportou resultados no estado da arte em *benchmarks* populares como GLUE [138], RACE [57] e SQuAD [99]. Há também variantes especializadas por domínio ou estilo de linguagem, como o BERTweet [87], pré-treinado em *tweets*, e o BioBERT [64] e o ClinicalBERT [3], voltados a textos biomédicos e clínicos. Essa diversidade ilustra como a arquitetura base pode ser adaptada mantendo os princípios de pré-treinamento bidirecional e *self-attention*, e é especialmente relevante para esta dissertação, já que os modelos densos avaliados nos experimentos pertencem a essa linhagem de codificadores baseados em *Transformer*. Cabe notar, ainda, que revisões da literatura apontam haver amplo espaço de melhoria para sistemas de recuperação baseados em BERT, com o estado da arte em torno de 50% de acurácia em MRR@10 [139].

Generative Pre-trained Transformer (GPT)

O GPT [96], sucessivamente ampliado em escala e capacidade nas versões GPT-2 [97], GPT-3 [11] e GPT-4 [1], adota uma abordagem unidirecional: processa os *tokens* da esquerda para a direita e é composto por pilhas de *decoder* do *Transformer*. Seu pré-treinamento foca na tarefa de *Language Modeling*, prevendo a próxima palavra dada a sequência anterior, $P(w_t \mid w_1, \ldots, w_{t-1})$. Essa característica o torna eficaz na geração de texto fluente e coerente, embora apresente limitações em tarefas que exigem compreensão bidirecional completa, como a classificação de texto [151].

Visão Geral

A Tabela 3 sintetiza os modelos descritos, destacando o tipo de representação, a arquitetura e o ano de introdução, e evidencia a transição dos modelos estáticos para os contextualizados.

Modelo	Descrição
LSA [22]	Modelo estático, dos anos 90, usa decomposição de valores singulares (SVD) para reduzir a dimensionalidade de uma matriz termo-documento e capturar relações semânticas.
Word2Vec [77]	Modelo estático de 2013, utiliza janelas deslizantes para treinar uma rede neural com os métodos SkipGram ou CBOW, capturando coocorrências entre palavras.
GloVe [93]	Modelo estático de 2014, usa frequência global de coocorrência de palavras, gerando embeddings que preservam relações semânticas no espaço vetorial.
FastText [9]	Modelo estático de 2016, considera subpalavras (n-gramas) para capturar morfologia e lidar com palavras raras, melhorando a generalização.
ELMo [95]	Modelo contextualizado de 2018, baseado em LSTMs bidirecionais. Gera embeddings dependentes do contexto da palavra na sentença.
BERT [21]	Modelo contextualizado de 2018, usa Transformers bidirecionais, treinado em Masked Language Modeling (MLM), gerando embeddings ricos em contexto.
GPT [96]	Modelo contextualizado de 2018, baseado em Transformers unidirecionais, treinado para prever a próxima palavra, com pré-treinamento e transfer learning para NLP.

Tabela 3 – Visão geral dos modelos de embeddings.

Os modelos apresentados não constituem uma lista exaustiva; há diversas outras variações e avanços relevantes. Para uma visão mais abrangente, recomenda-se a consulta a artigos de revisão da área [71, 44]. Estabelecida a evolução no nível de palavra, a próxima

seção desloca o foco para o que de fato importa em recuperação: representar frases e documentos inteiros.

3.3 Representação de Frases e Documentos

Ao passar de palavras para frases e documentos, mudam a unidade de entrada, a saída e o objetivo de treinamento. Um modelo de embedding de palavra recebe como entrada um termo (ou sua janela de contexto) e produz, como saída, um vetor por palavra, aprendido a partir de padrões. Já um modelo de frase ou documento recebe como entrada uma sequência inteira de *tokens* e produz um único vetor de tamanho fixo que representa o texto como um todo; o objetivo de treinamento deixa de ser prever palavras vizinhas e passa a aproximar, no espaço vetorial, textos semanticamente relacionados, afastando os não relacionados. Essa é justamente a propriedade útil em recuperação, em que consulta e documento são comparados como unidades semânticas completas.

A forma mais simples de obter a representação de uma frase a partir de embeddings de palavra é a média dos seus vetores.

$$s = \frac{1}{n} \sum_{i=1}^{n} w_i, \tag{3.5}$$

A média é barata e pode servir como *baseline* competitivo em alguns cenários [4]; contudo, por descartar a ordem das palavras, ela é insensível à estrutura sintática. O trabalho de [60], por exemplo, mostrou que, com embeddings de LSA, a média produz o mesmo vetor para as duas frases seguintes, apesar de significados opostos:

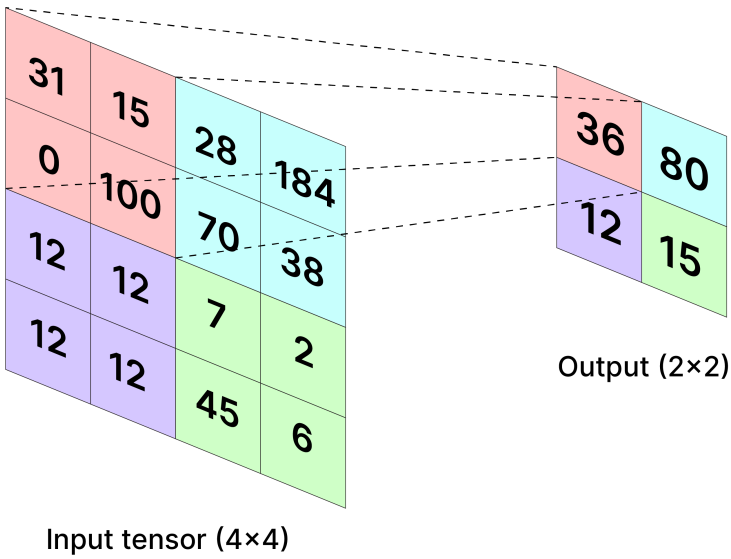
1. “It was not the sales manager who hit the bottle that day, but the office worker with the serious drinking problem.”;
2. “That day the office manager, who was drinking, hit the problem sales worker with a bottle, but it was not serious.”

A média é apenas uma das formas possíveis de composição, outras, como soma e multiplicação, também são exploradas [79], e cada uma apresenta vantagens conforme o caso de uso; a agregação por média, apesar de simples, já demonstrou ganhos relevantes em tarefas como a tradução automática [153].

De modo geral, a operação de resumir um conjunto de vetores em um único vetor é chamada de *pooling*, e a média é apenas o seu caso mais simples (o *mean pooling*). Na prática da programação, ao submeter um texto a um modelo, ele é primeiro dividido em *tokens* e codificado, produzindo uma matriz com um vetor por *token*; o *pooling* é a etapa que reduz essa matriz a um único vetor de tamanho fixo, independentemente do

comprimento do texto. As variantes mais comuns são o *mean pooling* (média dos vetores de *token*, como na Equação 3.5 e ilustrado na Figura 9), o *max pooling* (que toma o maior valor em cada dimensão) e o uso do vetor de um *token* especial de resumo, o `[CLS]` em modelos como o BERT. A comparação sistemática entre essas estratégias no contexto de embeddings de sentença foi conduzida por trabalhos como o InferSent [15] e o Sentence-BERT (SBERT) [100], este último dando origem à biblioteca `sentence-transformers`, na qual a agregação é implementada como uma camada explícita aplicada sobre a saída do codificador.

Figura 9 – Ilustração do *average pooling*: os vetores de cada *token* são agregados por média em um único vetor de tamanho fixo.
Fonte: https://iq.opengenus.org/global-average-pooling/.



Para superar as limitações da agregação simples, surgiram modelos que aprendem diretamente representações de unidades maiores. O Paragraph Vector [62], por exemplo, estende o Word2Vec acrescentando um vetor próprio para o documento, ajustado em conjunto com os embeddings das palavras durante o treinamento, o que captura informação semântica e sintática além de uma simples média. Na mesma direção, o SkipThought [54] adapta a ideia do Skip-gram ao nível de sentença, codificando uma sentença para reconstruir as vizinhas, com treinamento sobre a coleção BookCorpus [152]. Há ainda diversas abordagens supervisionadas voltadas a tarefas específicas, baseadas em redes recursivas e convolucionais [121, 49, 149, 150]. Estudos posteriores observaram, contudo, que os ganhos de modelos complexos concentravam-se em cenários *in-domain*, ao passo que, *out-of-domain*, métodos mais simples baseados em médias ponderadas de vetores podiam superá-los [143, 4], frequentemente avaliados sobre grandes bases de pares de sentenças como a PPDB (*Paraphrase Database*) [30].

Uma abordagem que ganhou destaque nesse contexto é o *aprendizado contrastivo*. A ideia central é ensinar o modelo a organizar o espaço vetorial por comparação, a partir de dois tipos de par:

- *pares positivos*: dois textos que deveriam ter representações próximas, como uma consulta e um documento relevante, ou duas paráfrases;
- *pares negativos*: textos não relacionados, que devem ficar distantes no espaço vetorial.

O codificador é então otimizado para aproximar os pares positivos e afastar os negativos. Em vez de prever palavras vizinhas, portanto, o modelo aprende diretamente uma noção de similaridade entre textos, que é exatamente o sinal útil em recuperação. O SimCSE [31] aplicou essa estratégia sobre o BERT, empregando inclusive a técnica de *Dropout* [124] como forma de *data augmentation*, e modelos subsequentes, como o E5 [140], que reporta superar o BM25 em recuperação *zero-shot*, seguiram essa mesma linha. Para acompanhar esse progresso de forma padronizada, o *Massive Text Embedding Benchmark* (MTEB) [84] avalia modelos em múltiplas tarefas e idiomas, evidenciando que não há um modelo universalmente superior; em seu painel público [144], modelos como o SGPT [82], o GRIT [83] e o NV-Retriever [19] figuram entre os de melhor desempenho, muitos deles ajustados em coleções de recuperação como o MS MARCO [88]. Vale ressaltar que esse painel é bastante dinâmico: novos modelos são propostos em ritmo acelerado, de modo que a lista de melhores colocados muda com frequência. Assim, recomenda-se ao leitor consultar diretamente o MTEB para obter um panorama atualizado do estado da arte.

Na prática, esses modelos de frase e documento não partem do zero: reaproveitam um codificador contextual de *tokens* e acrescentam uma etapa de agregação (*pooling*) que resume os vetores de *token* em um único vetor da sentença, reajustado por um objetivo de nível de sentença, frequentemente contrastivo. Esse arranjo é usualmente implementado por arquiteturas *bi-encoder*, nas quais consulta e documento são codificados de forma independente e comparados por similaridade vetorial [100]. Como as representações dos documentos podem ser pré-computadas e indexadas, essa arquitetura viabiliza a busca semântica em larga escala com bom equilíbrio entre eficácia e custo computacional. É a essa família de codificadores de sentença baseados em *Transformer* que pertencem os três modelos densos avaliados nesta dissertação:

- **MiniLM** (`all-MiniLM-L6-v2`) [142]: codificador compacto destilado da linhagem BERT/RoBERTa, com apenas seis camadas, posteriormente ajustado como modelo de *sentence embeddings*;
- **BGE** (`bge-small-en-v1.5`) [146]: codificador de arquitetura BERT treinado com aprendizado contrastivo, voltado à busca semântica e com boa eficiência em escala;
- **Contriever** (`contriever-msmarco`) [45]: também baseado na arquitetura BERT, pré-treinado por aprendizado contrastivo não supervisionado e, nesta variante, ajustado na coleção de recuperação MS MARCO.

Cabe destacar a conexão com os modelos apresentados na Seção 3.2: os três descendem diretamente do BERT, herdando seu *encoder* do tipo *Transformer*. O que os distingue não é o paradigma de representação, comum a eles, mas o *modo de construção*: o MiniLM privilegia a compactação por destilação, enquanto o BGE e o Contriever variam nos dados e na supervisão do treinamento contrastivo. Essa proximidade é intencional no recorte adotado: em vez de contrastar famílias distintas de representação, avalia-se o re-ranqueamento contextual sobre variações de uma mesma linhagem consolidada.

A etapa de *pooling* também varia entre eles, em suas configurações padrão. O MiniLM e o Contriever adotam o *mean pooling*, tomando a média dos vetores de *token* (Equação 3.5), ao passo que o BGE utiliza o vetor do *token* especial `[CLS]` como representação da sentença. Assim, mesmo dentro de uma mesma linhagem, as três estratégias de agregação discutidas anteriormente aparecem na prática.

A escolha desse trio foi guiada por dois critérios práticos. Primeiro, todos são **modelos de embedding de sentença**, adequados a comparar consultas e documentos como unidades completas, exatamente o tipo de representação exigido pelo experimento. Segundo, **reprodutibilidade computacional**: são modelos compactos, executáveis em ambiente de pesquisa acessível, incluindo um equipamento de porte pessoal (um MacBook Air com chip M4), sem dependência de infraestrutura especializada. Esse critério é reforçado pela disponibilidade pública dos três modelos na biblioteca *Hugging Face* [144], com pesos pré-treinados de acesso aberto, o que dispensa qualquer etapa de treinamento e facilita a replicação dos experimentos. A diversidade entre *paradigmas de recuperação* é contemplada, no experimento, pela presença de um modelo esparso (BM25) ao lado desses três modelos densos.

3.4 Combinação de Representações

Combinar representações é uma ideia herdada dos métodos de *ensemble* em aprendizado de máquina, agregar sinais complementares de diferentes modelos para obter uma representação mais robusta do que a de qualquer modelo isolado [123]. Essas estratégias costumam ser organizadas segundo o estágio em que ocorrem: na *early fusion*, as representações são combinadas no espaço vetorial antes da recuperação; na *late fusion*, combinam-se as listas ranqueadas produzidas após a recuperação [120, 5]. O propósito, em ambos os casos, é aproveitar a complementaridade entre modelos que capturam aspectos distintos do texto.

Trabalhos anteriores oferecem evidências de que essa combinação pode ser benéfica. A fusão de embeddings já trouxe ganhos em tarefas como NER e *chunking* [128], bem como em comparações sistemáticas de modelos como GloVe, Word2Vec, CSLM [117] e w2vf-deps [67], fundidos por técnicas como concatenação, PCA e *autoencoders* [134, 34, 33].

Um exemplo voltado à recuperação da informação é o Dual Embedding Space Model (DESM) [86], que combina as matrizes IN e OUT do Word2Vec para superar o BM25; a fusão também foi explorada em cenários de classificação, recomendação e multimodalidade [74, 98, 42, 7, 106, 2], embora a qualidade dos ganhos dependa fortemente da tarefa e do corpus [66, 53]. A combinação de representações mostra-se ainda particularmente relevante para idiomas sub-representados nos modelos pré-treinados: na língua portuguesa, há esforços de produção de embeddings dedicados [41] e avaliações de diferentes técnicas [17], ainda que sem foco em RI e sem contemplar modelos mais recentes como o BERTimbau [122].

No recorte desta dissertação, contudo, a combinação de embeddings não é o objeto principal de análise experimental. A combinação permanece como contexto conceitual e direção futura.

3.5 Funções de Ranqueamento

Uma representação textual não determina, por si só, a ordem dos resultados. Para produzir uma lista ranqueada, o sistema atribui uma pontuação a cada par formado pela consulta q e por um documento d. Funções de similaridade ou relevância são ordenadas do maior para o menor valor; funções de distância, no sentido inverso. Essa ordenação constitui o ranqueamento inicial.

Nos modelos lexicais, a pontuação é calculada a partir das ocorrências dos termos da consulta no documento. No BM25, cada termo contribui segundo sua raridade na coleção, sua frequência no documento, a saturação dessa frequência e a normalização pelo comprimento do texto; a soma dessas contribuições forma o escore final [101].

Nos modelos densos, a ordenação decorre da proximidade entre os vetores da consulta e dos documentos. O produto interno e a similaridade de cosseno atribuem valores maiores a vetores mais próximos em orientação, enquanto a distância euclidiana atribui valores menores a pontos geometricamente próximos [113, 100]. Para vetores normalizados, produto interno e cosseno produzem a mesma ordenação. Em ambos os paradigmas, a lista resume comparações diretas entre a consulta e cada documento; o capítulo seguinte discute como refiná-la por informações adicionais.

Em síntese, este capítulo percorreu a evolução das representações textuais, dos embeddings estáticos de palavra aos codificadores contextualizados de frases e documentos, delimitou o recorte adotado no experimento e mostrou como essas representações originam listas ranqueadas. Estabelecida essa base, o próximo capítulo trata do segundo estágio da recuperação: o re-ranqueamento das listas já produzidas.

4 Re-ranqueamento

Os capítulos anteriores estabeleceram os fundamentos da recuperação da informação, os principais pontos em que esses sistemas podem ser aprimorados e a importância central das representações textuais nesse processo. A contribuição desta dissertação parte justamente desses elementos: a partir de um sistema construído com essas peças, aprende-se informação contextual sobre as relações entre os próprios documentos recuperados e, com base nesse aprendizado, reordena-se qualquer conjunto de resultados. Essa etapa de reordenação é o *re-ranqueamento*, foco deste capítulo.

O re-ranqueamento consiste em refinar uma lista de resultados previamente recuperada, com o objetivo de melhorar a qualidade do ordenamento final. Em sistemas de recuperação típicos, os documentos são inicialmente selecionados por uma função de pontuação relativamente simples, como a similaridade léxica ou a correspondência entre representações independentes, que nem sempre captura sinais de relevância mais ricos [72]. Os métodos de re-ranqueamento atuam como um segundo estágio sobre os documentos mais bem posicionados, reordenando-os a partir de evidências adicionais, estratégia explorada, por exemplo, em abordagens clássicas baseadas em agrupamentos de documentos [65]. Este capítulo revisa, então, as principais famílias de técnicas correlatas a este trabalho: as estratégias clássicas de reordenação (Seção 4.1), o uso de redes neurais artificiais nessa tarefa (Seção 4.2), os métodos de fusão e agregação de listas ranqueadas (Seção 4.3) e, por fim, os métodos contextuais não supervisionados baseados em aprendizado de variedades (Seção 4.4), efetivamente empregados nesta dissertação.

4.1 Re-ranqueamento Clássico

O re-ranqueamento não é um desafio recente na recuperação da informação. As primeiras abordagens, como a realimentação de relevância (*relevance feedback*) [105, 111] e os modelos de relevância [61], já buscavam refinar os resultados iniciais por meio de ajustes posteriores à recuperação. A ideia comum a essas técnicas é reaproveitar os documentos mais bem posicionados para *reformular a consulta* e, então, refazer a busca com termos mais informativos. Elas diferem, essencialmente, na forma como identificam quais documentos tratar como relevantes.

Na **realimentação de relevância**, essa identificação depende de *julgamentos explícitos do usuário*: após a consulta inicial, o usuário indica quais documentos retornados são de fato relevantes, e o sistema usa esses exemplos para ajustar a consulta. No modelo vetorial, o método de Rocchio [105] desloca o vetor da consulta em direção aos documentos marcados como relevantes e para longe dos irrelevantes.

Na **realimentação de pseudo-relevância** (*pseudo-relevance feedback*), o usuário é retirado do processo: o sistema *assume automaticamente* que os k primeiros documentos recuperados são relevantes e extrai deles os termos para expandir a consulta. Os modelos de relevância [61] formalizam essa ideia, estimando uma distribuição de termos relevantes a partir dos documentos mais bem ranqueados, sem necessidade de supervisão manual. A Figura 10 sintetiza o fluxo comum às duas técnicas e o ponto exato em que se distinguem.

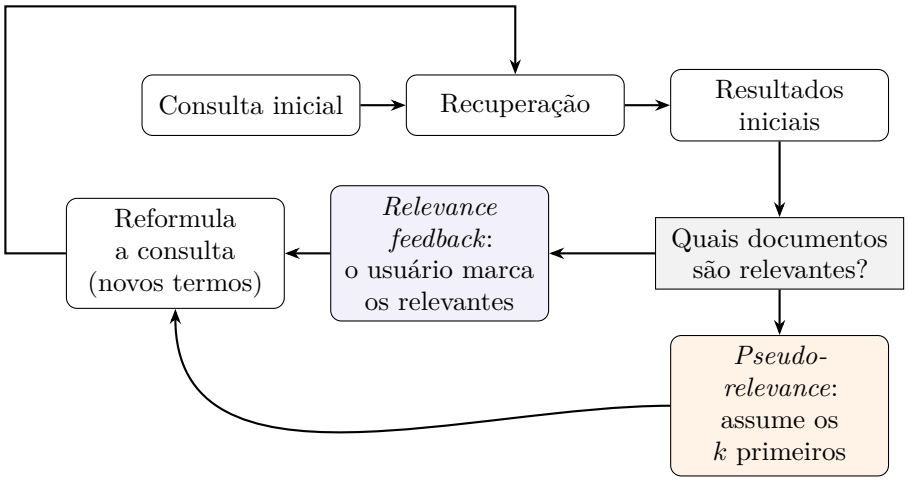


Figura 10 – Fluxo comum à realimentação de relevância e à pseudo-relevância. Ambas reaproveitam os resultados para reformular a consulta e refazer a busca; a diferença está na forma de identificar os documentos relevantes: por marcação do usuário ou por suposição sobre os primeiros colocados.
Fonte: Elaboração própria.

Enquanto a realimentação de relevância concentra-se em refinar a representação da consulta, as abordagens de aprendizado de ranqueamento (*Learning-to-Rank*, LTR) reformularam o problema como uma tarefa de otimização supervisionada. Em vez de modificar a consulta, os métodos LTR aprendem uma função de ranqueamento diretamente a partir de julgamentos de relevância rotulados [72]. Essas técnicas costumam ser organizadas em três famílias, que se distinguem pela *unidade* sobre a qual a função de perda é calculada durante o treinamento, conforme ilustra a Figura 11.

Na abordagem *pointwise*, cada documento é avaliado isoladamente: a função aprende a prever um valor de relevância absoluto por documento, tratando o problema como regressão ou classificação. Sua limitação é ignorar que ranquear é, essencialmente, comparar, de modo que acertar valores absolutos não garante a ordem correta. A abordagem *pairwise*, como no RankSVM [47], corrige esse ponto ao trabalhar sobre *pares* de documentos: a função aprende qual dos dois deve vir antes, penalizando inversões de ordem. Ainda assim, ao considerar pares isolados, ela trata todas as inversões como igualmente ruins, sem perceber que um erro no topo da lista ranqueada é mais grave que um erro no final. A abordagem *listwise*, como no LambdaMART [12], aborda a *lista inteira* de uma vez, otimizando diretamente (ou aproximando) métricas de ranqueamento, como o *NDCG*, o

que permite priorizar as primeiras posições. Apesar do avanço em relação às heurísticas clássicas, as três famílias compartilham uma limitação central: dependem de grandes volumes de anotações de relevância para o treinamento.

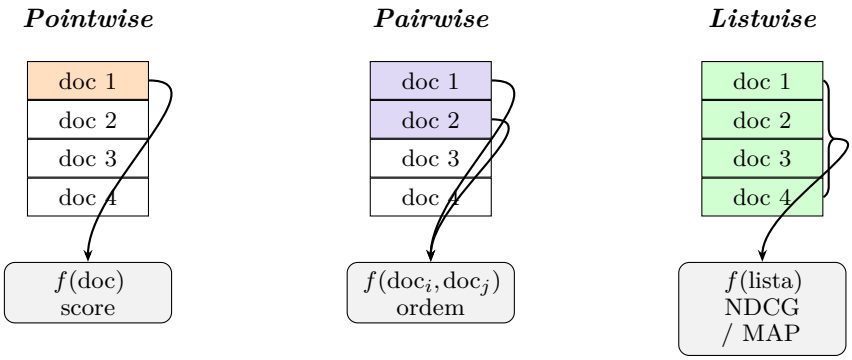


Figura 11 – As três famílias de *Learning-to-Rank*, distintas pela unidade avaliada pela função de perda: um documento por vez (*pointwise*), pares de documentos (*pairwise*) ou a lista inteira (*listwise*).
Fonte: Elaboração própria.

4.2 Recuperação e Re-ranqueamento Neural

A introdução de arquiteturas neurais profundas ampliou as abordagens de recuperação e re-ranqueamento, deslocando o foco da correspondência léxica esparsa para representações semânticas densas aprendidas. Embora modelos clássicos como o LSA já buscassem capturar relações semânticas latentes, os métodos neurais reduziram substancialmente a dependência da sobreposição exata de termos, codificando consultas e documentos em espaços vetoriais contínuos que permitem a correspondência por significado. Duas estratégias arquiteturais tornaram-se proeminentes nesse cenário e, juntas, ajudam a explicar por que a recuperação e o re-ranqueamento costumam ser tratados como estágios distintos: os *bi-encoders* e os *cross-encoders*, contrastados na Figura 12.

Na arquitetura *bi-encoder*, consulta e documento são codificados de forma independente em vetores densos, e a relevância é estimada por uma medida de similaridade, como o produto interno ou a similaridade do cosseno. Como as representações dos documentos podem ser pré-computadas e indexadas, geralmente com o apoio de índices de vizinhos aproximados como o FAISS [48], essa arquitetura é adequada à recuperação em larga escala, sendo a base dos modelos densos de primeiro estágio, como o DPR [51], o Contriever [45] e os modelos avaliados nesta dissertação. A limitação é que, ao codificar os textos separadamente, o *bi-encoder* não modela interações diretas entre os termos da consulta e do documento.

É justamente essa lacuna que a arquitetura *cross-encoder* preenche, sendo tipicamente empregada como componente de re-ranqueamento posterior à recuperação. Em *pipelines* de dois estágios, um recuperador inicial gera um conjunto de candidatos que

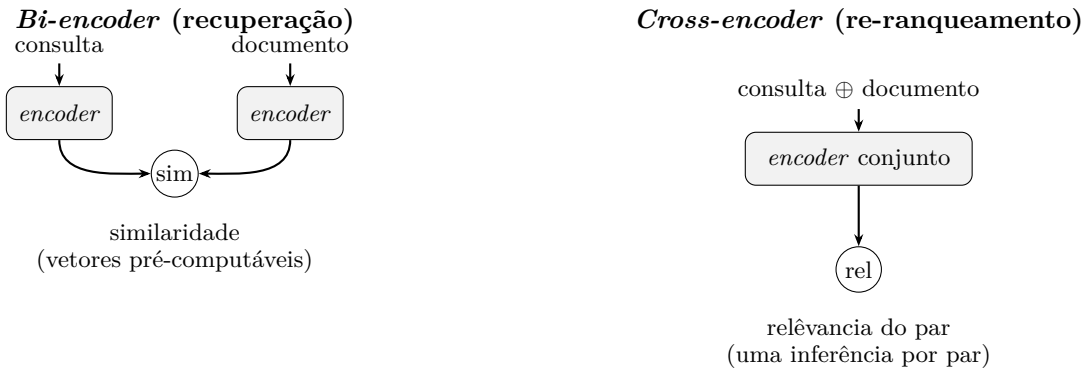


Figura 12 – Contraste entre as arquiteturas *bi-encoder* e *cross-encoder*. O *bi-encoder* codifica consulta e documento de forma independente, permitindo pré-computar e indexar os vetores dos documentos, o que o torna adequado à recuperação em larga escala. O *cross-encoder* processa o par consulta-documento em conjunto, capturando interações mais ricas ao custo de uma inferência por par, o que o torna típico do re-ranqueamento de poucos candidatos.
Fonte: Elaboração própria.

é então refinado pela codificação *conjunta* de cada par consulta-documento. Modelos baseados em *Transformer*, como o BERT, fortaleceram essa abordagem, uma vez que a autoatenção bidirecional possibilita uma interação contextual profunda entre os *tokens* da consulta e do documento; o re-ranqueamento de passagens com BERT [89] consolidou os *cross-encoders* como fortes *baselines* supervisionados. Essa qualidade tem um custo: por exigir uma inferência completa para cada par, o *cross-encoder* é inviável para pontuar a coleção inteira, ficando restrito a um conjunto reduzido de candidatos. Ademais, ainda opera em cenário supervisionado e estima a relevância de cada documento isoladamente, sem modelar explicitamente as relações estruturais entre os itens recuperados. Pela sua relevância como *baseline* supervisionado, um *cross-encoder* (`ms-marco-MiniLM-L-6-v2`) é utilizado, mais adiante, em uma comparação ilustrativa com o re-ranqueamento contextual proposto, em um subconjunto de bases.

4.3 Fusão e Agregação de Listas Ranqueadas

Diferentemente das abordagens anteriores, que reordenam uma única lista de resultados, as técnicas de fusão e agregação combinam evidências de múltiplas listas ranqueadas produzidas por diferentes sistemas de recuperação. Essa perspectiva é particularmente relevante para este trabalho, pois opera diretamente sobre listas ordenadas, aproximando-se dos métodos contextuais discutidos na sequência.

Essas técnicas costumam ser divididas em duas grandes famílias. A primeira é **baseada em *score***: usa as pontuações numéricas de relevância que cada sistema atribui aos documentos. Os métodos clássicos dessa família são o CombSUM, que soma as pontuações do mesmo documento nos vários sistemas, e o CombMNZ, que multiplica essa soma pelo

número de sistemas em que o documento aparece, favorecendo itens recuperados por muitos sistemas ao mesmo tempo [27]. Sua principal limitação é exigir que as pontuações de sistemas diferentes sejam normalizadas para uma escala comparável, o que nem sempre é trivial. A segunda família é **baseada em posição**: ignora as pontuações e utiliza apenas a ordem dos documentos em cada lista. O *Borda Count*, por exemplo, atribui a cada documento um número de pontos decrescente conforme sua posição (o primeiro colocado recebe mais pontos, o segundo um pouco menos, e assim por diante) e soma esses pontos entre as listas; já o *Condorcet* decide a ordem final por comparações par a par, verificando qual documento vence o outro na maioria das listas ranqueadas [23]. Por dependerem apenas da posição, esses métodos dispensam a normalização de *scores*.

A *Reciprocal Rank Fusion* (RRF) [16] pertence à família baseada em posição e é uma das técnicas mais difundidas: atribui a cada documento uma pontuação inversamente proporcional à sua posição em cada lista ranqueada, dispensando calibração ou supervisão. Formalmente, a pontuação de um documento d é dada por $\sum_r \frac{1}{k+\text{pos}_r(d)}$, em que $\text{pos}_r(d)$ é a posição de d na lista ranqueada r e k é uma constante (tipicamente $k = 60$). A Figura 13 ilustra a ideia: um mesmo documento tende a subir na lista ranqueada final quando aparece bem posicionado em vários sistemas ao mesmo tempo. A RRF é brevemente comparada, no capítulo de avaliação, como *baseline* de re-ranqueamento a partir de fusão; sua escolha justifica-se por ser um método simples, sem supervisão e amplamente utilizado.

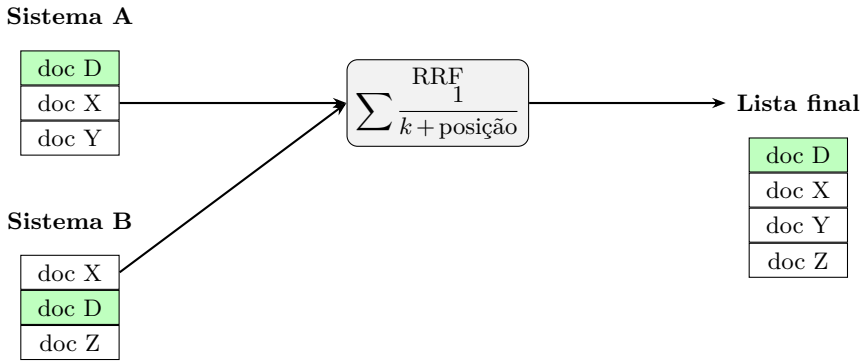


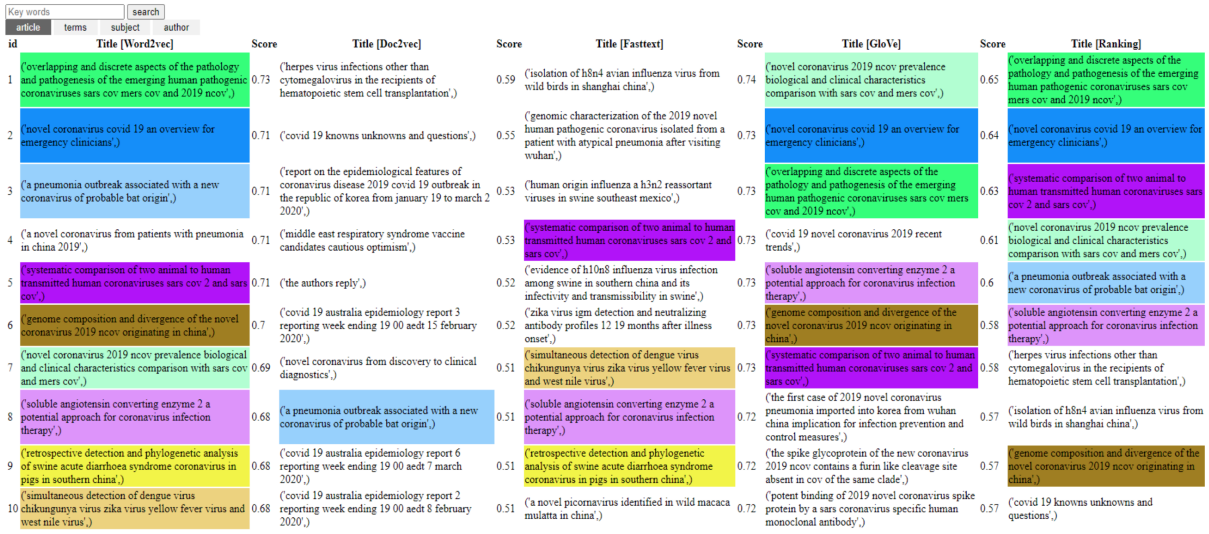
Figura 13 – Ideia da *Reciprocal Rank Fusion* (RRF). Cada documento recebe uma pontuação que soma o inverso de suas posições nos diferentes sistemas; documentos bem posicionados em várias listas ranqueadas, como o `doc D`, são promovidos na lista ranqueada final.
Fonte: Elaboração própria.

A abordagem [148] adota a fusão das listas ranqueadas após a recuperação, ou seja, os *embeddings* são gerados de forma independente e posteriormente integrados na lista ranqueada final. O trabalho avalia quatro algoritmos de *embeddings*: Word2Vec, Doc2Vec, FastText e GloVe. Além disso, é gerada uma quinta lista, que combina as quatro listas individuais utilizando a soma das posições em que cada documento aparece. Primeiramente, cada resultado recuperado nas listas individuais recebe uma pontuação com base em sua posição. Como as listas já estão ordenadas, o primeiro colocado recebe 9 pontos, o segundo

recebe 8 pontos, e assim sucessivamente até o décimo colocado, que recebe 0 pontos. A soma dessas pontuações em todas as listas individuais define o escore final de cada documento na lista agregada.

Por exemplo, considere uma consulta A, para a qual o documento D é recuperado nas seguintes posições nas listas individuais: {9, 8, 7, 1}. As pontuações atribuídas a D seriam {0, 1, 2, 8}, resultando em um escore final de 11 na lista combinada. A ordenação na lista agregada é, portanto, definida com base nos valores somados. Segundo os resultados apresentados no artigo, essa abordagem demonstrou desempenho superior nas métricas de Precisão@1 e Revocação@10. A Figura 14 demonstra em uma inteface gráfica um exemplo de comparação entre as listas produzidas em cada sistema e a combinação obtida.

Figura 14 – Interface de apresentação das listas ranqueadas na combinação de embeddings [148].
Fonte: [148]



Como complemento à avaliação experimental, será apresentada mais adiante uma comparação ilustrativa entre a fusão clássica por *Reciprocal Rank Fusion* (RRF), aplicada a diferentes combinações de listas ranqueadas produzidas pelos sistemas de recuperação, e métodos de re-ranqueamento contextual aplicados a listas individuais. A intenção é contrastar uma abordagem que agrega diretamente as posições dos documentos em listas independentes com outra que explora as relações de vizinhança entre os itens recuperados para refinar sua ordenação.

4.4 Re-ranqueamento Contextual Não Supervisionado

Em paralelo às abordagens supervisionadas, uma linha distinta de pesquisa explora a estrutura de variedade (*manifold*) das listas ranqueadas em vez de aprender a partir de rótulos de relevância. Até onde se tem conhecimento, os estudos mais reconhecidos

dessa linha concentram-se na recuperação multimídia, sobretudo de imagens. A premissa central do aprendizado de variedades é que a relevância deve ser inferida considerando a estrutura local do espaço de recuperação, e não tratando os documentos como pontos isolados [131]. Assim, em vez de pontuar cada documento independentemente em relação à consulta, esses métodos refinam a lista inicial a partir das relações de vizinhança entre os próprios itens recuperados.

Uma revisão abrangente da evolução e das categorias desses métodos é apresentada por Pereira-Ferrero et al. [94]. O estudo organiza a literatura de aprendizado não supervisionado de afinidades para recuperação de imagens em abordagens baseadas em difusão, em informações de ranqueamento e em aprendizado profundo, além de reunir uma revisão sistemática e uma avaliação experimental. Em razão do escopo desta dissertação, a exposição a seguir não pretende reproduzir essa revisão, mas concentra-se nos fundamentos e nos três métodos efetivamente empregados nos experimentos: CPRR, BFS-Tree e RDPAC.

A intuição por trás dessa ideia é que os documentos não se distribuem de forma arbitrária no espaço de representação: eles tendem a se concentrar sobre uma superfície de menor dimensão, a *variedade*. Nesse cenário, a distância direta entre a consulta e um documento pode ser enganosa, pois dois itens podem estar próximos em linha reta e, ainda assim, pertencer a estruturas distintas. Seguir a vizinhança, isto é, observar quais documentos compartilham vizinhos em comum, revela melhor a estrutura de relevância do que a proximidade imediata. A Figura 15 ilustra esse contraste.

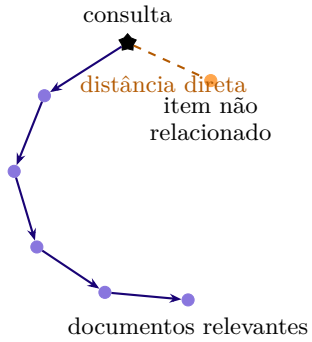


Figura 15 – Intuição do aprendizado de variedades. A distância direta (tracejada) aproxima a consulta de um item não relacionado, enquanto o caminho pela vizinhança (setas) percorre a estrutura da variedade e conecta a consulta aos documentos de fato relevantes.
Fonte: Elaboração própria.

Três métodos representativos dessa família fundamentam os experimentos desta dissertação: o CPRR [129], o BFS-Tree [92] e o RDPAC [37]. Antes de detalhar cada um, convém observar o que têm em comum, pois compartilham o mesmo esqueleto de processamento e diferem apenas no núcleo que propaga a afinidade entre os documentos.

Estrutura Comum aos Métodos

Os três métodos partem de um mesmo insumo, as listas ranqueadas do primeiro estágio de recuperação, e operam sobre a estrutura de vizinhança entre os documentos, e não sobre os vetores ou as pontuações originais. Mais que isso, compartilham o mesmo esqueleto de quatro etapas, ilustrado na Figura 16: (i) tomam como entrada uma lista ranqueada por documento, tratando cada documento como consulta e listando seus vizinhos mais próximos; (ii) convertem essas posições em uma matriz de afinidades, atribuindo pesos maiores aos vizinhos mais bem colocados; (iii) propagam a afinidade pela estrutura de vizinhança, reforçando pares apoiados por vizinhos em comum; e (iv) reordenam as listas segundo a matriz resultante. O que distingue os três métodos é exclusivamente a terceira etapa, o *núcleo de propagação*; as etapas de entrada, construção da matriz e reordenação são essencialmente as mesmas.

Na etapa (iii), o mecanismo de propagação varia entre os métodos. O CPRR utiliza produtos cartesianos sobre conjuntos de vizinhança para reforçar relações recorrentes [129]; o BFS-Tree organiza as referências em árvores cujas conexões são ponderadas por medidas de correlação entre listas ranqueadas [92]; e o RDPAC emprega um processo de difusão construído a partir de ponderações de ranqueamento [37]. As subseções seguintes detalham essas diferenças.

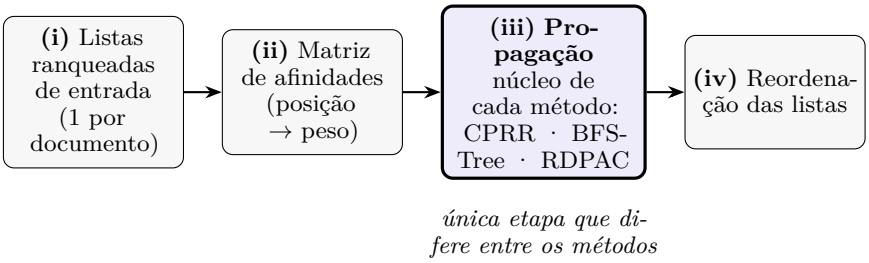


Figura 16 – Esqueleto comum aos três métodos de re-ranqueamento contextual. As etapas (i), (ii) e (iv) são praticamente idênticas; o que caracteriza cada método é apenas a etapa (iii), o núcleo que propaga a afinidade pela vizinhança. As próximas subseções detalham como CPRR, BFS-Tree e RDPAC realizam essa etapa.
Fonte: Elaboração própria.

Base de Exemplo

Para ilustrar alguns dos conceitos gerais em que se baseiam os métodos, adota-se uma única base fictícia, reutilizada em todos os exemplos desta seção. A base contém seis documentos de dois temas: três de futebol (F1, F2 e F3) e três de terror (T1, T2 e T3). O recuperador de primeiro estágio produz, para cada documento tratado como consulta, a lista ranqueada de seus vizinhos mais próximos, apresentada na Figura 17. Para simplificar, apenas uma das listas contém ruído: na consulta F1, o documento de terror T1 se infiltra na terceira posição, *à frente* de F3, um documento do próprio tema; as demais listas estão corretamente ordenadas. Trata-se, portanto, de um erro *unilateral*, pois a lista de T1, por

sua vez, não retribui essa proximidade (ela ranqueia os documentos de futebol apenas ao final, como esperado). É justamente esse desalinhamento que o re-ranqueamento deve corrigir: explorando a estrutura de vizinhança, deslocar o documento não relevante T1 para depois de F3 na lista de F1. As próximas subseções discutem como cada método realiza a etapa (iii) sobre exatamente essa mesma entrada.

futebol : F1, F2, F3 terror : T1, T2, T3

consulta	lista ranqueada (posições 1–6)					
F1	F1	F2	T1	F3	T2	T3
F2	F2	F1	F3	T1	T2	T3
F3	F3	F1	F2	T1	T2	T3
T1	T1	T2	T3	F1	F2	F3
T2	T2	T1	T3	F1	F2	F3
T3	T3	T1	T2	F1	F2	F3

Figura 17 – Base de exemplo utilizada em todas as ilustrações desta seção: seis documentos de dois temas (futebol e terror) e suas listas ranqueadas completas de entrada. A primeira posição de cada lista é o próprio documento; as demais ordenam todos os outros por similaridade. Para simplificar, somente a lista de F1 contém ruído: o documento de terror T1 (célula de cor destoante) invade a *terceira* posição, à frente de F3. O erro é unilateral, a lista de T1 não retribui essa proximidade.
Fonte: Elaboração própria.

CPRR (*Cartesian Product of Ranking References*)

A hipótese do CPRR [129] é que, se dois documentos aparecem juntos em uma mesma lista ranqueada, em posições no topo da lista, então provavelmente são de fato relacionados, ainda que a similaridade original entre eles fosse baixa. Para reforçar essas relações, o método toma a vizinhança de tamanho k de um documento e forma o *produto cartesiano* desse conjunto consigo mesmo, ou seja, gera todos os pares possíveis entre seus vizinhos. Cada vizinho recebe um peso conforme sua posição na lista (o primeiro vale mais), e cada par é reforçado por um valor igual ao produto dos pesos dos dois documentos; assim, pares formados por documentos bem posicionados recebem mais reforço.

Aplicando à base de exemplo, tome os três primeiros vizinhos de F1: F2, T1 e F3 (pesos 3, 2 e 1). O produto cartesiano gera os três pares mostrados na Figura 18. Isoladamente, esta referência ruidosa chega a atribuir o *maior* reforço ao par espúrio F2–T1 (6), justamente porque o documento não relevante T1 está bem posicionado nesta lista. A força do CPRR, porém, não está em uma referência isolada: o cálculo é repetido para *todos* os documentos como consulta, acumulado na matriz compartilhada e reiterado por um pequeno número de iterações (t). Pares de documentos genuinamente relacionados, como F1–F2 e F2–F3, reaparecem nas listas de vários documentos do mesmo tema e acumulam reforço a cada iteração, ao passo que o par espúrio F2–T1, preso a um ruído pontual, não se sustenta. É esse acúmulo, e não uma vizinhança isolada, que faz as relações verdadeiras prevalecerem.

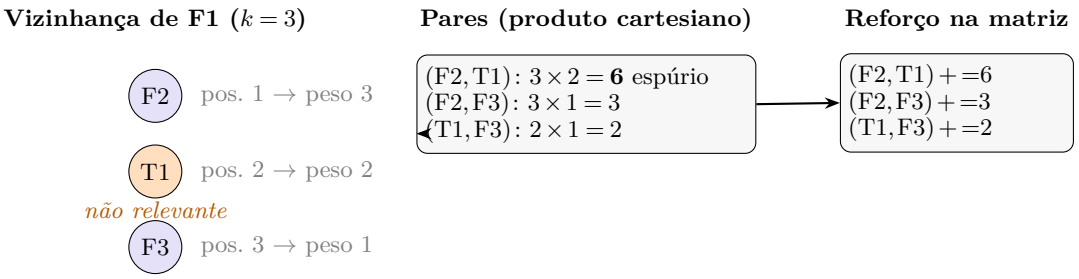
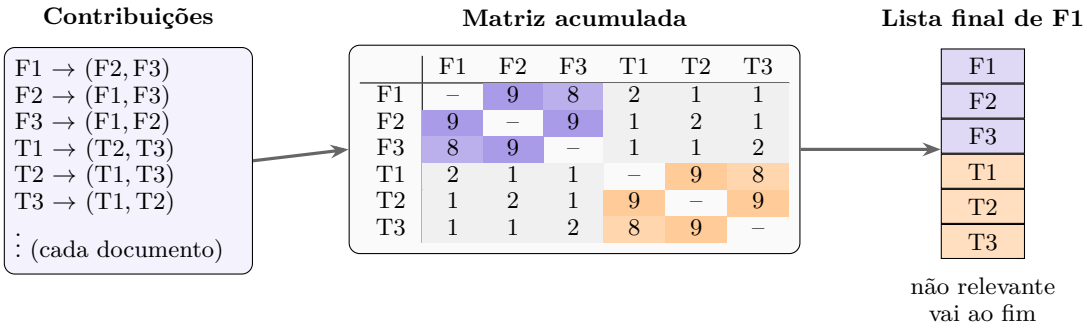


Figura 18 – Núcleo do CPRR sobre a base de exemplo (vizinhança de F1). Cada vizinho recebe um peso conforme sua posição; formam-se todos os pares (produto cartesiano) e cada par é reforçado pelo produto dos pesos. Nesta referência ruidosa, o par espúrio F2–T1 chega a receber o maior reforço, pois o documento não relevante T1 está bem posicionado. É o acúmulo desse cálculo sobre *todos* os documentos e ao longo das iterações que faz os pares recorrentes do mesmo tema (F1–F2, F2–F3) prevalecerem sobre os espúrios, que não se repetem.
Fonte: Elaboração própria.

A Figura 19 torna visível esse acúmulo. À esquerda, cada documento, tratado como consulta, despeja na matriz compartilhada as contribuições dos pares formados entre seus vizinhos do mesmo tema. Como esses pares se repetem de lista em lista, suas células vão somando e escurecendo; ao final, a matriz revela dois blocos densos de afinidade, um por tema, enquanto as ligações espúrias entre temas (que não se repetem) permanecem fracas. Ao reordenar a lista de F1 segundo essa matriz, os documentos de futebol ocupam as primeiras posições e o documento não relevante T1 é deslocado para o fim.



	F1	F2	F3	T1	T2	T3
F1	–	9	8	2	1	1
F2	9	–	9	1	2	1
F3	8	9	–	1	1	2
T1	2	1	1	–	9	8
T2	1	2	1	9	–	9
T3	1	1	2	8	9	–

Figura 19 – Acúmulo do CPRR na base de exemplo. Cada documento, como consulta, contribui com os pares entre seus vizinhos do mesmo tema; como esses pares se repetem entre as listas, somam-se na matriz compartilhada e formam dois blocos densos (futebol em azul, terror em laranja). As ligações espúrias entre temas, por não se repetirem, ficam fracas (≈ 1). A matriz reordena a lista de F1, promovendo os documentos de futebol e deslocando o documento não relevante T1 para o fim. Valores ilustrativos após o acúmulo e as iterações.
Fonte: Elaboração própria.

BFS-Tree

No CPRR, cada vizinhança é processada de forma isolada, e as relações indiretas só surgem pelo acúmulo global e pelas iterações. O BFS-Tree [92] adota uma estratégia

diferente: em uma única passada, encadeia explicitamente as vizinhanças, seguindo de um documento para seus vizinhos, depois para os vizinhos *desses* vizinhos, e assim por diante. Para isso, organiza a exploração como uma árvore percorrida em largura (*Breadth-First Search*, BFS), a técnica clássica de percorrer um grafo nível a nível.

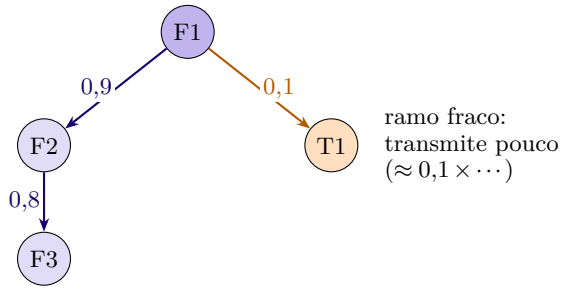
Partindo de um documento raiz, o primeiro nível da árvore é formado por seus k vizinhos; o segundo, pelos vizinhos de cada um desses vizinhos. A intuição é a de uma rede de indicações: se F1 aponta para F2 como semelhante, e F2 aponta para F3, então F3 é um bom candidato a também ser relevante para F1. A simples proximidade na árvore, porém, não basta: um vizinho pode ter sido incluído por engano no ranqueamento inicial. Para evitar propagar esses erros, o BFS-Tree pondera cada passo pela *correlação entre as listas ranqueadas* dos dois documentos: se ambos consideram praticamente os mesmos itens como semelhantes, suas listas estão fortemente correlacionadas e a conexão é confiável. Essa correlação é medida por métricas de comparação de listas ranqueadas, como o *Rank-Biased Overlap* (RBO), que dá mais peso à concordância nas primeiras posições.

A Figura 20 ilustra o núcleo do método sobre a base de exemplo, com F1 como raiz. Dois de seus vizinhos são F2 (mesmo tema) e o documento não relevante T1. As listas de F1 e F2 são muito parecidas (ambas contêm F1, F2, F3), resultando em alta correlação ($\text{cor} \approx 0{,}9$); já as listas de F1 e T1 quase não se sobrepõem, resultando em correlação baixa ($\text{cor} \approx 0{,}1$). Dois mecanismos determinam quanto de afinidade a raiz transmite adiante. Primeiro, as correlações *multiplicam-se ao longo do caminho*: F3, alcançado por F1 através de F2, recebe afinidade $\text{cor}(\text{F1},\text{F2}) \cdot \text{cor}(\text{F2},\text{F3}) = 0{,}9 \times 0{,}8 = 0{,}72$, valor alto porque toda a cadeia é consistente. Já qualquer documento alcançado pelo ramo de T1 recebe pouco, pois a cadeia começa por uma correlação baixa (0,1). Segundo, os caminhos *se acumulam*: um documento encontrado por vários caminhos confiáveis soma as contribuições de todos, o que o faz subir no ranqueamento.

Essas contribuições são acumuladas em uma matriz de afinidades ao longo de toda a coleção que, ao final, reordena as listas, promovendo os documentos bem conectados à raiz por caminhos confiáveis.

RDPAC (*Rank-based Diffusion Process with Assured Convergence*)

Enquanto o CPRR *constrói* a afinidade contando coocorrências em vizinhanças, o RDPAC [37] parte da matriz de afinidades já pronta e a *propaga*: seu núcleo é uma *difusão*, isto é, um espalhamento repetido dos valores existentes pela rede de vizinhanças. A cada rodada, todo documento distribui parte de sua afinidade acumulada aos seus vizinhos e recebe deles a parte correspondente; em seguida, a matriz é renormalizada. Não se formam novos pares nem se conta nada: apenas se redistribui, rodada após rodada, a afinidade que já estava na matriz.



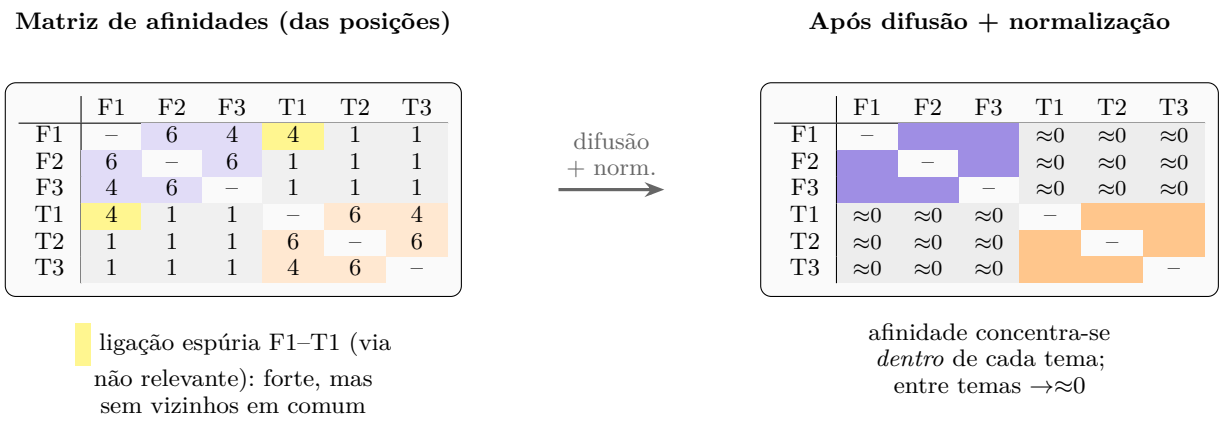
$$\text{afinidade}(\text{F1},\text{F3}) = \text{cor}(\text{F1},\text{F2}) \cdot \text{cor}(\text{F2},\text{F3}) = 0{,}9 \times 0{,}8 = \mathbf{0{,}72}$$

Figura 20 – Núcleo do BFS-Tree sobre a base de exemplo, com F1 como raiz. As correlações entre listas ranqueadas ponderam cada passo e se multiplicam ao longo do caminho: F3, alcançado por F1 através de F2 (ambos de futebol, listas muito correlacionadas), recebe afinidade alta; o ramo que passa pelo documento não relevante T1, cuja lista pouco se correlaciona com a de F1, transmite quase nada. Documentos alcançados por múltiplos caminhos confiáveis somam as contribuições.
Fonte: Elaboração própria.

É essa repetição que dá à difusão seu alcance: na primeira rodada, os valores se espalham para os vizinhos diretos; na seguinte, para os vizinhos dos vizinhos, e assim por diante, de modo que a afinidade avança um passo pela rede a cada rodada. O espalhamento repetido, porém, traz um risco: sem controle, os valores podem crescer indefinidamente, tornando o resultado instável. O RDPAC contorna isso com dois cuidados. Primeiro, restringe a difusão a uma vizinhança de k elementos, ou seja, cada documento só troca afinidade com seus k vizinhos mais próximos (um grafo de k vizinhos mais próximos), o que evita propagação para itens irrelevantes e reduz o custo computacional. Segundo, e sobretudo, após cada rodada *normaliza* a matriz de afinidades, reescalando os valores; é essa renormalização sistemática que impede o crescimento sem limite e assegura que o processo converja, característica que o distingue e que dá nome ao método.

Para tornar esse funcionamento concreto, a Figura 21 acompanha a difusão sobre a matriz de afinidades da base de exemplo. À esquerda, a matriz recém-construída a partir das posições (etapa comum). Como a lista de F1 tem ruído, o documento não relevante cria uma ligação espúria entre temas com valor nada desprezível (F1–T1 vale 4), comparável a um par legítimo do mesmo tema (F1–F3 também vale 4); localmente, portanto, a matriz é ambígua. A difusão então propaga afinidade pelos vizinhos em comum: F1 e F2 reforçam-se por compartilharem o vizinho F3, e o mesmo ocorre dentro do bloco de terror; já o par F1–T1 não tem vizinho algum em comum (os vizinhos de F1 são de futebol, os de T1 são de terror), de modo que não recebe reforço e, após a normalização, colapsa para um valor desprezível. À direita, a matriz estabilizada mostra a afinidade concentrada *dentro* de cada tema e praticamente nula entre eles, exatamente o sinal que, ao reordenar as listas, promove os documentos do mesmo tema e rebaixa o documento não relevante.

Esses dois cuidados, a restrição de vizinhança e a normalização, são o que assegura



	F1	F2	F3	T1	T2	T3
F1	–	6	4	4	1	1
F2	6	–	6	1	1	1
F3	4	6	–	1	1	1
T1	4	1	1	–	6	4
T2	1	1	1	6	–	6
T3	1	1	1	4	6	–

	F1	F2	F3	T1	T2	T3
F1	–			≈0	≈0	≈0
F2		–		≈0	≈0	≈0
F3			–	≈0	≈0	≈0
T1	≈0	≈0	≈0	–		
T2	≈0	≈0	≈0		–	
T3	≈0	≈0	≈0			–

Figura 21 – Núcleo do RDPAC: difusão sobre a matriz de afinidades da base de exemplo. À esquerda, a matriz construída a partir das posições: os blocos internos de cada tema (azul, laranja) são fortes, mas o ruído faz surgir uma ligação espúria entre temas (amarelo, F1–T1 $= 4$) tão intensa quanto um par legítimo (F1–F3 $= 4$), tornando a matriz localmente ambígua. A difusão reforça apenas os pares que compartilham vizinhos; como F1 e T1 não têm vizinhos em comum, após a normalização a ligação espúria colapsa para ≈ 0 (à direita), enquanto os pares do mesmo tema se consolidam. A normalização a cada rodada impede o crescimento indefinido e assegura a convergência.
Fonte: Elaboração própria.

a *convergência* do método: em vez de crescer sem limite, as afinidades se estabilizam em valores bem definidos após algumas rodadas. É essa propriedade que dá nome ao método, cujas três características ficam assim resumidas: baseado em informações de ranqueamento (usa apenas posições), por difusão (propaga afinidade pela vizinhança) e com convergência assegurada (os valores se estabilizam). Ao final, a matriz estabilizada reordena as listas, promovendo os documentos que acumularam mais afinidade ao longo da difusão.

Uma característica essencial e comum a esses três métodos é que eles operam exclusivamente sobre as posições ordinais das listas ranqueadas, sem acessar vetores de características, pontuações de similaridade ou qualquer informação específica da modalidade dos dados. Essa propriedade os torna, em princípio, diretamente transferíveis para qualquer cenário de recuperação. Apesar de demonstrarem ganhos expressivos em *benchmarks* de recuperação de imagens [131, 130, 91], sua aplicação à recuperação textual permanecia, até onde se tem conhecimento, inexplorada, lacuna que esta dissertação investiga sistematicamente e que constitui o cerne de sua contribuição experimental.

Esses três métodos estão reunidos no *Unsupervised Distance Learning Framework* (UDLF) [131], *framework* que padroniza a representação de listas ranqueadas e a execução de algoritmos de re-ranqueamento e fusão baseados em informações de ranqueamento. Por consumir exclusivamente posições ordinais, o UDLF é, em princípio, agnóstico à modalidade dos dados, o que torna sua aplicação ao domínio textual, objeto desta dissertação, uma extensão natural, ainda que até então inédita. O próximo capítulo detalha como esses métodos foram operacionalizados sobre listas ranqueadas textuais, descrevendo os modelos

de recuperação empregados, os conjuntos de dados utilizados e o protocolo experimental adotado para avaliar seu impacto.

Além de padronizar a execução dos métodos de re-ranqueamento contextual descritos nesta seção, o UDLF também viabiliza estratégias de fusão contextual baseadas em informações de ranqueamento, nas quais listas produzidas por múltiplos sistemas podem compor a evidência de vizinhança usada para construir uma lista final. Essa possibilidade não foi avaliada no protocolo experimental desta dissertação, que se concentrou em medir o efeito de CPRR, BFS-Tree e RDPAC sobre listas individuais de recuperadores esparsos e densos. Sua exploração constitui uma direção promissora para investigar se a complementaridade entre diferentes recuperadores pode ser combinada à evidência contextual de vizinhança.

5 Metodologia Proposta

Os capítulos anteriores percorreram a evolução das representações textuais e das técnicas de re-ranqueamento, das estratégias clássicas às neurais supervisionadas. Cada uma dessas famílias apresenta vantagens e limitações: os *cross-encoders* oferecem alta qualidade, mas exigem dados rotulados e inferência custosa; a fusão de listas ranqueadas combina sistemas, mas não cria sinal de relevância novo; e os modelos densos, embora poderosos, ainda ranqueiam cada documento de forma isolada. Diante desse panorama, emerge a oportunidade investigada nesta dissertação: avaliar se uma técnica de re-ranqueamento contextual não supervisionada, amplamente avaliada em recuperação de imagens, pode ser transferida para o domínio textual.

A premissa central, retomada da introdução, é que o re-ranqueamento baseado em aprendizado de variedades, ao explorar a estrutura de vizinhança entre os documentos recuperados em vez de tratá-los como pontos isolados, pode constituir uma abordagem *agnóstica à modalidade dos dados*. Por operar apenas sobre as posições ordinais das listas ranqueadas, sem acessar vetores de características ou rótulos de relevância, a técnica pode aproveitar representações geradas por diferentes modelos, esparsos ou densos, independentemente do paradigma de treinamento. Essa propriedade a torna particularmente conveniente em cenários nos quais anotações de relevância são escassas ou inexistentes e é a hipótese examinada pelo protocolo experimental deste trabalho.

Em termos operacionais, cada execução percorre um fluxo único. Inicialmente, os documentos de um conjunto de dados são indexados por um sistema de recuperação, seja ele esparso ou denso. Em seguida, cada documento do corpus também é submetido como consulta, produzindo uma lista ordenada de seus vizinhos mais similares. Esse conjunto de listas ranqueadas iniciais descreve a estrutura de vizinhança da coleção e é convertido para o formato consumido pelo UDLF, que retém apenas as posições dos documentos. Um dos métodos contextuais então explora essas relações para reordenar as listas, cujos resultados são reconvertidos para o formato de avaliação e analisados pelas métricas de efetividade. Embora conceitualmente linear, essa execução envolve indexação, conversões de formato, configuração de parâmetros, rastreamento de artefatos e avaliação; a Figura 22 sintetiza esse fluxo.

Para investigar essa hipótese de forma sistemática, este trabalho conduz uma avaliação em larga escala que reúne **nove conjuntos de dados** de cinco domínios distintos, **quatro modelos de recuperação** (um esparso e três densos) e **três métodos de re-ranqueamento contextual** (CPRR, BFS-TREE e RDPAC), variados em **nove valores** do parâmetro de vizinhança K, totalizando **972 experimentos** de re-ranqueamento.

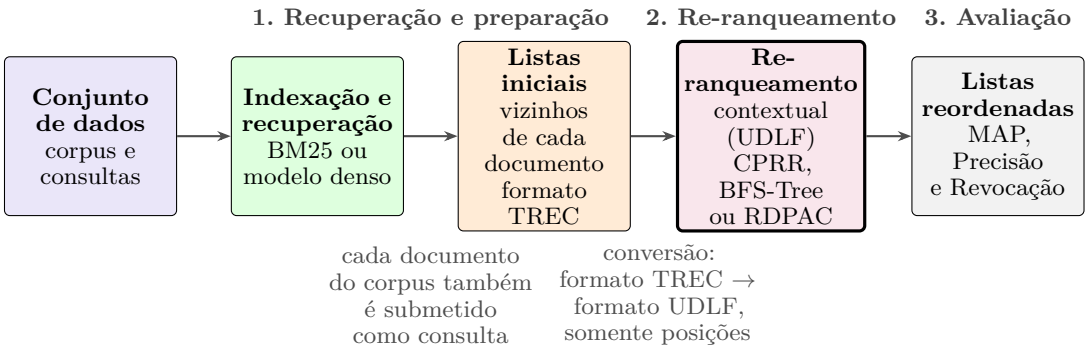


Figura 22 – Visão geral de uma execução experimental. Os documentos de cada conjunto são indexados e recuperados por um modelo esparso ou denso. Como o re-ranqueamento contextual depende das relações internas da coleção, cada documento também é utilizado como consulta, produzindo uma lista de vizinhos por documento. Essas listas são convertidas para o formato baseado exclusivamente em posições consumido pelo UDLF, que aplica um método contextual e produz listas reordenadas, posteriormente avaliadas por MAP, Precisão e Revocação.
Fonte: Elaboração própria.

Todos os métodos são aplicados sobre as listas produzidas por cada modelo de recuperação, sem qualquer ajuste supervisionado, e avaliados pelas métricas MAP, Precisão e Revocação em múltiplos pontos de corte. A operacionalização desse processo é viabilizada pelo *Unsupervised Distance Learning Framework* (UDLF) [131], originalmente concebido para recuperação de imagens, cuja adaptação ao domínio textual é descrita nas próximas seções. A Figura 23 ilustra esse desenho combinatório, em que cada conjunto de dados se desdobra em quatro modelos, cada modelo em três métodos e cada método em nove valores de K.

Complementarmente à avaliação principal, dois experimentos ilustrativos posicionam o re-ranqueamento contextual em relação a abordagens alternativas: uma comparação com um *cross-encoder* supervisionado e outra com a fusão de listas ranqueadas por *Reciprocal Rank Fusion*. Esses experimentos não constituem um estudo comparativo extensivo, mas oferecem contexto sobre o comportamento da técnica frente a um *baseline* supervisionado e a um *baseline* de *ensemble*.

5.1 Fluxo de Dados e Etapas

A execução dos experimentos foi estruturada sobre dois pilares principais: (i) a implementação de quatro sistemas de recuperação da informação independentes, um esparso (BM25) e três densos (MiniLM, BGE e Contriever), responsáveis pela geração inicial das listas ranqueadas; e (ii) o uso do *Unsupervised Distance Learning Framework* (UDLF) [131] para a aplicação dos três métodos de re-ranqueamento contextual (CPRR, BFS-TREE e RDPAC) sobre cada uma dessas listas.

O UDLF, originalmente concebido para recuperação de imagens, exige como entrada

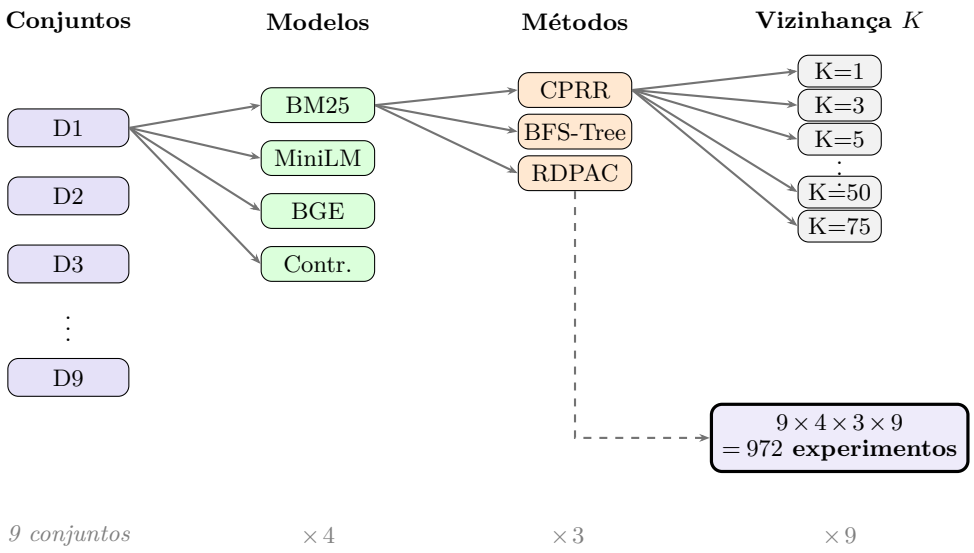


Figura 23 – Desenho combinatório da avaliação. Cada um dos **9 conjuntos de dados** é recuperado por **4 modelos**; cada par (conjunto, modelo) é re-ranqueado por **3 métodos**; e cada tripla (conjunto, modelo, método) é avaliada em **9 valores** do parâmetro de vizinhança K. Para não poluir, apenas um ramo é expandido em cada nível; a ramificação completa produz $9 \times 4 \times 3 \times 9 = 972$ experimentos de re-ranqueamento.
Fonte: Elaboração própria.

dados no formato de listas ranqueadas em notação horizontal (denominadas aqui *UDLF Lista Ranqueada*), onde cada linha da lista corresponde a uma consulta e as colunas notam os documentos recuperados, ordenados por relevância. Outra implicação importante é que, para cada conjunto de dados utilizado, não apenas as consultas devem ser submetidas ao programa, mas todos os itens do corpus devem ser utilizados como consulta, pois o contexto de similaridade entre os documentos do corpus é requisito funcional para a execução das técnicas.

Essa diferença de estrutura implica um processo de transformação em duas etapas: inicialmente, a saída TREC (denominação das saídas convencionais em listas verticais, adotadas na convenção da conferência) dos sistemas de recuperação é convertida para o formato *UDLF Lista Ranqueada* para permitir a aplicação dos métodos de re-ranqueamento; em seguida, a saída produzida pelo UDLF é reconvertida para o formato TREC, a fim de viabilizar o cálculo das métricas de avaliação.

O pipeline experimental, portanto, compreende cinco etapas, detalhadas a seguir com suas respectivas representações visuais.

5.1.1 Etapa 1 - Indexação dos Documentos

Inicialmente, realiza-se o *download* dos conjuntos de dados e a indexação dos documentos em cada sistema de RI, habilitando-os para responder às consultas. Essa etapa garante que o conteúdo esteja estruturado para busca eficiente. A Figura 24 ilustra o fluxo

desse processo.

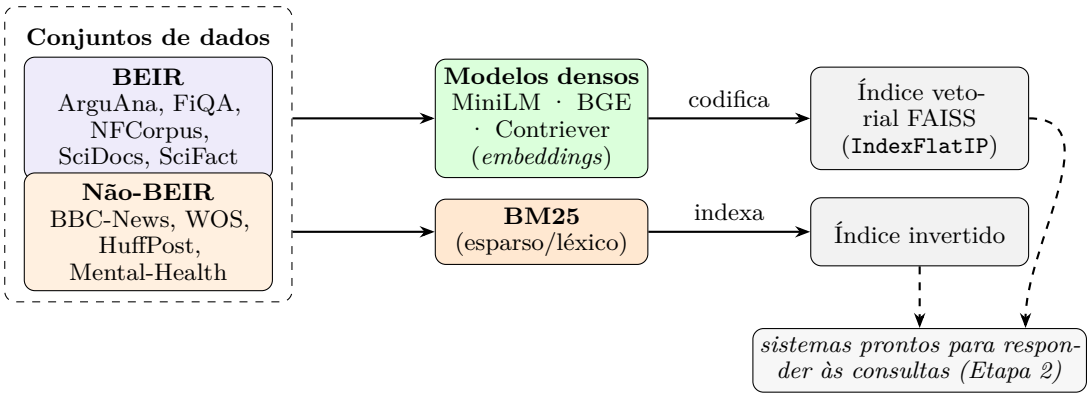


Figura 24 – Etapa 1 - Indexação dos documentos nos sistemas de RI. Cada conjunto de dados (BEIR e Não-BEIR) tem seus documentos indexados por dois caminhos: os **modelos densos** codificam cada documento em um vetor, armazenado em um índice vetorial FAISS do tipo `IndexFlatIP`, enquanto o **BM25** constrói um índice invertido esparso. Ao final, todos os sistemas ficam prontos para responder às consultas.
Fonte: Elaboração própria.

Nos três recuperadores densos, os vetores foram indexados com a biblioteca FAISS [48], utilizando o índice exato `IndexFlatIP`, que ordena os documentos pelo produto interno com a consulta. Para MiniLM e BGE, os vetores foram normalizados antes da indexação, fazendo com que o produto interno corresponda à similaridade de cosseno. No Contriever, manteve-se a configuração do modelo sem normalização; nesse caso, a ordenação utiliza diretamente o produto interno. Essa distinção foi preservada tanto na indexação quanto na execução das consultas.

5.1.2 Etapa 2 - Execução das Consultas

Com os sistemas indexados, cada modelo é consultado para produzir as listas ranqueadas iniciais no formato TREC. Há, porém, uma particularidade decisiva para o re-ranqueamento contextual: como os métodos baseados em aprendizado de variedades operam sobre a estrutura de vizinhança *entre os próprios documentos*, não basta submeter apenas as consultas do conjunto de dados. É preciso submeter *todo documento do corpus como se fosse uma consulta*, obtendo, para cada um, a lista dos documentos mais similares a ele. É esse conjunto de listas, uma por documento, que descreve as relações de similaridade internas do corpus e constitui a entrada exigida pelo UDLF. A Figura 25 ilustra essa etapa.

5.1.3 Etapa 3 - Preparação da Entrada para o UDLF

A saída dos modelos está no formato TREC, uma tabela *vertical* em que cada linha registra um par consulta–documento com sua posição e pontuação. O UDLF, porém,

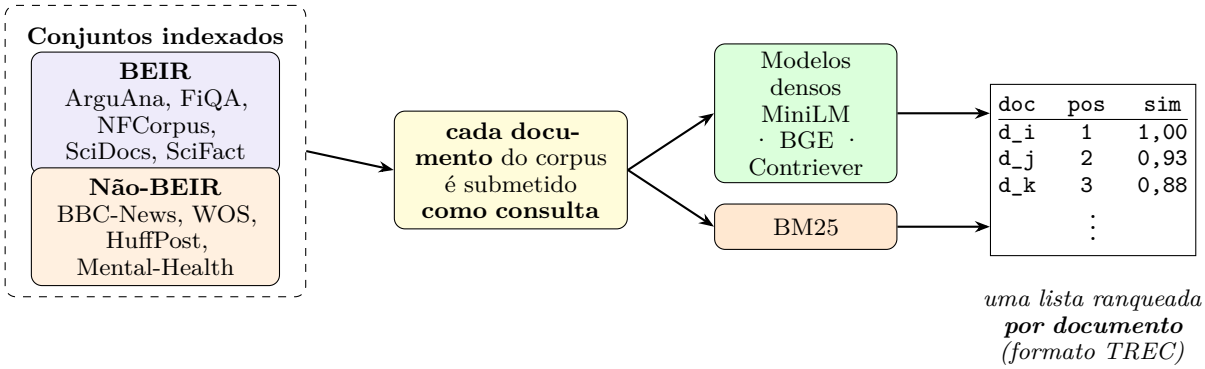


Figura 25 – Etapa 2 - Execução das consultas e geração das listas ranqueadas iniciais. Diferentemente de uma avaliação de recuperação convencional, em que se submetem apenas as consultas do conjunto, aqui **cada documento do corpus é submetido como consulta** a cada modelo. Produz-se, assim, uma lista ranqueada por documento, com seus vizinhos mais similares, conjunto que descreve as relações de similaridade internas do corpus e serve de entrada para o aprendizado de variedades do UDLF.
Fonte: Elaboração própria.

espera a entrada em formato *horizontal*: um arquivo (`ranked-list.txt`) em que cada linha corresponde a um documento-consulta e enumera, da esquerda para a direita, os identificadores de seus vizinhos em ordem de similaridade. A conversão consiste, portanto, em agrupar as linhas do TREC por consulta e transpô-las, descartando as pontuações e retendo apenas a ordem, coerente com o fato de que os métodos operam somente sobre posições ordinais.

Além dessa lista, o UDLF exige um segundo arquivo, `lists.txt`, que é simplesmente a lista *vertical* de todas as chaves (os identificadores de todos os documentos que aparecem nas listas ranqueadas), uma por linha. Ele funciona como um dicionário que associa cada identificador à sua posição, permitindo ao *framework* referir-se aos documentos por índice. Esse par de arquivos, `ranked-list.txt` (as vizinhanças) e `lists.txt` (as chaves), constitui entrada suficiente para a reorganização: a partir dele, o UDLF reconstrói toda a estrutura de vizinhança necessária ao re-ranqueamento. A Figura 26 ilustra essa conversão.

5.1.4 Etapa 4 - Configuração e Execução do Re-ranqueamento

Com a entrada preparada, cada execução é especificada por um arquivo de configuração de extensão `.ini`, que declara ao *framework* três informações: qual **método** aplicar (CPRR, BFS-TREE ou RDPAC), qual **parâmetro de vizinhança** K utilizar e qual **lista ranqueada de entrada** processar (o par `ranked-list.txt` e `lists.txt` da etapa anterior). Como cada combinação de conjunto de dados, modelo, método e valor de K demanda o seu próprio `.ini`, essa etapa gera uma grande quantidade de arquivos de configuração, um por experimento.

A saída do UDLF é, novamente, uma lista ranqueada, agora reordenada pelo método

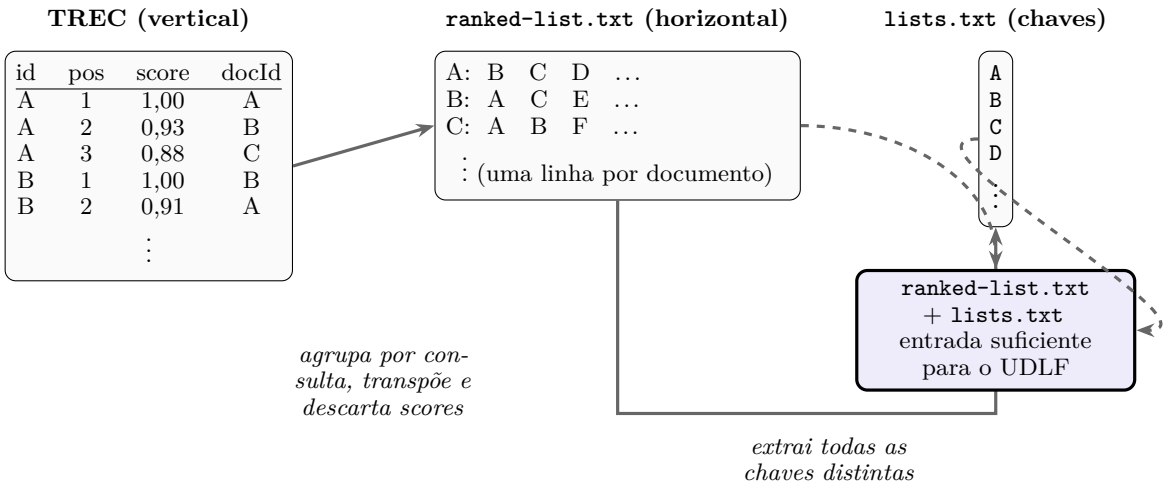


Figura 26 – Etapa 3 - Conversão das listas ranqueadas para o formato *UDLF Lista Ranqueada*. A saída TREC *vertical* (um par consulta–documento por linha) é agrupada por consulta e transposta em `ranked-list.txt`, arquivo *horizontal* em que cada linha lista, em ordem de similaridade, os vizinhos de um documento (as pontuações são descartadas, pois os métodos usam apenas posições). Gera-se ainda `lists.txt`, a lista vertical de todas as chaves (identificadores). Esse par de arquivos é a entrada suficiente para a reorganização pelo UDLF.
Fonte: Elaboração própria.

contextual. Para viabilizar a avaliação, ela é reconvertida ao formato TREC vertical. Cabe notar que, nessa reconversão, as pontuações são atribuídas de forma *linear* a partir das posições (o primeiro colocado recebe a maior pontuação, decrescendo uniformemente), pois o UDLF preserva apenas a ordem, não valores de similaridade calibrados. Por esse motivo, este trabalho não emprega métricas sensíveis à magnitude das pontuações, como o *NDCG*, restringindo-se a métricas baseadas em posição e em relevância binária (MAP, Precisão e Revocação). A Figura 27 esquematiza essa etapa.

5.1.5 Etapa 5 - Avaliação dos Resultados

A etapa final compara as listas reordenadas com as listas de primeiro estágio, medindo o efeito do re-ranqueamento. Para todos os conjuntos de dados avaliam-se as métricas MAP, Precisão e Revocação em diferentes pontos de corte (@5, @10, @20, @50, @100 e @200). Toda avaliação depende de um conjunto de relevância, os *qrels*, que relaciona cada consulta aos documentos considerados relevantes no corpus; a forma de obtê-lo, porém, difere entre os dois grupos de conjuntos.

Nos conjuntos **BEIR**, os *qrels* já são fornecidos pela coleção: um arquivo que lista, para cada consulta, os documentos julgados relevantes. Basta, portanto, reutilizá-los diretamente.

Nos conjuntos **não-BEIR** (BBC-News, WOS, HuffPost e Mental-Health), que são coleções de classificação sem *qrels* próprios, o conjunto de relevância foi construído a

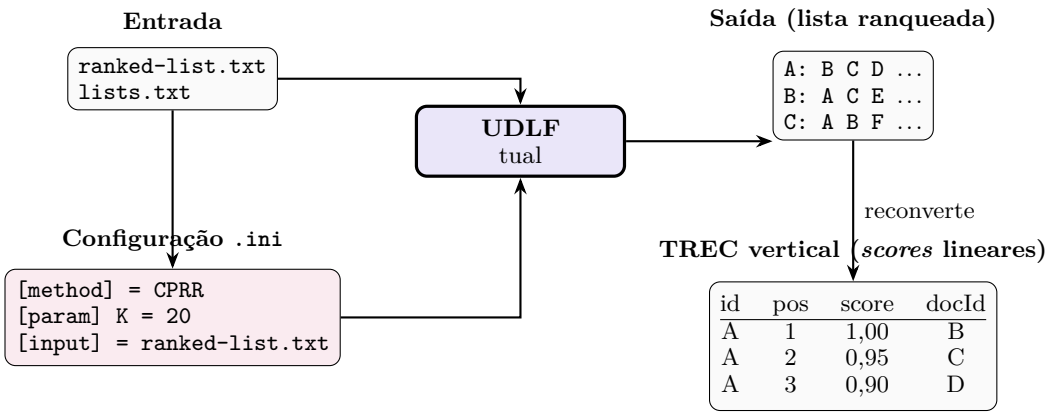


Figura 27 – Etapa 4 - Aplicação dos métodos de re-ranqueamento contextual no UDLF. Um arquivo `.ini` especifica ao *framework* o método, o parâmetro de vizinhança K e a lista ranqueada de entrada; como há um `.ini` por combinação de conjunto, modelo, método e K, geram-se muitos arquivos de configuração. A saída é outra lista ranqueada, reconvertida ao formato TREC vertical com *scores* atribuídos linearmente a partir das posições, o UDLF preserva apenas a ordem, razão pela qual não se empregam métricas sensíveis à magnitude como o *NDCG*.
Fonte: Elaboração própria.

partir do atributo de *categoria* de cada documento: uma consulta da categoria A considera relevante todo documento também da categoria A (excluindo a própria consulta). Gera-se, assim, um arquivo de *qrels* no mesmo formato dos conjuntos BEIR, permitindo tratamento uniforme na avaliação. A Figura 28 sintetiza a construção dos *qrels* e o cálculo das métricas.

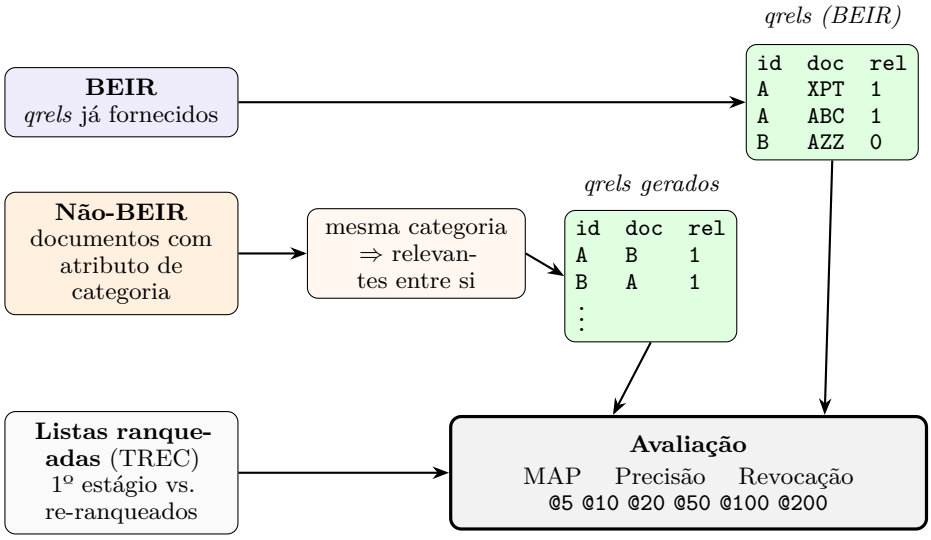


Figura 28 – Etapa 5 - Construção dos conjuntos de relevância e cálculo das métricas. Nos conjuntos BEIR, os *qrels* são fornecidos pela própria coleção. Nos conjuntos não-BEIR, são construídos a partir do atributo de categoria: documentos de mesma categoria são relevantes entre si. Com os *qrels* e as listas (de primeiro estágio e reordenadas) no formato TREC, calculam-se MAP, Precisão e Revocação nos pontos de corte @5 a @200.
Fonte: Elaboração própria.

Essa configuração define o protocolo metodológico do estudo e estabelece as etapas

reprodutíveis de execução do pipeline. A avaliação quantitativa e qualitativa dos resultados obtidos a partir desse protocolo é apresentada no capítulo seguinte.

5.2 O Framework UDLF Text Espresso

O detalhamento das etapas anteriores evidencia que executar o experimento sem automação exigiria um processo manual complexo e altamente suscetível a falhas: são muitas conversões de formato, uma quantidade grande de arquivos de configuração e centenas de execuções que precisam ser rastreadas com consistência. Nessas condições, o processo seria difícil de acompanhar e não ofereceria o rigor e a reprodutibilidade exigidos pela experimentação proposta. Para resolver isso, foi construído, no âmbito deste trabalho, um arcabouço experimental dedicado, o *UDLF Text Espresso*, que automatiza o pipeline de ponta a ponta.

O arcabouço integra, em um único fluxo reprodutível, todas as etapas descritas: coleta de dados, indexação, recuperação (esparsa e densa), re-ranqueamento e cálculo das métricas. Internamente, o re-ranqueamento apoia-se no PyUDLF[1], uma camada de API em Python que envolve o executável do UDLF, permitindo configurar métodos e parâmetros por código, em uma única linguagem, sem manipular o binário diretamente. As execuções são orientadas por arquivos de configuração declarativos: cada corrida especifica o conjunto de dados, o modelo de recuperação, o método de re-ranqueamento e os parâmetros (como o valor de K), de modo que qualquer combinação do desenho experimental pode ser reproduzida a partir da mesma configuração. A Figura 29 resume como esses componentes se conectam.

Vale distinguir o papel de cada camada, pois o PyUDLF, sozinho, cobre apenas o re-ranqueamento. O *UDLF Text Espresso* é a aplicação que orquestra todo o restante do ciclo experimental: sabe *onde* obter cada conjunto de dados e, no caso das coleções BEIR, faz automaticamente o *download* a partir do endereço público; dispõe de módulos de recuperação para todos os modelos avaliados (o esparso BM25 e os densos); e realiza a avaliação da recuperação com o cálculo das métricas. O PyUDLF entra apenas como a etapa final de re-ranqueamento dentro desse fluxo. Trata-se, portanto, de uma aplicação **modular**: acrescentar um novo recurso (um modelo de recuperação, um conjunto de dados ou uma métrica) resume-se a adicionar o módulo correspondente e configurá-lo no *pipeline* de execução, seja tornando-o padrão, seja acionando-o por parâmetros explícitos.

Esse desenho traz três benefícios diretos para o estudo. Primeiro, **determinismo e rastreabilidade**: cada etapa registra suas saídas em uma estrutura de diretórios canônica (por exemplo, os índices, as listas ranqueadas de primeiro estágio, as entradas do UDLF e as métricas de cada experimento), o que torna possível auditar e reexecutar qualquer parte

[1] https://github.com/UDLF/pyUDLF

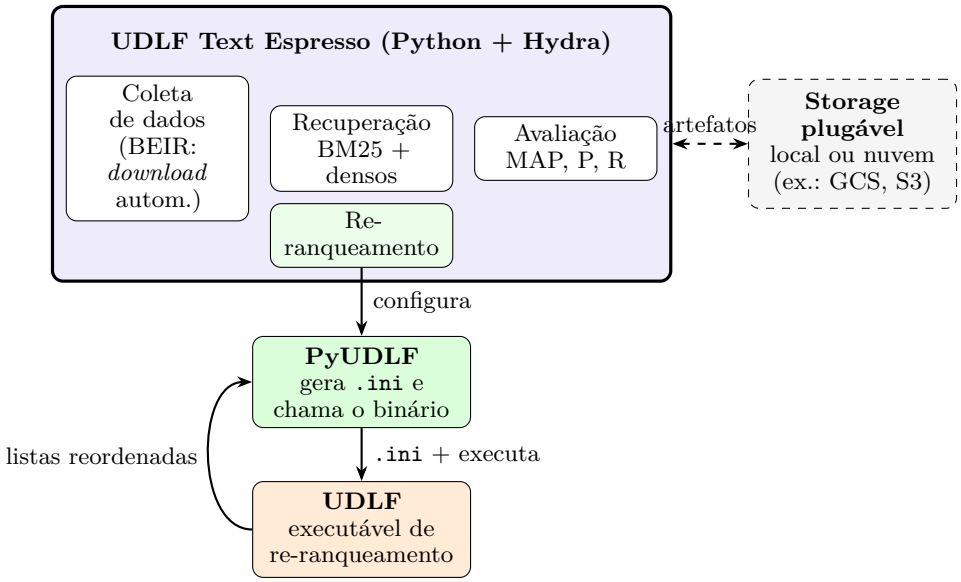


Figura 29 – Arquitetura do *UDLF Text Espresso*. A aplicação, escrita em Python e orquestrada pelo Hydra, reúne módulos próprios para coleta de dados (com *download* automático das coleções BEIR), recuperação (BM25 e modelos densos) e avaliação. Apenas o módulo de re-ranqueamento delega ao PyUDLF, que gera o `.ini` e aciona o executável do UDLF, retornando as listas reordenadas. Os artefatos de cada etapa podem ser persistidos em um *storage* plugável, local ou em nuvem. Novos recursos entram como módulos adicionais no mesmo *pipeline*.
Fonte: Elaboração própria.

do processo. Segundo, **reaproveitamento**: índices e listas ranqueadas já computadas são armazenados em cache e reutilizados entre execuções, evitando recomputar etapas custosas ao variar apenas o método ou o parâmetro de vizinhança. Terceiro, **escalabilidade experimental**: por reduzir cada experimento a uma configuração declarativa, o arcabouço viabiliza a execução sistemática das centenas de combinações que compõem esta avaliação, mantendo padrões uniformes e facilmente reprodutíveis. É essa automação que torna praticável a avaliação em larga escala apresentada no próximo capítulo.

A orquestração é feita com o *Hydra* [147], uma biblioteca de gerenciamento de configuração que permite compor e sobrescrever parâmetros diretamente pela linha de comando. Na prática, com o ambiente preparado, um experimento completo reduz-se a uma única chamada, em que se declaram o conjunto de dados, o modelo e as etapas a executar. Por exemplo, para coletar os dados, indexar, recuperar e avaliar o *baseline* BM25 sobre o conjunto SciFact:

```
python -m udlf_text_espresso.runner \
  run_id=paper-assets \
  dataset=scifact \
  model=bm25 \
  pipeline.steps='[data_collection,index,retrieve,metrics]'
```

Trocar o modelo (por exemplo, `model=bge-small-en-v1.5`) ou os passos executados (por

exemplo, `pipeline.steps='[rerank,metrics]'`) basta para compor uma nova execução, reaproveitando os índices e listas ranqueadas já em cache. Essa simplicidade de uso, uma vez preparado o ambiente, é o que permite percorrer todo o desenho experimental com esforço manual mínimo e alta reprodutibilidade.

Definidos o protocolo metodológico e o arcabouço que o operacionaliza, o próximo capítulo aplica esse pipeline e analisa os resultados obtidos.

6 Avaliação Experimental

Este capítulo apresenta os resultados obtidos com a metodologia proposta, cobrindo caracterização dos conjuntos avaliados, configuração dos experimentos, análise quantitativa, comparação com *baselines* e discussão qualitativa.

6.1 Conjuntos de Dados

O experimento aplica o pipeline previamente descrito em nove conjuntos de dados: cinco coleções estabelecidas do *framework* BEIR [126], as cinco menores, e quatro conjuntos de dados adaptados, originalmente projetados para classificação mas transformados para a avaliação de recuperação.

Esta coleção diversa permite uma avaliação abrangente em diferentes domínios, tamanhos de corpus e características de relevância. A Tabela 4 resume as características principais de todos os conjuntos de dados.

Tabela 4 – Resumo dos conjuntos de dados utilizados nos experimentos. Os conjuntos BEIR utilizam julgamentos de relevância padrão (*qrels*); os conjuntos baseados em categoria tratam cada documento como uma consulta e definem a relevância pelo compartilhamento de categoria (excluindo a própria consulta).

Conjunto	#Docs	#Consultas	Méd. Rel/Q	Tipo de Aval.	Domínio
ArguAna	8,674	1,406	1.0	Qrels	Argumentação
SciFact	5,183	300	1.1	Qrels	Afirmações científicas
NFCorpus	3,633	323	38.2	Qrels	Biomédico
SciDocs	25,657	1,000	4.9	Qrels	Documentos científicos
FiQA	57,638	648	2.6	Qrels	Finanças
BBC-News	2,225	2,225†	450.6	Categoria (5)	Notícias
WOS	46,985	46,985†	389.0	Categoria (134)	Acadêmico
HuffPost	32,730	32,730†	2,044.0	Categoria (42)	Notícias
Mental-Health	20,353	20,353†	5,593.8	Categoria (4)	Fóruns de saúde

† Cada documento serve simultaneamente como consulta e como resultado potencial. A média de relevantes por consulta (Méd. Rel/Q) dos conjuntos baseados em categoria conta os documentos da mesma categoria (excluindo a própria consulta).

6.1.1 Conjuntos de Dados BEIR

O BEIR (*Benchmarking-IR*) [126] é um *benchmark* público e amplamente adotado para avaliação de sistemas de recuperação, que agrega dezenas de coleções heterogêneas (de domínios como argumentação, finanças, medicina e ciência) sob um formato uniforme, com ênfase na avaliação *zero-shot*, isto é, sem ajuste do modelo à coleção-alvo. Cada

Tabela 5 – Distribuição de documentos relevantes por consulta (Rel/Q). Para os conjuntos BEIR, Rel/Q é o número de documentos julgados relevantes por consulta a partir dos *qrels* oficiais. Para os conjuntos baseados em categoria, Rel/Q conta os documentos da mesma categoria, excluindo a própria consulta; os percentis são ponderados pelo tamanho da categoria.

Conjunto	Tipo de Aval.	Média	σ	Mín	Mediana	P75	P95	Máx
ArguAna	Qrels	1	0	1	1	1	1	1
SciFact	Qrels	1.1	0.5	1	1	1	2	5
NFCorpus	Qrels	38.2	71.7	1	16	39	158	475
SciDocs	Qrels	4.9	0.3	2	5	5	5	5
FiQA	Qrels	2.6	2.1	1	2	3	7	15
BBC-News	Categoria (5)	450.6	55	385	416	509	510	510
WOS	Categoria (134)	389	119.5	0	383	417	651	749
HuffPost	Categoria (42)	2,044	1,865.3	124	1,744	3,539	5,593	5,593
Mental-Health	Categoria (4)	5,593.8	1,386.8	2,581	5,451	7,053	7,053	7,053

coleção acompanha seus *qrels*, os julgamentos de relevância que relacionam consultas e documentos. Por ser padronizado e de acesso aberto, o BEIR facilita a comparação direta com outros trabalhos da área que reportam resultados sobre as mesmas coleções, o que motivou sua adoção neste estudo. Os cinco menores conjuntos de dados BEIR utilizados foram:

ArguAna

O ArguAna[136] foca na recuperação de argumentos, contendo 8.674 documentos de portais de debates. Cada consulta é uma declaração sobre um tópico controverso, e a tarefa é recuperar contra-argumentos. Este conjunto de dados é único por ter exatamente um documento relevante por consulta (1.406 consultas no total), tornando-o particularmente desafiador para algoritmos de ranqueamento, pois não há flexibilidade nos julgamentos de relevância.

FiQA

O conjunto de dados de Perguntas e Respostas Financeiras (FiQA)[75] contém 57.638 documentos financeiros provenientes de posts do Reddit e StackExchange. Com 648 consultas, representa necessidades de informação financeira do mundo real. O conjunto de dados exibe densidade de relevância moderada com uma média de 2,63 documentos relevantes por consulta (intervalo: 1-15), refletindo a natureza especializada das consultas financeiras.

NFCorpus

O NFCorpus [10] (Corpus de Fatos Nutricionais) compreende 3.633 documentos médicos do PubMed. Apesar de seu tamanho de corpus pequeno, possui 323 consultas com distribuições de relevância altamente variáveis (média: 38,2 documentos relevantes por consulta, intervalo: 1-475). Esta alta variância o torna um caso de teste interessante, já que algumas consultas têm relevância densa enquanto outras são esparsas.

SciDocs

O SciDocs[14] contém 25.657 artigos científicos e é projetado para tarefas de recomendação de citações. Com 1.000 consultas e um padrão de relevância consistente (aproximadametne documentos relevantes por consulta, intervalo:1-5), representa um cenário de recuperação bem balanceado típico de aplicações de busca acadêmica.

SciFact

O SciFact [137] foca na verificação de afirmações científicas com 5.183 resumos científicos. As 300 consultas representam afirmações científicas que precisam de evidências de apoio ou refutação. Com uma média de 1,13 documentos relevantes por consulta, apresenta um desafio de recuperação orientado à precisão.

6.1.2 Conjuntos de Dados Não-BEIR

Para avaliar a generalização da abordagem, foram adaptados quatro conjuntos de dados não originalmente projetados para tarefas de recuperação. Em todos eles, cada documento atua simultaneamente como consulta e como alvo de recuperação, e a relevância é definida pelo compartilhamento de categoria.

Cabe registrar que HuffPost e Mental-Health possuem dimensão elevada para o protocolo adotado, sobretudo porque cada documento também é submetido como consulta e o custo do pipeline cresce com o tamanho do corpus. A utilização integral dessas coleções dificultaria a execução sistemática de todas as combinações de modelo, método e parâmetro, enquanto uma truncagem arbitrária poderia comprometer a diversidade de domínios e perfis de relevância. Optou-se, portanto, por uma *amostragem estratificada por categoria*: de cada categoria, seleciona-se aleatoriamente uma fração fixa dos documentos, preservando as proporções originais entre as classes. Para garantir reprodutibilidade, a seleção utiliza uma semente fixa (o parâmetro `random_state=42` da biblioteca *pandas* [127]), de modo que a mesma amostra seja obtida em novas execuções. Os conjuntos BBC-News e WOS, de porte compatível com o protocolo completo, foram utilizados integralmente. Os detalhes de cada amostragem são apresentados a seguir.

BBC News

O conjunto de dados BBC News[36] contém 2.225 artigos de notícias em cinco categorias (negócios, entretenimento, política, esporte, tecnologia). Este conjunto de dados de classificação é transformado em um modelo para de recuperação através de:

- Uso de cada documento como consulta potencial e alvo de recuperação, com isso, a criação de 2.225 consultas.
- Definição de relevância baseada na associação de categoria: documentos são relevantes se compartilham a mesma categoria.

Esta transformação resulta em alta densidade de relevância (média: 450,64 documentos relevantes por consulta), simulando um cenário de recomendação de documentos onde usuários buscam artigos similares.

Web of Science (WOS)

O conjunto de dados WOS[56] compreende 46.984 publicações científicas com metadados ricos incluindo áreas de pesquisa. Similar ao BBC News, o conjunto é adaptado para recuperação através de:

- Uso do campo "área" para definir relações de relevância
- Criação de consultas a partir dos próprios documentos do corpus
- Estabelecimento de relevância quando documentos compartilham a mesma área de pesquisa

Com 46.985 consultas e uma média de 389,0 documentos relevantes por consulta (intervalo: 1-749), este conjunto de dados apresenta um desafio de recuperação em larga escala com distribuições de relevância altamente variáveis.

HuffPost

O conjunto de dados HuffPost [78] reúne notícias do portal *The Huffington Post*, distribuídas em 42 categorias temáticas. Por ser originalmente muito extenso (cerca de 164 mil notícias), utilizou-se a amostra estratificada de **20% por categoria** descrita acima, resultando em 32.730 documentos que preservam as proporções das classes originais. Por manter um grande número de categorias de granularidade fina, representa um cenário desafiador de recuperação por categoria, com média de 2.044,0 documentos relevantes por consulta. Sua alta cardinalidade de categorias o torna complementar ao BBC-News, que possui apenas cinco classes mais amplas.

Mental-Health

O conjunto de dados Mental-Health [85] compreende publicações de fóruns de saúde mental, organizadas em quatro categorias (*Anxiety*, *Depression*, *Normal* e *Suicidal*). A partir da coleção original (cerca de 40 mil publicações), descartaram-se textos com menos de 50 caracteres e aplicou-se a amostra estratificada de **50% por categoria**, resultando em 20.353 documentos. Com a maior densidade de relevância dentre todos os conjuntos avaliados (média de 5.593,8 documentos relevantes por consulta), constitui um cenário extremo de recuperação em que cada consulta possui um grande número de documentos relevantes, útil para observar o comportamento do re-ranqueamento sob alta densidade de relevantes.

6.2 Configuração Experimental

6.2.1 UDLF

Toda a avaliação foi conduzida por meio do *UDLF Text Espresso* (Seção 5.2), que orquestra as etapas de coleta, recuperação, re-ranqueamento e avaliação e, internamente, aciona o *Unsupervised Distance Learning Framework* (UDLF) para o re-ranqueamento contextual. A primeira etapa foi conduzida pelos quatro modelos de recuperação já apresentados, um esparso (BM25) e três densos (MiniLM, BGE e Contriever), e o re-ranqueamento pelos três métodos contextuais descritos no Capítulo 4 (CPRR, BFS-TREE e RDPAC). Todos os métodos foram executados com janela de re-ranqueamento de 200 documentos (`window_size: 200`), tarefa `UDL` e formato de entrada `RK`.

A escolha da janela de 200 documentos não é meramente operacional: ela é decisiva para a viabilidade do experimento. Como as técnicas de aprendizado de variedades exigem que *cada documento do corpus* seja submetido como consulta (Seção 5.2), o número de consultas é da ordem do tamanho do corpus, N. Sem um limite, cada uma dessas N consultas produziria uma lista ranqueada com todos os demais documentos, gerando uma estrutura de relações de tamanho $N \times N$, isto é, com crescimento *quadrático* em relação ao tamanho do corpus, tanto no espaço de armazenamento quanto no custo de processamento das etapas subsequentes, sobretudo o re-ranqueamento no UDLF. Ao truncar cada lista a um teto fixo de 200 vizinhos, o volume total de dados passa a crescer como $200 \times N$, ou seja, *linearmente* com o tamanho do corpus. É esse teto que torna os cálculos do UDLF tratáveis e acelera o pipeline como um todo; ainda assim, os modelos de primeiro estágio apresentam tempos de execução distintos, que também aumentam com o tamanho do conjunto de dados, motivo pelo qual os experimentos foram conduzidos majoritariamente sobre as coleções de menor porte.

6.2.2 Definição do Parâmetro K

O parâmetro K é o principal hiperparâmetro comum entre os métodos UDLF: ele define o tamanho da vizinhança utilizada para construir a estrutura de correlação entre as listas ranqueadas. Como se mostra adiante (Seção 6.3.1), esse parâmetro pode determinar diretamente a qualidade do re-ranqueamento e seu valor ótimo varia de conjunto para conjunto, em associação com a densidade de relevantes e outras características do cenário de recuperação. Por isso, em vez de fixá-lo, K foi iterado sobre uma lista de candidatos, buscando o melhor cenário para cada conjunto de dados:

$$\{1, 3, 5, 10, 20, 30, 40, 50, 75\}$$

A escolha da melhor configuração segue uma convenção uniforme: para cada tripla (conjunto, modelo, método), seleciona-se o valor de K que maximiza a melhoria relativa em Precisão@20 sobre o *baseline* de recuperação, com empates resolvidos pelo menor K. A Precisão@20 foi adotada como critério por ser a métrica mais interpretável do ponto de vista do usuário (proporção de documentos relevantes visíveis entre os vinte primeiros). O comportamento de K é analisado em detalhe na Seção 6.3.

6.3 Resultados Quantitativos

Esta seção apresenta os resultados quantitativos da avaliação experimental. O desenho compreende $9 \times 4 \times 3 \times 9 = 972$ experimentos de re-ranqueamento (nove conjuntos de dados, quatro modelos, três métodos e nove valores de K), além dos 36 *baselines* de recuperação. Todas as configurações planejadas e todos os *baselines* foram efetivamente coletados, permitindo comparar as 36 combinações de conjunto de dados e recuperador. Os resultados completos são organizados por conjunto de dados, uma tabela para cada um, reportando MAP, Precisão (P) e Revocação (R) nos pontos de corte @20, @50 e @100.

Em cada tabela, os modelos de recuperação aparecem em blocos: uma linha de *baseline* (*base*), seguida de uma linha por método de re-ranqueamento, anotada com o valor de K selecionado para aquela combinação. O valor de K é escolhido pela maior melhoria relativa em Precisão@20 sobre o respectivo *baseline*, com empates resolvidos pelo menor K. Valores em negrito indicam melhoria sobre o *baseline* de recuperação. As Tabelas 6, 7, 8, 9 e 10 apresentam, respectivamente, os resultados de ArguAna, SciFact, NFCorpus, SciDocs e FiQA. As Tabelas 11, 12, 13 e 14 reúnem os resultados de WOS, BBC-News, HuffPost e Mental-Health. Essas tabelas reportam os valores absolutos nos três pontos de corte, enquanto a tabela de síntese explicita as variações relativas em MAP@20, P@20 e R@20. Todos os números são gerados automaticamente a partir dos dados de avaliação, sem edição manual.

Tabela 6 – Eficácia do re-ranqueamento em **ArguAna** em diferentes pontos de corte. O melhor K por modelo/método é escolhido pela maior melhoria em P@20; valores em negrito indicam melhoria sobre o *baseline* de recuperação.

	@20			@50			@100		
Método	MAP	P	R	MAP	P	R	MAP	P	R
BM25 (base)	0.2191	0.0418	0.8357	0.2211	0.0179	0.8962	0.2215	0.0093	0.9282
+ CPRR (K=10)	0.1887	**0.0462**	**0.9246**	0.1894	**0.0188**	**0.9417**	0.1894	**0.0094**	**0.9438**
+ BFS-TREE (K=50)	**0.2510**	**0.0463**	**0.9253**	**0.2517**	**0.0189**	**0.9445**	**0.2517**	**0.0094**	**0.9445**
+ RDPAC (K=30)	**0.2857**	**0.0464**	**0.9282**	**0.2863**	**0.0188**	**0.9424**	**0.2863**	**0.0094**	**0.9445**
MiniLM (base)	0.2988	0.0484	0.9680	0.2995	0.0198	0.9900	0.2996	0.0100	0.9957
+ CPRR (K=1)	**0.3086**	**0.0486**	**0.9723**	**0.3092**	**0.0198**	**0.9908**	**0.3093**	0.0100	0.9950
+ BFS-TREE (K=1)	**0.3083**	**0.0486**	**0.9723**	**0.3090**	**0.0198**	**0.9922**	**0.3090**	0.0099	0.9943
+ RDPAC (K=1)	**0.3097**	**0.0486**	**0.9716**	**0.3104**	**0.0198**	**0.9915**	**0.3104**	0.0100	0.9957
BGE (base)	0.3364	0.0487	0.9737	0.3370	0.0198	0.9908	0.3371	0.0100	0.9950
+ CPRR (K=10)	0.2615	**0.0489**	**0.9780**	0.2621	**0.0198**	**0.9922**	0.2621	0.0099	0.9936
+ BFS-TREE (K=30)	0.3268	**0.0490**	**0.9808**	0.3272	**0.0198**	**0.9915**	0.3272	0.0099	0.9936
+ RDPAC (K=1)	**0.3465**	**0.0487**	**0.9744**	**0.3472**	**0.0198**	**0.9915**	**0.3472**	0.0099	0.9943
Contriever (base)	0.2806	0.0482	0.9644	0.2816	0.0198	0.9900	0.2816	0.0099	0.9929
+ CPRR (K=20)	0.1907	**0.0488**	**0.9751**	0.1914	**0.0199**	**0.9950**	0.1914	**0.0100**	**0.9957**
+ BFS-TREE (K=20)	**0.2942**	**0.0491**	**0.9815**	**0.2948**	**0.0199**	**0.9957**	**0.2948**	**0.0100**	**0.9957**
+ RDPAC (K=10)	**0.3523**	**0.0492**	**0.9836**	**0.3527**	**0.0199**	**0.9950**	**0.3527**	**0.0100**	**0.9957**

Tabela 7 – Eficácia do re-ranqueamento em **SciFact** em diferentes pontos de corte. O melhor K por modelo/método é escolhido pela maior melhoria em P@20; valores em negrito indicam melhoria sobre o *baseline* de recuperação.

	@20			@50			@100		
Método	MAP	P	R	MAP	P	R	MAP	P	R
BM25 (base)	0.4812	0.0390	0.7083	0.4827	0.0167	0.7503	0.4834	0.0089	0.7896
+ CPRR (K=10)	0.3464	**0.0408**	**0.7311**	0.3487	**0.0177**	**0.7924**	0.3489	**0.0091**	**0.8088**
+ BFS-TREE (K=20)	**0.4853**	**0.0415**	**0.7411**	**0.4873**	**0.0178**	**0.7984**	**0.4876**	**0.0092**	**0.8171**
+ RDPAC (K=3)	0.4287	**0.0402**	**0.7194**	0.4302	**0.0171**	**0.7598**	0.4307	**0.0089**	**0.7904**
MiniLM (base)	0.6005	0.0475	0.8373	0.6025	0.0202	0.8920	0.6031	0.0105	0.9250
+ CPRR (K=5)	0.5619	**0.0490**	**0.8610**	0.5637	**0.0206**	**0.9077**	0.5641	**0.0106**	**0.9283**
+ BFS-TREE (K=20)	**0.6017**	**0.0490**	**0.8617**	**0.6032**	**0.0205**	**0.8983**	**0.6039**	**0.0107**	**0.9433**
+ RDPAC (K=5)	0.3932	**0.0482**	**0.8467**	0.3948	**0.0202**	0.8850	0.3953	0.0104	0.9167
BGE (base)	0.6791	0.0497	0.8753	0.6812	0.0211	0.9283	0.6815	0.0108	0.9533
+ CPRR (K=10)	0.5409	**0.0505**	**0.8867**	0.5418	0.0208	0.9133	0.5424	0.0108	0.9533
+ BFS-TREE (K=30)	**0.6851**	**0.0508**	**0.8933**	**0.6858**	0.0207	0.9083	**0.6863**	0.0107	0.9433
+ RDPAC (K=3)	0.5929	**0.0500**	**0.8767**	0.5947	0.0211	0.9283	0.5951	0.0108	0.9533
Contriever (base)	0.6128	0.0483	0.8602	0.6147	0.0205	0.9130	0.6149	0.0104	0.9220
+ CPRR (K=3)	0.6020	**0.0485**	0.8570	0.6041	**0.0207**	**0.9153**	0.6043	**0.0105**	**0.9227**
+ BFS-TREE (K=20)	**0.6151**	**0.0493**	**0.8650**	**0.6166**	**0.0206**	0.9083	**0.6168**	**0.0105**	**0.9250**
+ RDPAC (K=3)	0.5357	**0.0487**	0.8583	0.5378	**0.0207**	**0.9150**	0.5379	**0.0105**	**0.9233**

Tabela 8 – Eficácia do re-ranqueamento em **NFCorpus** em diferentes pontos de corte. O melhor K por modelo/método é escolhido pela maior melhoria em P@20; valores em negrito indicam melhoria sobre o *baseline* de recuperação.

	@20			@50			@100		
Método	MAP	P	R	MAP	P	R	MAP	P	R
BM25 (base)	0.0817	0.1094	0.1147	0.0865	0.0670	0.1438	0.0881	0.0412	0.1601
+ CPRR (K=30)	0.0590	**0.1211**	**0.1215**	0.0635	**0.0684**	**0.1451**	0.0651	**0.0415**	**0.1615**
+ BFS-TREE (K=40)	**0.0859**	**0.1195**	**0.1219**	**0.0906**	**0.0687**	**0.1493**	**0.0920**	**0.0414**	**0.1619**
+ RDPAC (K=10)	0.0707	**0.1152**	**0.1161**	0.0753	0.0666	**0.1472**	0.0765	0.0404	0.1597
MiniLM (base)	0.1226	0.1782	0.1889	0.1348	0.1153	0.2508	0.1424	0.0798	0.3115
+ CPRR (K=20)	0.1075	**0.1991**	**0.1910**	0.1232	**0.1268**	**0.2780**	0.1297	**0.0843**	**0.3265**
+ BFS-TREE (K=20)	**0.1353**	**0.2019**	**0.2071**	**0.1508**	**0.1276**	**0.2807**	**0.1579**	**0.0853**	**0.3293**
+ RDPAC (K=10)	0.1083	**0.1966**	**0.1992**	0.1240	**0.1252**	**0.2777**	0.1312	**0.0849**	**0.3309**
BGE (base)	0.1390	0.1895	0.2021	0.1505	0.1164	0.2591	0.1568	0.0775	0.3058
+ CPRR (K=20)	0.1240	**0.2088**	**0.2083**	0.1390	**0.1287**	**0.2767**	0.1453	**0.0840**	**0.3255**
+ BFS-TREE (K=20)	**0.1505**	**0.2096**	**0.2121**	**0.1669**	**0.1313**	**0.2792**	**0.1733**	**0.0843**	**0.3276**
+ RDPAC (K=10)	0.1084	**0.2003**	0.1954	0.1243	**0.1276**	**0.2732**	0.1310	**0.0833**	**0.3212**
Contriever (base)	0.1261	0.1701	0.1789	0.1380	0.1109	0.2413	0.1446	0.0759	0.2980
+ CPRR (K=20)	0.1048	**0.1875**	**0.1958**	0.1176	**0.1195**	**0.2595**	0.1242	**0.0799**	**0.3116**
+ BFS-TREE (K=30)	**0.1370**	**0.1904**	**0.2002**	**0.1507**	**0.1221**	**0.2655**	**0.1573**	**0.0814**	**0.3104**
+ RDPAC (K=10)	0.0956	**0.1820**	**0.1824**	0.1099	**0.1214**	**0.2588**	0.1164	**0.0814**	**0.3094**

Tabela 9 – Eficácia do re-ranqueamento em **SciDocs** em diferentes pontos de corte. O melhor K por modelo/método é escolhido pela maior melhoria em P@20; valores em negrito indicam melhoria sobre o *baseline* de recuperação.

	@20			@50			@100		
Método	MAP	P	R	MAP	P	R	MAP	P	R
BM25 (base)	0.0587	0.0326	0.1326	0.0622	0.0184	0.1875	0.0635	0.0114	0.2331
+ CPRR (K=20)	**0.0637**	**0.0390**	**0.1589**	**0.0677**	**0.0215**	**0.2190**	**0.0686**	**0.0123**	**0.2509**
+ BFS-TREE (K=30)	**0.0666**	**0.0381**	**0.1552**	**0.0711**	**0.0221**	**0.2253**	**0.0722**	**0.0127**	**0.2595**
+ RDPAC (K=5)	**0.0608**	**0.0377**	**0.1539**	**0.0644**	**0.0209**	**0.2134**	**0.0655**	**0.0124**	**0.2523**
MiniLM (base)	0.1432	0.0774	0.3137	0.1524	0.0419	0.4250	0.1559	0.0252	0.5101
+ CPRR (K=20)	0.1359	**0.0785**	**0.3185**	0.1459	**0.0430**	**0.4354**	0.1493	**0.0257**	**0.5198**
+ BFS-TREE (K=20)	**0.1487**	**0.0796**	**0.3228**	**0.1587**	**0.0432**	**0.4373**	**0.1619**	**0.0255**	**0.5161**
+ RDPAC (K=3)	0.1371	**0.0779**	**0.3161**	0.1461	**0.0422**	**0.4284**	0.1496	**0.0252**	**0.5109**
BGE (base)	0.1272	0.0685	0.2774	0.1349	0.0373	0.3786	0.1381	0.0228	0.4627
+ CPRR (K=20)	**0.1275**	**0.0734**	**0.2979**	**0.1361**	**0.0394**	**0.4005**	**0.1392**	**0.0237**	**0.4814**
+ BFS-TREE (K=20)	**0.1366**	**0.0742**	**0.3012**	**0.1454**	**0.0398**	**0.4037**	**0.1484**	**0.0237**	**0.4820**
+ RDPAC (K=10)	0.1071	**0.0734**	**0.2979**	0.1157	**0.0393**	**0.3992**	0.1188	**0.0238**	**0.4831**
Contriever (base)	0.1057	0.0580	0.2355	0.1120	0.0312	0.3166	0.1142	0.0186	0.3781
+ CPRR (K=20)	**0.1130**	**0.0654**	**0.2658**	**0.1188**	**0.0338**	**0.3433**	**0.1208**	**0.0197**	**0.3996**
+ BFS-TREE (K=20)	**0.1180**	**0.0652**	**0.2646**	**0.1244**	**0.0341**	**0.3460**	**0.1265**	**0.0199**	**0.4037**
+ RDPAC (K=10)	0.0962	**0.0627**	**0.2547**	0.1032	**0.0343**	**0.3476**	0.1052	**0.0199**	**0.4049**

Tabela 10 – Eficácia do re-ranqueamento em **FiQA** em diferentes pontos de corte. O melhor K por modelo/método é escolhido pela maior melhoria em P@20; valores em negrito indicam melhoria sobre o *baseline* de recuperação.

	@20			@50			@100		
Método	MAP	P	R	MAP	P	R	MAP	P	R
BM25 (base)	0.0939	0.0212	0.1950	0.0961	0.0114	0.2464	0.0970	0.0067	0.2908
+ CPRR (K=20)	0.0730	**0.0219**	0.1917	0.0760	**0.0120**	**0.2607**	0.0770	**0.0071**	**0.3068**
+ BFS-TREE (K=10)	**0.0957**	**0.0235**	**0.2091**	**0.0983**	**0.0120**	**0.2663**	**0.0993**	**0.0072**	**0.3097**
+ RDPAC (K=5)	0.0628	**0.0218**	0.1893	0.0655	**0.0115**	**0.2533**	0.0665	**0.0070**	**0.3033**
MiniLM (base)	0.2744	0.0576	0.4777	0.2804	0.0282	0.5796	0.2825	0.0163	0.6480
+ CPRR (K=3)	0.2685	**0.0583**	**0.4847**	0.2741	**0.0283**	**0.5827**	0.2762	**0.0163**	**0.6509**
+ BFS-TREE (K=40)	**0.2806**	**0.0590**	**0.4821**	**0.2867**	**0.0289**	**0.5802**	**0.2892**	**0.0168**	**0.6571**
+ RDPAC (K=3)	0.2657	**0.0580**	**0.4810**	0.2715	**0.0284**	**0.5807**	0.2736	**0.0164**	0.6480
BGE (base)	0.2915	0.0577	0.4695	0.2966	0.0280	0.5526	0.2985	0.0158	0.6228
+ CPRR (K=20)	0.2259	**0.0586**	**0.4702**	0.2314	**0.0286**	**0.5666**	0.2334	**0.0161**	**0.6406**
+ BFS-TREE (K=30)	**0.3008**	**0.0597**	**0.4851**	**0.3061**	**0.0289**	**0.5691**	**0.3081**	**0.0164**	**0.6490**
+ RDPAC (K=3)	0.2703	**0.0579**	**0.4712**	0.2755	**0.0281**	**0.5548**	0.2774	**0.0159**	**0.6260**
Contriever (base)	0.2421	0.0512	0.4350	0.2476	0.0256	0.5347	0.2495	0.0147	0.5981
+ CPRR (K=3)	0.2391	**0.0523**	**0.4447**	0.2443	**0.0258**	**0.5384**	0.2462	**0.0148**	**0.5995**
+ BFS-TREE (K=30)	**0.2470**	**0.0530**	**0.4377**	**0.2519**	**0.0258**	0.5216	**0.2538**	**0.0148**	0.5975
+ RDPAC (K=3)	0.2340	**0.0521**	**0.4449**	0.2391	**0.0257**	**0.5374**	0.2410	**0.0148**	0.5979

Tabela 11 – Eficácia do re-ranqueamento em **WOS** em diferentes pontos de corte. O melhor K por modelo/método é escolhido pela maior melhoria em P@20; valores em negrito indicam melhoria sobre o *baseline* de recuperação.

	@20			@50			@100		
Método	MAP	P	R	MAP	P	R	MAP	P	R
BM25 (base)	0.0171	0.3793	0.0220	0.0298	0.3079	0.0437	0.0439	0.2569	0.0719
+ CPRR (K=1)	0.0171	0.3792	0.0220	0.0298	0.3079	0.0437	0.0439	0.2568	0.0719
+ BFS-TREE (K=10)	0.0171	0.3792	0.0220	0.0298	0.3079	0.0437	0.0439	0.2568	0.0719
+ RDPAC (K=1)	0.0171	0.3792	0.0220	0.0298	0.3079	0.0437	0.0439	0.2568	0.0719
MiniLM (base)	0.0172	0.4164	0.0243	0.0378	0.3896	0.0563	0.0653	0.3572	0.1023
+ CPRR (K=75)	**0.0194**	**0.4313**	**0.0253**	**0.0426**	**0.4086**	**0.0592**	**0.0727**	**0.3755**	**0.1078**
+ BFS-TREE (K=75)	0.0166	0.3986	0.0234	0.0370	0.3844	0.0557	**0.0666**	**0.3657**	**0.1050**
+ RDPAC (K=1)	0.0159	0.4011	0.0235	0.0350	0.3779	0.0546	0.0611	0.3490	0.0999
BGE (base)	0.0170	0.4153	0.0242	0.0367	0.3860	0.0555	0.0626	0.3505	0.0995
+ CPRR (K=75)	**0.0198**	**0.4408**	**0.0258**	**0.0433**	**0.4157**	**0.0601**	**0.0729**	**0.3767**	**0.1075**
+ BFS-TREE (K=75)	0.0168	0.4048	0.0237	**0.0375**	**0.3891**	**0.0562**	**0.0663**	**0.3657**	**0.1044**
+ RDPAC (K=1)	0.0161	0.4064	0.0238	0.0351	0.3803	0.0547	0.0606	0.3480	0.0989
Contriever (base)	0.0146	0.3649	0.0213	0.0304	0.3335	0.0480	0.0503	0.2986	0.0851
+ CPRR (K=75)	**0.0190**	**0.4195**	**0.0246**	**0.0408**	**0.3929**	**0.0568**	**0.0668**	**0.3464**	**0.0989**
+ BFS-TREE (K=75)	**0.0163**	**0.3896**	**0.0229**	**0.0358**	**0.3710**	**0.0537**	**0.0612**	**0.3373**	**0.0964**
+ RDPAC (K=10)	**0.0159**	**0.3840**	**0.0226**	**0.0351**	**0.3671**	**0.0531**	**0.0604**	**0.3353**	**0.0958**

Tabela 12 – Eficácia do re-ranqueamento em **BBC-News** em diferentes pontos de corte. O melhor K por modelo/método é escolhido pela maior melhoria em P@20; valores em negrito indicam melhoria sobre o *baseline* de recuperação.

	@20			@50			@100		
Método	MAP	P	R	MAP	P	R	MAP	P	R
BM25 (base)	0.0291	0.3990	0.0179	0.0699	0.3730	0.0419	0.1231	0.3341	0.0753
+ CPRR (K=50)	**0.0322**	**0.4297**	**0.0193**	**0.0828**	**0.4215**	**0.0473**	**0.1521**	**0.3855**	**0.0864**
+ BFS-TREE (K=75)	**0.0319**	**0.4287**	**0.0193**	**0.0833**	**0.4245**	**0.0477**	**0.1561**	**0.3939**	**0.0884**
+ RDPAC (K=50)	**0.0313**	**0.4234**	**0.0190**	**0.0812**	**0.4185**	**0.0470**	**0.1510**	**0.3859**	**0.0867**
MiniLM (base)	0.0319	0.8574	0.0385	0.0836	0.8566	0.0960	0.1632	0.8302	0.1857
+ CPRR (K=75)	**0.0337**	**0.8826**	**0.0397**	**0.0901**	**0.8950**	**0.1006**	**0.1791**	**0.8777**	**0.1970**
+ BFS-TREE (K=75)	**0.0319**	**0.8589**	**0.0386**	**0.0855**	**0.8726**	**0.0980**	**0.1720**	**0.8641**	**0.1938**
+ RDPAC (K=1)	0.0316	0.8570	**0.0385**	0.0831	0.8557	0.0960	0.1619	0.8270	0.1852
BGE (base)	0.0321	0.8603	0.0386	0.0838	0.8567	0.0960	0.1630	0.8276	0.1850
+ CPRR (K=75)	**0.0345**	**0.8989**	**0.0404**	**0.0923**	**0.9094**	**0.1022**	**0.1826**	**0.8872**	**0.1990**
+ BFS-TREE (K=75)	**0.0331**	**0.8823**	**0.0397**	**0.0888**	**0.8924**	**0.1002**	**0.1774**	**0.8776**	**0.1968**
+ RDPAC (K=30)	**0.0330**	**0.8788**	**0.0395**	**0.0882**	**0.8887**	**0.0998**	**0.1764**	**0.8744**	**0.1961**
Contriever (base)	0.0325	0.8665	0.0390	0.0849	0.8601	0.0966	0.1644	0.8265	0.1854
+ CPRR (K=75)	**0.0351**	**0.9076**	**0.0409**	**0.0944**	**0.9218**	**0.1038**	**0.1873**	**0.8986**	**0.2021**
+ BFS-TREE (K=75)	**0.0337**	**0.8903**	**0.0401**	**0.0909**	**0.9047**	**0.1018**	**0.1818**	**0.8877**	**0.1997**
+ RDPAC (K=50)	**0.0336**	**0.8848**	**0.0398**	**0.0902**	**0.9005**	**0.1014**	**0.1801**	**0.8831**	**0.1987**

Tabela 13 – Eficácia do re-ranqueamento em **HuffPost** em diferentes pontos de corte. O melhor K por modelo/método é escolhido pela maior melhoria em P@20; valores em negrito indicam melhoria sobre o *baseline* de recuperação.

	@20			@50			@100		
Método	MAP	P	R	MAP	P	R	MAP	P	R
BM25 (base)	0.0011	0.1746	0.0022	0.0017	0.1571	0.0046	0.0023	0.1427	0.0081
+ CPRR (K=30)	**0.0012**	**0.1761**	**0.0023**	**0.0018**	**0.1596**	**0.0048**	**0.0025**	**0.1451**	**0.0083**
+ BFS-TREE (K=40)	**0.0011**	0.1740	**0.0023**	**0.0018**	**0.1605**	**0.0048**	**0.0025**	**0.1477**	**0.0084**
+ RDPAC (K=5)	**0.0011**	**0.1749**	**0.0022**	**0.0017**	**0.1589**	**0.0048**	**0.0024**	**0.1437**	**0.0082**
MiniLM (base)	0.0029	0.3237	0.0049	0.0058	0.3126	0.0114	0.0097	0.2982	0.0208
+ CPRR (K=75)	**0.0035**	**0.3355**	**0.0054**	**0.0070**	**0.3263**	**0.0126**	**0.0115**	**0.3107**	**0.0227**
+ BFS-TREE (K=75)	**0.0029**	0.3156	**0.0049**	**0.0061**	**0.3146**	**0.0119**	**0.0107**	**0.3082**	**0.0224**
+ RDPAC (K=5)	**0.0029**	0.3162	0.0049	**0.0062**	**0.3170**	**0.0119**	**0.0107**	**0.3072**	**0.0221**
BGE (base)	0.0029	0.3270	0.0050	0.0056	0.3131	0.0113	0.0093	0.2961	0.0205
+ CPRR (K=75)	**0.0038**	**0.3579**	**0.0059**	**0.0076**	**0.3450**	**0.0135**	**0.0122**	**0.3220**	**0.0236**
+ BFS-TREE (K=75)	**0.0031**	**0.3326**	**0.0053**	**0.0065**	**0.3296**	**0.0126**	**0.0111**	**0.3171**	**0.0230**
+ RDPAC (K=5)	**0.0031**	**0.3312**	**0.0052**	**0.0066**	**0.3295**	**0.0124**	**0.0108**	**0.3123**	**0.0222**
Contriever (base)	0.0034	0.3578	0.0057	0.0069	0.3478	0.0131	0.0119	0.3345	0.0242
+ CPRR (K=75)	**0.0042**	**0.3761**	**0.0063**	**0.0087**	**0.3666**	**0.0148**	**0.0145**	**0.3503**	**0.0267**
+ BFS-TREE (K=75)	**0.0035**	0.3554	**0.0059**	**0.0076**	**0.3543**	**0.0140**	**0.0134**	**0.3473**	**0.0263**
+ RDPAC (K=5)	**0.0035**	0.3543	**0.0058**	**0.0077**	**0.3576**	**0.0140**	**0.0134**	**0.3470**	**0.0259**

Tabela 14 – Eficácia do re-ranqueamento em **Mental-Health** em diferentes pontos de corte. O melhor K por modelo/método é escolhido pela maior melhoria em P@20; valores em negrito indicam melhoria sobre o *baseline* de recuperação.

	@20			@50			@100		
Método	MAP	P	R	MAP	P	R	MAP	P	R
BM25 (base)	0.0011	0.4553	0.0018	0.0025	0.4433	0.0043	0.0046	0.4282	0.0082
+ CPRR (K=75)	**0.0013**	**0.5137**	**0.0020**	**0.0031**	**0.4867**	**0.0047**	**0.0055**	**0.4560**	**0.0088**
+ BFS-TREE (K=75)	**0.0012**	**0.4845**	**0.0019**	**0.0028**	**0.4634**	**0.0045**	**0.0051**	**0.4411**	**0.0086**
+ RDPAC (K=50)	**0.0012**	**0.4861**	**0.0019**	**0.0028**	**0.4634**	**0.0045**	**0.0051**	**0.4392**	**0.0085**
MiniLM (base)	0.0015	0.5619	0.0022	0.0037	0.5608	0.0056	0.0072	0.5506	0.0109
+ CPRR (K=75)	**0.0016**	**0.5907**	**0.0024**	**0.0041**	**0.5916**	**0.0059**	**0.0080**	**0.5768**	**0.0114**
+ BFS-TREE (K=3)	**0.0015**	**0.5770**	**0.0023**	**0.0038**	**0.5754**	**0.0057**	**0.0074**	**0.5596**	**0.0110**
+ RDPAC (K=1)	**0.0015**	**0.5800**	**0.0023**	**0.0038**	**0.5729**	**0.0057**	**0.0074**	**0.5563**	**0.0110**
BGE (base)	0.0015	0.5544	0.0022	0.0036	0.5488	0.0055	0.0070	0.5356	0.0106
+ CPRR (K=75)	**0.0016**	**0.5900**	**0.0024**	**0.0041**	**0.5833**	**0.0058**	**0.0078**	**0.5629**	**0.0112**
+ BFS-TREE (K=50)	**0.0015**	**0.5739**	**0.0023**	**0.0038**	**0.5703**	**0.0057**	**0.0074**	**0.5553**	**0.0110**
+ RDPAC (K=75)	**0.0015**	**0.5757**	**0.0023**	**0.0038**	**0.5718**	**0.0057**	**0.0074**	**0.5551**	**0.0110**
Contriever (base)	0.0012	0.4772	0.0019	0.0030	0.4729	0.0048	0.0056	0.4632	0.0092
+ CPRR (K=75)	**0.0016**	**0.5715**	**0.0023**	**0.0039**	**0.5479**	**0.0055**	**0.0071**	**0.5116**	**0.0103**
+ BFS-TREE (K=75)	**0.0015**	**0.5492**	**0.0022**	**0.0035**	**0.5239**	**0.0053**	**0.0065**	**0.4932**	**0.0099**
+ RDPAC (K=50)	**0.0015**	**0.5565**	**0.0022**	**0.0036**	**0.5346**	**0.0053**	**0.0066**	**0.4999**	**0.0100**

Tabela 15 – Melhor configuração de re-ranqueamento por conjunto e modelo no corte @20 (selecionada pela maior melhoria relativa em P@20). Valores em negrito indicam melhoria sobre o *baseline*; entre parênteses, a variação relativa.

Conjunto	Modelo	Método (K)	MAP@20	P@20	R@20
ArguAna	BM25	(base)	0.2191	0.0418	0.8357
		RDPAC (30)	**0.2857** (+30.4%)	**0.0464** (+11.1%)	**0.9282** (+11.1%)
	MiniLM	(base)	0.2988	0.0484	0.9680
		CPRR (1)	**0.3086** (+3.3%)	**0.0486** (+0.4%)	**0.9723** (+0.4%)
	BGE	(base)	0.3364	0.0487	0.9737
		BFS-TREE (30)	0.3268 (-2.9%)	**0.0490** (+0.7%)	**0.9808** (+0.7%)
	Contriever	(base)	0.2806	0.0482	0.9644
		RDPAC (10)	**0.3523** (+25.5%)	**0.0492** (+2.0%)	**0.9836** (+2.0%)
SciFact	BM25	(base)	0.4812	0.0390	0.7083
		BFS-TREE (20)	**0.4853** (+0.8%)	**0.0415** (+6.4%)	**0.7411** (+4.6%)
	MiniLM	(base)	0.6005	0.0475	0.8373
		CPRR (5)	0.5619 (-6.4%)	**0.0490** (+3.2%)	**0.8610** (+2.8%)
	BGE	(base)	0.6791	0.0497	0.8753
		BFS-TREE (30)	**0.6851** (+0.9%)	**0.0508** (+2.3%)	**0.8933** (+2.1%)
	Contriever	(base)	0.6128	0.0483	0.8602
		BFS-TREE (20)	**0.6151** (+0.4%)	**0.0493** (+2.1%)	**0.8650** (+0.6%)
NFCorpus	BM25	(base)	0.0817	0.1094	0.1147
		CPRR (30)	0.0590 (-27.7%)	**0.1211** (+10.6%)	**0.1215** (+5.9%)
	MiniLM	(base)	0.1226	0.1782	0.1889
		BFS-TREE (20)	**0.1353** (+10.4%)	**0.2019** (+13.3%)	**0.2071** (+9.6%)
	BGE	(base)	0.1390	0.1895	0.2021
		BFS-TREE (20)	**0.1505** (+8.3%)	**0.2096** (+10.6%)	**0.2121** (+4.9%)
	Contriever	(base)	0.1261	0.1701	0.1789
		BFS-TREE (30)	**0.1370** (+8.7%)	**0.1904** (+11.9%)	**0.2002** (+11.9%)
SciDocs	BM25	(base)	0.0587	0.0326	0.1326
		CPRR (20)	**0.0637** (+8.5%)	**0.0390** (+19.7%)	**0.1589** (+19.8%)
	MiniLM	(base)	0.1432	0.0774	0.3137
		BFS-TREE (20)	**0.1487** (+3.8%)	**0.0796** (+2.8%)	**0.3228** (+2.9%)
	BGE	(base)	0.1272	0.0685	0.2774
		BFS-TREE (20)	**0.1366** (+7.4%)	**0.0742** (+8.4%)	**0.3012** (+8.6%)
	Contriever	(base)	0.1057	0.0580	0.2355
		CPRR (20)	**0.1130** (+6.9%)	**0.0654** (+12.7%)	**0.2658** (+12.9%)
FiQA	BM25	(base)	0.0939	0.0212	0.1950
		BFS-TREE (10)	**0.0957** (+1.9%)	**0.0235** (+10.5%)	**0.2091** (+7.2%)
	MiniLM	(base)	0.2744	0.0576	0.4777
		BFS-TREE (40)	**0.2806** (+2.2%)	**0.0590** (+2.4%)	**0.4821** (+0.9%)
	BGE	(base)	0.2915	0.0577	0.4695
		BFS-TREE (30)	**0.3008** (+3.2%)	**0.0597** (+3.5%)	**0.4851** (+3.3%)
	Contriever	(base)	0.2421	0.0512	0.4350

continua na próxima página

Tabela 15 – continuação

Conjunto	Modelo	Método (K)	MAP@20	P@20	R@20
		BFS-TREE (30)	**0.2470** (+2.0%)	**0.0530** (+3.5%)	**0.4377** (+0.6%)
WOS	BM25	(base)	0.0171	0.3793	0.0220
		BFS-TREE (10)	0.0171 (-0.0%)	0.3792 (-0.0%)	0.0220 (-0.0%)
	MiniLM	(base)	0.0172	0.4164	0.0243
		CPRR (75)	**0.0194** (+13.0%)	**0.4313** (+3.6%)	**0.0253** (+3.8%)
	BGE	(base)	0.0170	0.4153	0.0242
		CPRR (75)	**0.0198** (+17.0%)	**0.4408** (+6.1%)	**0.0258** (+6.5%)
	Contriever	(base)	0.0146	0.3649	0.0213
		CPRR (75)	**0.0190** (+30.2%)	**0.4195** (+15.0%)	**0.0246** (+15.6%)
BBC-News	BM25	(base)	0.0291	0.3990	0.0179
		CPRR (50)	**0.0322** (+10.7%)	**0.4297** (+7.7%)	**0.0193** (+7.6%)
	MiniLM	(base)	0.0319	0.8574	0.0385
		CPRR (75)	**0.0337** (+5.8%)	**0.8826** (+2.9%)	**0.0397** (+3.1%)
	BGE	(base)	0.0321	0.8603	0.0386
		CPRR (75)	**0.0345** (+7.5%)	**0.8989** (+4.5%)	**0.0404** (+4.8%)
	Contriever	(base)	0.0325	0.8665	0.0390
		CPRR (75)	**0.0351** (+7.9%)	**0.9076** (+4.7%)	**0.0409** (+4.9%)
HuffPost	BM25	(base)	0.0011	0.1746	0.0022
		CPRR (30)	**0.0012** (+8.0%)	**0.1761** (+0.8%)	**0.0023** (+4.2%)
	MiniLM	(base)	0.0029	0.3237	0.0049
		CPRR (75)	**0.0035** (+21.8%)	**0.3355** (+3.6%)	**0.0054** (+9.6%)
	BGE	(base)	0.0029	0.3270	0.0050
		CPRR (75)	**0.0038** (+32.9%)	**0.3579** (+9.5%)	**0.0059** (+17.8%)
	Contriever	(base)	0.0034	0.3578	0.0057
		CPRR (75)	**0.0042** (+24.6%)	**0.3761** (+5.1%)	**0.0063** (+11.5%)
Mental-Health	BM25	(base)	0.0011	0.4553	0.0018
		CPRR (75)	**0.0013** (+24.7%)	**0.5137** (+12.8%)	**0.0020** (+13.7%)
	MiniLM	(base)	0.0015	0.5619	0.0022
		CPRR (75)	**0.0016** (+11.0%)	**0.5907** (+5.1%)	**0.0024** (+5.2%)
	BGE	(base)	0.0015	0.5544	0.0022
		CPRR (75)	**0.0016** (+12.6%)	**0.5900** (+6.4%)	**0.0024** (+6.5%)
	Contriever	(base)	0.0012	0.4772	0.0019
		CPRR (75)	**0.0016** (+33.5%)	**0.5715** (+19.8%)	**0.0023** (+19.0%)

De modo geral, os resultados indicam que o re-ranqueamento contextual não supervisionado melhora a qualidade da recuperação na ampla maioria das combinações de conjunto de dados e modelo, sobretudo em Precisão@20. A Tabela 15 consolida essa visão, reunindo, para cada conjunto e modelo, a melhor configuração encontrada e sua variação relativa sobre o *baseline*. Os ganhos manifestam-se tanto sobre o *baseline* esparso (BM25) quanto sobre os modelos densos, sugerindo que o sinal contextual capturado pela

estrutura de vizinhança é, em boa medida, complementar à informação já codificada pela representação de primeiro estágio. Entre os métodos, o BFS-TREE apresenta o comportamento mais consistente nos conjuntos BEIR, de relevância esparsa, enquanto o CPRR se destaca nos conjuntos baseados em categoria, de vizinhanças densas; o RDPAC mostra-se competitivo em casos específicos, com destaque para o ganho expressivo em MAP sobre o *baseline* BM25 em ArguAna.

Observa-se, contudo, heterogeneidade na magnitude dos ganhos. Em vários casos, melhorias em Precisão são acompanhadas de redução em MAP, indicando que o reranqueamento pode redistribuir a massa de relevância entre diferentes pontos de corte: ao trazer documentos relevantes para dentro do corte @20, o método por vezes reordena o topo de maneira que baixa a precisão média ao longo das posições. Esse efeito é mais evidente no SciFact, em que o CPRR e o RDPAC elevam a Precisão@20 sobre o *baseline* enquanto reduzem o MAP de forma expressiva, por exemplo, sobre o MiniLM, o RDPAC ($K = 5$) melhora a Precisão@20, mas o MAP cai de 0,60 para 0,39. Esse comportamento é coerente com a estrutura do SciFact, que possui em média apenas 1,13 documento relevante por consulta: com tão poucos relevantes, o MAP é altamente sensível à posição exata desse único item, de modo que qualquer reordenação no topo tem impacto desproporcional na métrica. Essa variação, relacionada às características de cada conjunto, em especial à densidade de documentos relevantes por consulta (Tabela 5), é retomada na discussão da conclusão.

6.3.1 Sensibilidade ao Parâmetro K

Como o parâmetro K é o principal hiperparâmetro variado neste estudo, investigou-se como o desempenho muda em função de seu valor. A Figura 30 apresenta, para cada método e valor de K, a média aritmética das melhorias relativas em Precisão@20 observadas nas 36 combinações entre nove conjuntos de dados e quatro modelos de recuperação. A Tabela 16 registra os mesmos resultados numericamente.

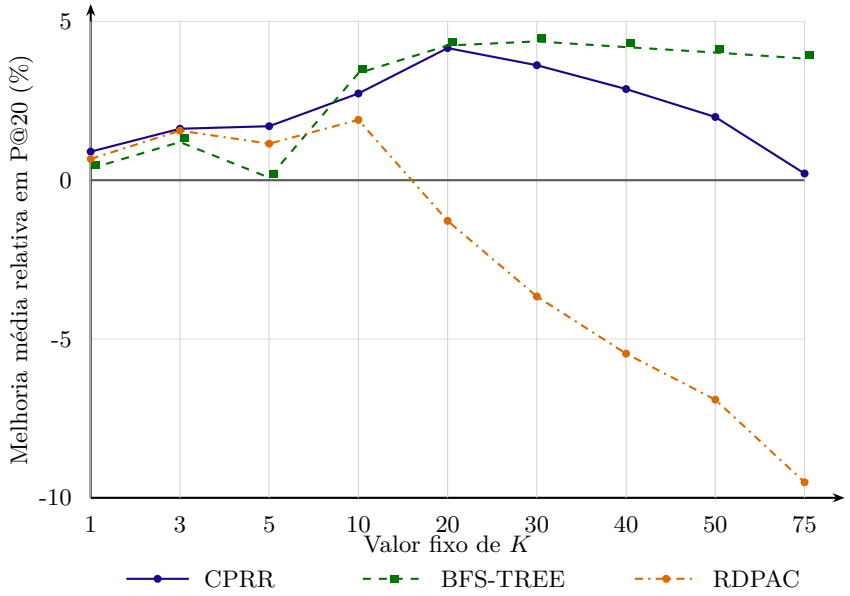


Figura 30 – Melhoria média relativa em Precisão@20 sobre o *baseline* para cada valor fixo de K. Cada ponto agrega as combinações disponíveis de conjunto de dados e modelo de recuperação; por isso, o gráfico descreve uma tendência global e não o comportamento de uma coleção específica.

Tabela 16 – Melhoria média relativa em Precisão@20 (%) sobre o *baseline* para cada valor fixo de K, agregada sobre as combinações de conjunto de dados e modelo de recuperação disponíveis.

K	1	3	5	10	20	30	40	50	75
CPRR	0.90	1.62	1.70	2.73	**4.16**	3.62	2.87	1.99	0.21
BFS-TREE	0.37	1.20	0.08	3.37	4.24	**4.37**	4.19	4.02	3.83
RDPAC	0.67	1.56	1.15	**1.90**	-1.28	-3.66	-5.46	-6.91	-9.51

Os três métodos apresentam seus melhores resultados médios em valores intermediários. O CPRR atinge a maior média em $K = 20$ (+4,16%), com mediana de +2,86% e ganho em 27 das 36 combinações. Para o BFS-TREE, $K = 30$ produz média de +4,37%, mediana de +2,45% e ganho em 29 combinações. O RDPAC alcança sua melhor média em $K = 10$ (+1,90%), embora a mediana seja mais modesta (+0,46%) e haja ganho em 19 combinações. Os resultados mostram que um valor fixo intermediário pode oferecer um ponto de partida razoável, sobretudo para CPRR e BFS-TREE, mas a dispersão entre cenários não sustenta a adoção de um K universal.

A média agregada, contudo, oculta diferenças relevantes entre os conjuntos. A Figura 31 relaciona a densidade média de relevância (documentos relevantes por consulta, Rel/Q) à mediana do melhor K em cada conjunto. Essa mediana resume as 12 combinações de quatro modelos de recuperação e três métodos contextuais, evitando que uma única configuração determine a leitura do conjunto. No recorte analisado, observa-se uma associação monotônica forte entre as duas variáveis ($\rho_s \approx 0{,}89$), mas essa estatística é descritiva e deve ser lida com cautela, pois envolve apenas nove conjuntos de dados.

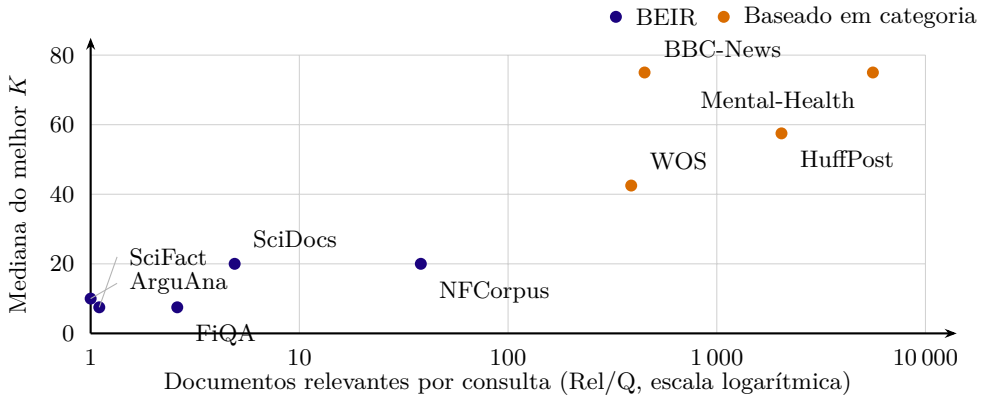


Figura 31 – Relação entre a densidade média de relevância (documentos relevantes por consulta, Rel/Q) e a mediana do melhor valor de K em cada conjunto. A mediana resume as 12 combinações de modelo de recuperação e método contextual por conjunto. O eixo horizontal está em escala logarítmica. Os pontos azuis representam conjuntos BEIR, avaliados por julgamentos de relevância oficiais, e os pontos laranja representam conjuntos com relevância definida por categoria.

Nos extremos, o padrão é compatível com a intuição do método: ArguAna, com um documento relevante por consulta, apresenta mediana do melhor K igual a 10; BBC-News e Mental-Health, com centenas ou milhares de relevantes por consulta, atingem mediana igual a 75. Os conjuntos BEIR de densidade intermediária, como SciDocs e NFCorpus, concentram-se em $K = 20$. A tendência, porém, não é estrita: HuffPost, apesar de mais denso que BBC-News, apresenta mediana de 57,5. Além da densidade de relevantes, a heterogeneidade temática, o recuperador de primeiro estágio e o próprio método contextual também podem afetar a configuração mais favorável.

Essa análise sugere que K controla uma escala de contexto que deve ser compatível com a estrutura local de cada coleção: vizinhanças maiores podem oferecer evidência adicional quando há grupos relevantes extensos, mas também podem introduzir itens pouco relacionados. Como os conjuntos BEIR usam julgamentos explícitos de relevância e os conjuntos baseados em categoria definem relevância pelo compartilhamento de rótulo, a relação observada não deve ser interpretada como uma regra causal ou como critério suficiente para escolher K em novos conjuntos. Ela fornece, antes, uma hipótese útil para orientar avaliações posteriores.

6.3.2 Comparação com Cross-Encoder Supervisionado

A fim de posicionar a abordagem não supervisionada frente ao estado da arte supervisionado, realizou-se uma comparação ilustrativa com um *cross-encoder* (`ms-marco-MiniLM-L-6-v2`), aplicado sobre a mesma janela de 200 documentos, nos conjuntos SciFact, SciDocs e NFCorpus. A Tabela 17 resume os resultados em Precisão@20, considerando o

BM25 e o melhor recuperador denso de cada conjunto de dados.

Tabela 17 – Comparação ilustrativa entre BM25, o melhor recuperador denso de cada conjunto, um *cross-encoder* supervisionado e o melhor método UDLF, em Precisão@20. Em caso de empate entre modelos densos, reporta-se um único representante.

Conjunto	Modelo	*Baseline*	Cross-Encoder	Melhor UDLF
SciFact	BM25	0.0390	0.0452 (+15.8%)	0.0415 (+6.4%)
	BGE	0.0497	0.0483 (−2.7%)	0.0508 (+2.3%)
SciDocs	BM25	0.0325	0.0450 (+38.1%)	0.0390 (+19.7%)
	MiniLM	0.0775	0.0626 (−19.2%)	0.0796 (+2.8%)
NFCorpus	BM25	0.1094	0.1260 (+15.1%)	0.1211 (+10.6%)
	BGE	0.1895	0.1803 (−4.8%)	0.2096 (+10.6%)

O padrão é consistente nos três conjuntos: o *cross-encoder* apresenta ganhos expressivos sobre o *baseline* lexical BM25, em que a interação fina entre consulta e documento corrige listas iniciais mais ruidosas. Sobre os melhores recuperadores densos, mais fortes, porém, o *cross-encoder* tende a estagnar ou a degradar o desempenho, ao passo que o re-ranqueamento contextual melhora as listas desses recuperadores sem utilizar dados rotulados. Reforça-se que esta comparação é ilustrativa e não constitui um estudo comparativo extensivo; seu propósito é contextualizar o comportamento da técnica, evidenciando que a abordagem não supervisionada é especialmente útil justamente nos cenários em que o *cross-encoder* é menos eficaz.

6.3.3 Comparação com Reciprocal Rank Fusion

Como complemento, o re-ranqueamento contextual foi comparado a uma estratégia popular de fusão, a *Reciprocal Rank Fusion* (RRF, $k = 60$), sobre o conjunto SciDocs. A Tabela 18 apresenta os resultados em Precisão@20.

Tabela 18 – Comparação ilustrativa entre configurações de fusão por *Reciprocal Rank Fusion* (RRF, $k = 60$) e o melhor método UDLF, em Precisão@20, sobre o conjunto SciDocs. Entre as combinações pareadas, reporta-se a de melhor desempenho e, separadamente, a combinação do melhor recuperador denso com o BM25.

Configuração	P@20
Melhor *baseline* de modelo único (MiniLM)	0.07745
RRF: MiniLM + BGE (melhor par)	0.07715
RRF: os quatro modelos	0.07365
RRF: BM25 + MiniLM	0.05905
Melhor UDLF (MiniLM, BFS-TREE, $K = 20$)	**0.07965**

Foram avaliadas todas as combinações pareadas entre os quatro recuperadores, além da fusão conjunta. Nenhuma supera o melhor *baseline* de modelo único: o melhor par, MiniLM e BGE, alcança 0.07715, resultado praticamente coincidente com o MiniLM isolado (0.07745), enquanto a fusão dos quatro modelos obtém 0.07365. A combinação do MiniLM com o BM25, cujo *baseline* é consideravelmente mais fraco, reduz a P@20 para 0.05905. Em contraste, o melhor resultado do re-ranqueamento contextual alcança 0.07965, uma melhoria de 2.8% sobre o MiniLM. Como a comparação está restrita ao SciDocs e ao valor $k = 60$ da RRF, ela não permite concluir que o re-ranqueamento contextual seja, em geral, superior à fusão de listas ranqueadas. Seu propósito é mostrar que, neste caso, explorar a estrutura de vizinhança de uma única lista foi ligeiramente mais favorável do que combinar as listas disponíveis. Cabe ainda distinguir essa comparação da fusão contextual oferecida pelo próprio UDLF: embora o *framework* também permita fundir listas por métodos contextuais, essa possibilidade não foi avaliada no presente trabalho.

6.4 Análise Qualitativa

Além das métricas agregadas, uma análise qualitativa permite compreender de forma mais intuitiva como o re-ranqueamento contextual reorganiza os resultados recuperados. Para isso, são exploradas duas perspectivas complementares: a projeção espacial dos documentos por meio da técnica UMAP e a inspeção direta das listas ranqueadas produzidas para consultas individuais.

Na primeira perspectiva, os documentos do corpus são projetados em duas dimensões utilizando UMAP, preservando, tanto quanto possível, a estrutura de vizinhança do espaço vetorial original. As Figuras 32 e 33 apresentam quatro consultas de cada coleção, uma por linha. Em cada par de painéis, o lado esquerdo mostra os 20 primeiros documentos da lista original e o lado direito mostra os 20 primeiros após a aplicação do CPRR com $K = 75$; portanto, K é o tamanho da vizinhança utilizada pelo método, e não a quantidade de pontos destacados. Os pontos pequenos e sem contorno representam o corpus projetado e são coloridos por categoria, os círculos maiores com contorno identificam os documentos recuperados no corte @20 e a estrela amarela marca a consulta. O valor de P@20 acima de cada painel resume a proporção de documentos recuperados que pertencem à categoria da consulta. Antes da aplicação do método, os resultados destacados aparecem distribuídos entre diferentes agrupamentos semânticos; depois do re-ranqueamento, tendem a se concentrar no agrupamento da consulta, comportamento compatível com o uso da estrutura de variedade para favorecer documentos semanticamente relacionados.

Essa reorganização espacial fornece uma evidência visual do funcionamento dos métodos reunidos no UDLF. Embora a projeção UMAP reduza um espaço de alta dimensionalidade para apenas duas dimensões e, portanto, não preserve perfeitamente todas

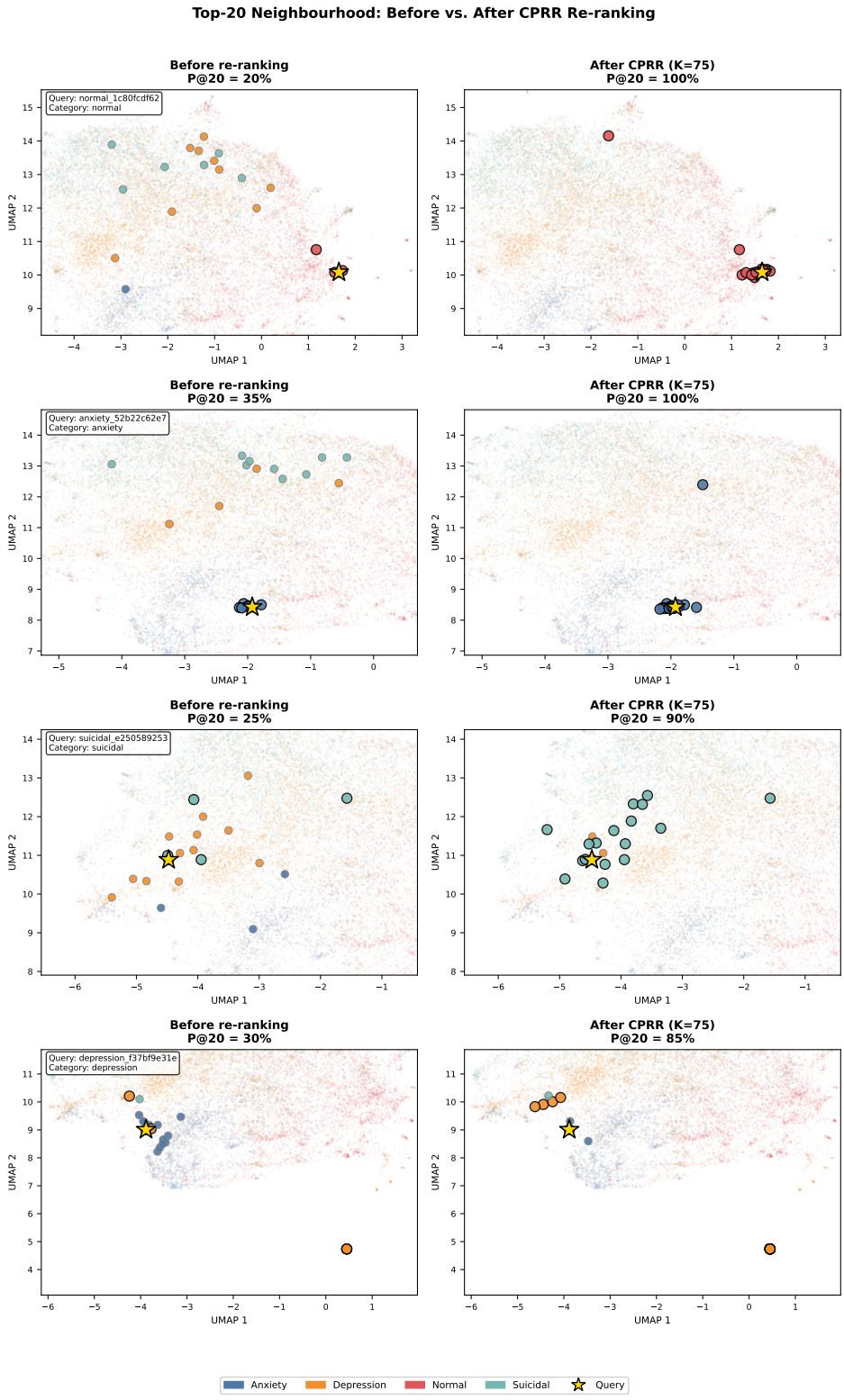


Figura 32 – Projeção UMAP de quatro consultas do conjunto Mental Health. Cada linha compara a lista original (à esquerda) com o resultado do CPRR com $K = 75$ (à direita). Os pontos pequenos representam o corpus e suas cores indicam as categorias; os círculos maiores com contorno são os documentos do corte @20, e a estrela amarela representa a consulta. O aumento de P@20 e a maior concentração de círculos da categoria correta no painel direito ilustram o efeito do re-ranqueamento nos casos selecionados.

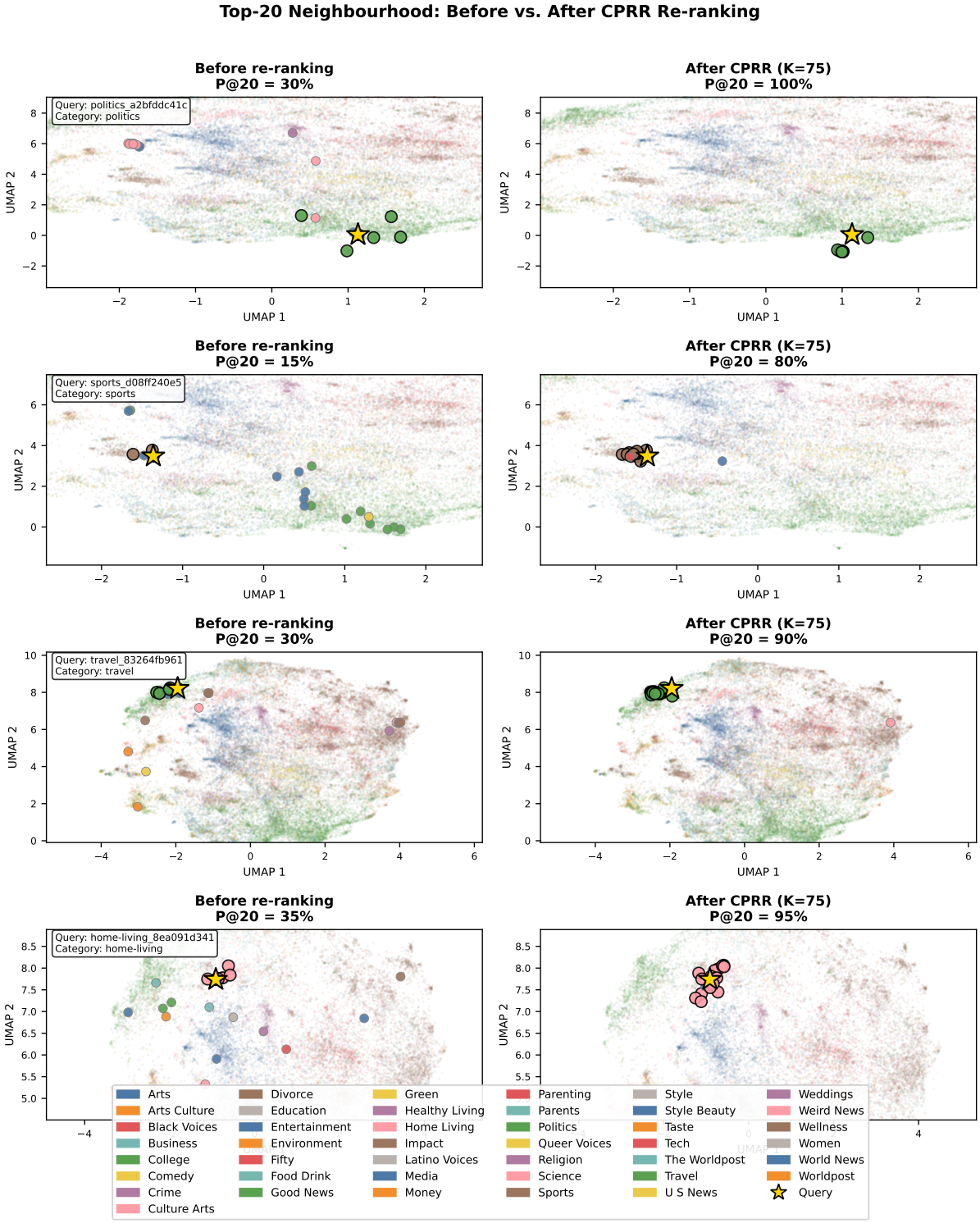


Figura 33 – Projeção UMAP de quatro consultas do conjunto HuffPost. Cada linha compara a lista original (à esquerda) com o resultado do CPRR com $K = 75$ (à direita). Os pontos pequenos representam o corpus e suas cores indicam as categorias; os círculos maiores com contorno são os documentos do corte @20, e a estrela amarela representa a consulta. O aumento de P@20 e a maior concentração de círculos da categoria correta no painel direito ilustram o efeito do re-ranqueamento nos casos selecionados.

as distâncias originais, ela permite observar claramente a tendência de concentração dos documentos relevantes após o re-ranqueamento.

A segunda perspectiva consiste na inspeção das listas ranqueadas para consultas individuais. As Figuras 34 e 35 ilustram exemplos concretos do efeito do re-ranqueamento contextual sobre uma consulta específica. Nessas visualizações, cada documento é representado por uma cor correspondente à sua categoria, permitindo identificar facilmente quais documentos foram promovidos ou rebaixados na lista ranqueada. Observa-se que documentos pertencentes à categoria correta tendem a ocupar posições superiores após o processamento, enquanto documentos de categorias distintas são deslocados para posições inferiores, reduzindo a dispersão temática entre os primeiros resultados. Os casos apresentados dão luz ao efeito prático do aumento de métricas como Precisão@K e MAP.

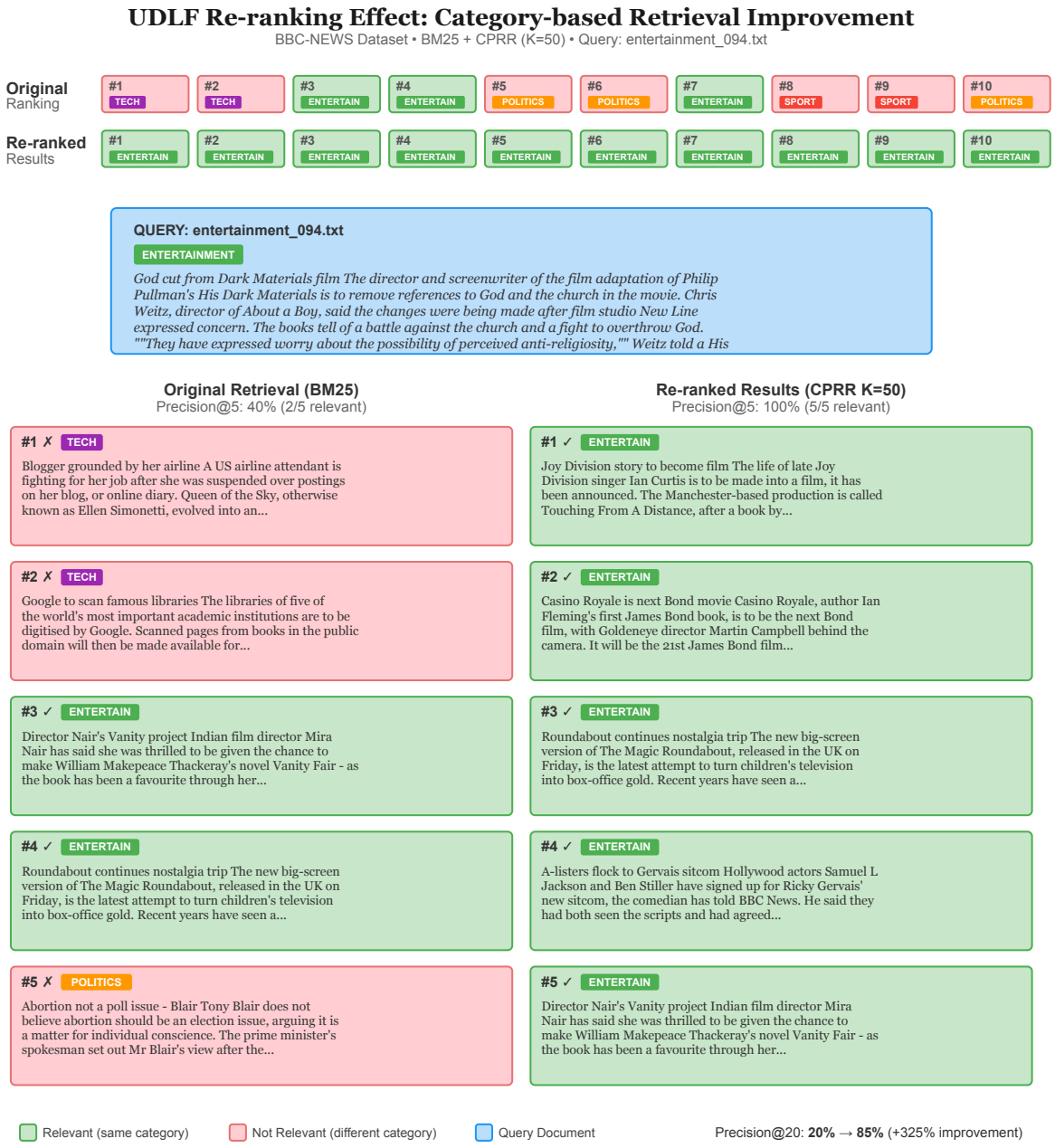


Figura 34 – Exemplo de re-ranqueamento contextual para uma consulta do conjunto BBC News. Cada documento é colorido de acordo com sua categoria, permitindo visualizar as alterações de posição promovidas pelo método CPRR.

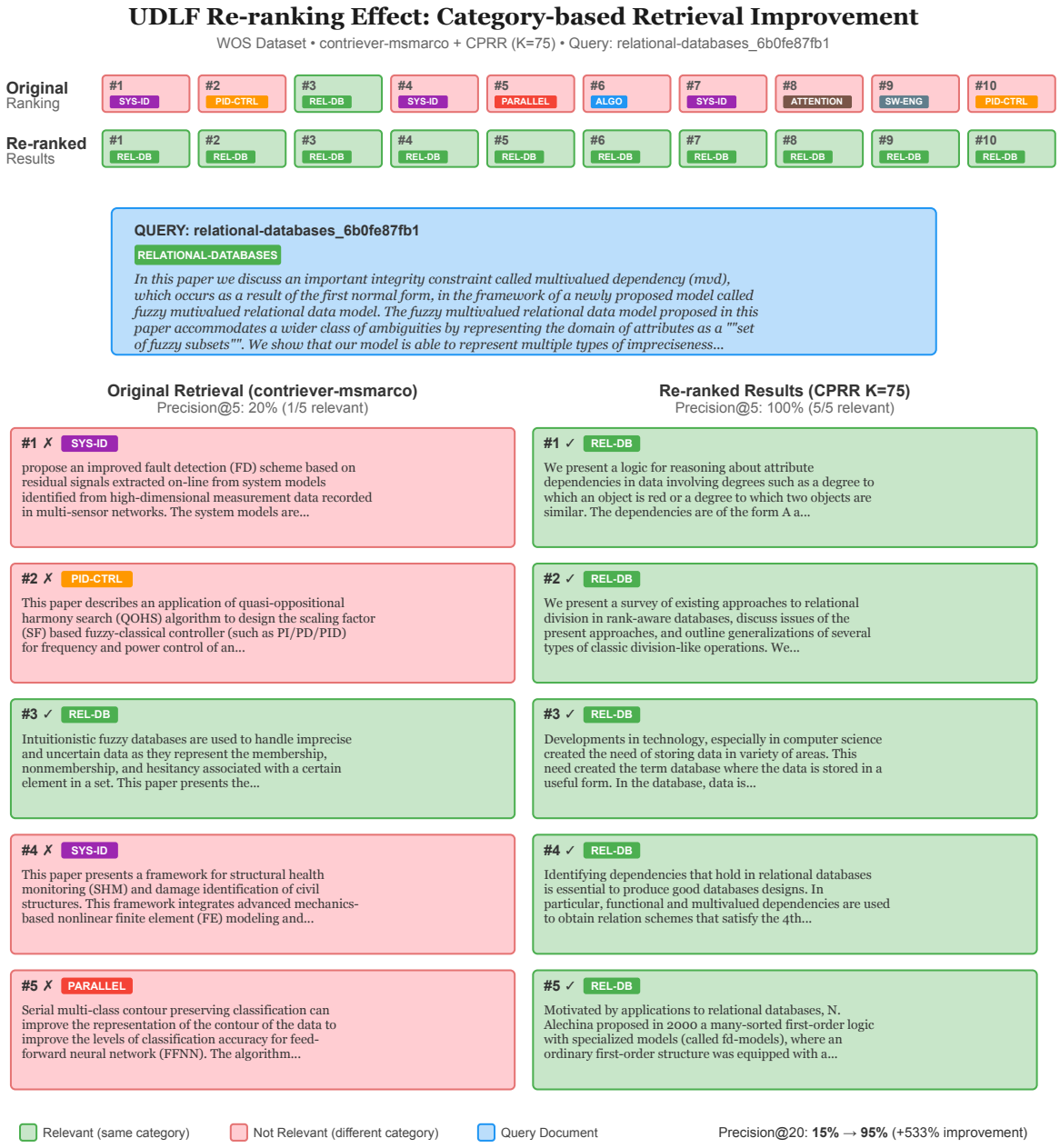


Figura 35 – Exemplo de re-ranqueamento contextual para uma consulta do conjunto WOS. Cada documento é colorido de acordo com sua categoria, permitindo visualizar as alterações de posição promovidas pelo método CPRR.

6.5 Discussão e Síntese dos Resultados

Os experimentos oferecem evidência de que o re-ranqueamento contextual baseado em informações de ranqueamento pode ser transferido para a recuperação textual e acrescentar informação aos sinais produzidos no primeiro estágio. Como o UDLF recebe somente posições ordinais, as mesmas operações foram aplicadas a listas ranqueadas provenientes do BM25 e de três recuperadores densos, sem treinamento adicional ou rótulos utilizados pelo método de re-ranqueamento. Esse resultado sustenta a hipótese central no escopo avaliado: relações recorrentes de vizinhança podem revelar informação complementar à comparação direta entre consulta e documento. Ao mesmo tempo, os dados não estabelecem que os métodos sejam universalmente independentes da modalidade, mas demonstram sua viabilidade em nove coleções textuais com características distintas.

6.5.1 Síntese dos Achados

O desenho experimental compreende 36 combinações entre conjunto de dados e recuperador, resultantes de nove conjuntos avaliados com quatro modelos de recuperação. Em 35 delas, a melhor configuração selecionada apresenta aumento de Precisão@20. A exceção foi o WOS com BM25, no qual a métrica variou de 0,3793 para 0,3792, uma redução residual inferior a 0,01%. Essa cobertura indica que existe uma configuração contextual favorável na ampla maioria dos cenários estudados. A leitura exige, contudo, uma ressalva: método e K foram selecionados retrospectivamente pela maior melhoria em Precisão@20. Portanto, a síntese descreve o potencial observado sob a configuração mais favorável de cada cenário, e não o desempenho garantido de um método fixo ou de um valor universal de K.

Os maiores ganhos ajudam a dimensionar esse potencial, mas não resumem sozinhos o comportamento dos métodos. No Mental-Health com Contriever e CPRR ($K = 75$), a Precisão@20 cresce 19,8% e o MAP@20, 33,5%, os maiores ganhos relativos observados nessas métricas entre as configurações selecionadas. Em contrapartida, algumas configurações escolhidas por Precisão@20 reduzem MAP, como no NFCorpus com BM25 e no SciFact com MiniLM. O re-ranqueamento pode, assim, aumentar a presença de relevantes nas primeiras vinte posições sem preservar a mesma qualidade de ordenação em todas as posições relevantes. A contribuição empírica é mais regular para Precisão@20, métrica usada na seleção, do que para MAP.

Também não se observa um único método dominante em todos os tipos de coleção. Entre as configurações resumidas na Tabela 15, o CPRR é selecionado em 20 das 36 combinações, com presença marcante nos conjuntos baseados em categoria; o BFS-TREE é selecionado em 14, sobretudo nos conjuntos BEIR; e o RDPAC aparece como melhor alternativa em dois casos, ambos no ArguAna. Sobre o BM25, produz o maior ganho de

MAP observado; sobre o Contriever, aumenta a Precisão@20 em 2,0% e o MAP@20 em 25,5%. Essas frequências são descritivas e dependem do critério de seleção adotado, mas sugerem que os métodos respondem de maneira diferente à densidade de relevância e à organização local das listas ranqueadas.

A análise do parâmetro K reforça essa dependência do cenário. Em média, o CPRR apresenta melhor comportamento em torno de $K = 20$, e o BFS-TREE em torno de $K = 30$, enquanto coleções com grupos relevantes extensos frequentemente favorecem vizinhanças maiores. A associação entre documentos relevantes por consulta e a mediana do melhor K é forte no conjunto de nove coleções, mas coincide parcialmente com a separação entre os conjuntos BEIR, com julgamentos individuais de relevância, e os conjuntos cuja relevância é definida por categoria. Por isso, o resultado fornece uma hipótese para seleção adaptativa de K, e não uma regra causal ou diretamente transferível para novas coleções.

As comparações adicionais posicionam a proposta, mas têm alcance mais restrito. Nos três conjuntos avaliados com o *cross-encoder*, o modelo supervisionado melhora o BM25, porém não melhora o melhor recuperador denso, enquanto o UDLF apresenta ganhos sobre ambos os tipos de *baseline*. No SciDocs, o re-ranqueamento contextual também supera as configurações de RRF avaliadas. Esses resultados mostram situações em que o sinal contextual é útil, mas a quantidade de conjuntos e configurações não permite afirmar superioridade geral sobre *cross-encoders* ou métodos de fusão.

Por fim, as projeções UMAP e as listas individuais tornam visível o mecanismo sugerido pelas métricas: documentos relevantes são promovidos quando participam de vizinhanças tematicamente mais coerentes. Essa evidência é complementar, não independente. A projeção reduz o espaço original a duas dimensões, e os exemplos selecionados não representam, isoladamente, a distribuição completa das consultas. Sua função é ilustrar uma interpretação compatível com os resultados quantitativos.

6.5.2 Limitações da Avaliação

A principal limitação diz respeito à seleção de hiperparâmetros. O melhor K e, nas tabelas de síntese, o melhor método foram determinados pela Precisão@20 observada nos próprios dados de avaliação, sem uma partição separada de validação. Os valores reportados caracterizam, portanto, o melhor desempenho encontrado na varredura experimental. Em uma aplicação real, na qual os julgamentos de relevância do novo conjunto não estariam disponíveis, seria necessário selecionar K por validação, por uma heurística não supervisionada ou por uma configuração previamente fixada.

A composição dos conjuntos também delimita a generalização. Os cinco conjuntos BEIR foram escolhidos entre os menores da coleção, permitindo a execução do protocolo completo, mas não avaliando diretamente o comportamento em corpora de maior escala. Nos

quatro conjuntos adicionais, a relevância é definida pelo compartilhamento de categoria; essa formulação viabiliza grande número de consultas e relevantes, porém difere dos julgamentos individuais do BEIR e pode favorecer métodos orientados à formação de vizinhanças temáticas. A relação observada entre densidade de relevância e K deve ser lida à luz dessa diferença de construção.

Há ainda limitações técnicas e estatísticas. O protocolo exige a geração de uma lista de vizinhos para cada documento do corpus, além das consultas originalmente fornecidas pelo conjunto. A janela de 200 documentos limita o volume armazenado a, no máximo, $200N$ relações para um corpus com N documentos, evitando a materialização de listas completas de tamanho quadrático. Ainda assim, o custo total inclui a codificação e a recuperação de primeiro estágio, a conversão e persistência dessas listas e o processamento contextual realizado pelo UDLF. Nesta última etapa, valores maiores de K ampliam a vizinhança examinada, enquanto o efeito exato sobre tempo e memória depende do mecanismo de propagação empregado por CPRR, BFS-TREE ou RDPAC.

O experimento não foi instrumentado para registrar tempo de execução, uso máximo de memória ou consumo de recursos por etapa. Por isso, os resultados permitem discutir eficácia, mas não comparar a eficiência computacional dos métodos nem quantificar o compromisso entre ganho e custo. Também não foi realizada uma análise de sensibilidade da janela de 200 documentos. No plano estatístico, não foram aplicados testes de significância ou intervalos de confiança às diferenças entre sistemas.

Finalmente, as comparações com alternativas externas não cobrem todo o desenho experimental: o *cross-encoder* foi avaliado em três conjuntos, e a RRF somente no SciDocs. Da mesma forma, a análise qualitativa utiliza casos selecionados e projeções bidimensionais. Esses recortes são suficientes para contextualizar os resultados e formular hipóteses, mas não para estabelecer conclusões gerais sobre superioridade, eficiência ou comportamento por consulta.

7 Conclusão

7.1 Síntese dos Resultados

Esta dissertação investigou se métodos de re-ranqueamento contextual não supervisionado, originalmente desenvolvidos para recuperação de imagens, poderiam ser transferidos para a recuperação textual. A hipótese central foi que relações recorrentes de vizinhança entre documentos oferecem informação complementar às comparações diretas entre consulta e documento. Os resultados sustentam essa hipótese no escopo avaliado: operando somente sobre as posições das listas ranqueadas, os métodos produziram melhorias sobre recuperadores esparsos e densos sem treinamento adicional ou uso de rótulos no re-ranqueamento.

O protocolo experimental contemplou 972 configurações, resultantes da combinação de nove conjuntos de dados, quatro modelos de recuperação, três métodos contextuais e nove valores de K, todas efetivamente executadas e coletadas. Nas 36 combinações de conjunto e recuperador, a melhor configuração encontrada aumentou a Precisão@20 em 35 casos, permanecendo praticamente estável no caso restante. No Mental-Health com Contriever e CPRR ($K = 75$), foram observados os maiores ganhos relativos entre as configurações selecionadas: 19,8% em Precisão@20 e 33,5% em MAP@20.

Esses números não representam, contudo, um comportamento uniforme entre métricas e configurações. A melhoria foi mais regular em Precisão@20, adotada como critério de seleção, enquanto algumas combinações apresentaram redução de MAP. O CPRR se destacou principalmente nos conjuntos baseados em categoria, o BFS-TREE foi selecionado com maior frequência nos conjuntos BEIR e o RDPAC apresentou seu resultado mais expressivo em uma situação específica, no ArguAna com BM25. Assim, os resultados não indicam um método ou valor de K universal, mas mostram que o sinal contextual pode complementar diferentes recuperadores quando sua escala de vizinhança é compatível com a coleção.

A análise de sensibilidade mostrou melhor desempenho médio do CPRR em torno de $K = 20$ e do BFS-TREE em torno de $K = 30$. Também foi observada uma associação entre a densidade de documentos relevantes por consulta e valores maiores de K. Como essa relação foi estimada sobre apenas nove conjuntos e coincide parcialmente com diferentes formas de definir relevância, ela deve ser entendida como um indício para investigações futuras, não como uma regra causal de configuração.

As comparações com um *cross-encoder* supervisionado e com RRF ajudaram a posicionar a proposta. Nos três conjuntos examinados, o *cross-encoder* melhorou o BM25,

mas não o melhor recuperador denso; no SciDocs, o UDLF também superou as configurações de RRF avaliadas. Esses experimentos foram ilustrativos e não sustentam uma alegação de superioridade geral. Em conjunto com as análises qualitativas, eles mostram cenários nos quais a informação de vizinhança oferece um sinal útil que não é capturado integralmente pelos modelos de primeiro estágio.

7.2 Contribuições

A primeira contribuição é metodológica: a dissertação transporta para o domínio textual três métodos contextuais baseados em informações de ranqueamento, CPRR, BFS-TREE e RDPAC, e descreve como listas produzidas por recuperadores textuais podem ser convertidas, processadas e avaliadas pelo UDLF. Como os métodos não acessam diretamente os vetores originais, a investigação estabelece uma ligação entre técnicas desenvolvidas para recuperação visual e problemas de recuperação textual.

A segunda contribuição é empírica. O trabalho organiza uma avaliação sobre cinco domínios, dois regimes de relevância, um recuperador esparso e três recuperadores densos, examinando tanto o efeito dos métodos quanto sua sensibilidade ao parâmetro K. Além dos resultados completos, são apresentadas comparações com abordagens supervisionadas e de fusão, bem como análises qualitativas que ajudam a interpretar as alterações observadas nas listas ranqueadas.

A terceira contribuição é de engenharia e reprodutibilidade. O *UDLF Text Espresso* consolida coleta de conjuntos, indexação, recuperação, conversão de formatos, execução do UDLF, rastreamento de artefatos e avaliação em um único fluxo experimental. Essa infraestrutura foi necessária para viabilizar o volume de execuções do estudo e permanece disponível para replicações e extensões.

Parte dos resultados desta dissertação foi consolidada no artigo *Context-Aware Text Re-Ranking through Unsupervised Manifold Learning*, aceito para publicação na Trilha Principal do 41º Simpósio Brasileiro de Banco de Dados (SBBD 2026) [133]. O artigo reúne uma versão condensada da proposta e dos experimentos centrais, enquanto esta dissertação amplia esse material com a fundamentação teórica, a descrição detalhada do *UDLF Text Espresso*, o protocolo experimental completo, a análise de sensibilidade ao parâmetro K e as comparações adicionais.

7.3 Limitações

As conclusões devem ser interpretadas dentro do desenho experimental adotado. O melhor valor de K e, nas sínteses, o melhor método foram selecionados a partir da Precisão@20 nos próprios dados de avaliação. Sem uma partição de validação, os resultados

descrevem o melhor desempenho encontrado para cada cenário, mas não definem como selecionar a configuração quando os julgamentos de relevância não estão disponíveis.

A escala e a construção dos conjuntos também delimitam o alcance dos achados. Os conjuntos BEIR avaliados estão entre os menores da coleção, enquanto os quatro conjuntos adicionais definem relevância pelo compartilhamento de categoria, produzindo uma estrutura distinta e mais densa. A janela foi fixada em 200 documentos. Embora esse limite restrinja o volume das listas processadas, não foram realizadas medições sistemáticas de tempo e memória que permitam comparar a eficiência dos métodos. Também não foram aplicados testes de significância às diferenças observadas.

Por fim, os estudos com alternativas externas foram parciais: o *cross-encoder* foi aplicado a três conjuntos, e a RRF foi examinada somente no SciDocs. As projeções UMAP e listas individuais complementam a interpretação, mas permanecem ilustrativas. Essas limitações não anulam os resultados observados; elas definem com maior precisão quais conclusões podem ser extraídas e quais ainda exigem investigação.

7.4 Trabalhos Futuros

O primeiro desdobramento é desenvolver estratégias de seleção de K que não dependam dos julgamentos usados na avaliação. A associação observada com a densidade de relevância pode orientar hipóteses, mas deve ser confrontada com características disponíveis sem supervisão, como distribuição de distâncias, estabilidade das vizinhanças e reciprocidade entre listas ranqueadas. Protocolos com validação separada também permitiriam medir quanto do ganho observado permanece quando a configuração é escolhida antes da avaliação final.

Outra frente é variar a janela de 200 documentos utilizada neste trabalho. Janelas menores podem reduzir custo e restringir o contexto, enquanto janelas maiores podem revelar relações adicionais ou introduzir vizinhos pouco relacionados. Avaliar essa variável em conjunto com tempo de execução, memória e qualidade permitirá caracterizar melhor o compromisso entre eficácia e custo. O mesmo protocolo deve ser aplicado a coleções maiores, novos domínios e diferentes granularidades de relevância.

As comparações externas também devem ser ampliadas. A aplicação de diferentes *cross-encoders* a todos os conjuntos, acompanhada de análise estatística por consulta, permitiria identificar com maior segurança os regimes em que supervisão e contexto oferecem benefícios distintos. Para fusão de listas ranqueadas, será importante variar a constante k da RRF, ampliar a avaliação para outros conjuntos e avaliar a fusão contextual disponível no próprio UDLF, comparando a combinação de sistemas com o re-ranqueamento de um único recuperador.

No plano de engenharia, o *UDLF Text Espresso* pode incorporar seleção automatizada de parâmetros, validação separada, medições de recursos e relatórios estatísticos. Essas extensões reduziriam intervenções manuais e facilitariam a reprodução do protocolo em novas modalidades e coleções.

Em síntese, esta dissertação mostra que a estrutura de vizinhança de listas ranqueadas textuais contém informação aproveitável por métodos contextuais não supervisionados. O principal resultado não é a identificação de uma configuração universal, mas a demonstração de que esse sinal pode complementar modelos esparsos e densos sem exigir novo treinamento.

Referências

[1] Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774* (2023).

[2] Al-Azani, S., and El-Alfy, E.-S. M. Early and late fusion of emojis and text to enhance opinion mining. *IEEE Access 9* (2021), 121031–121045.

[3] Alsentzer, E., Murphy, J. R., Boag, W., Weng, W.-H., Jin, D., Naumann, T., and McDermott, M. Publicly available clinical bert embeddings. *arXiv preprint arXiv:1904.03323* (2019).

[4] Arora, S., Liang, Y., and Ma, T. A simple but tough-to-beat baseline for sentence embeddings. In *International conference on learning representations* (2017).

[5] Atrey, P. K., Hossain, M. A., El Saddik, A., and Kankanhalli, M. S. Multimodal fusion for multimedia analysis: a survey. *Multimedia systems 16* (2010), 345–379.

[6] Baeza-Yates, R., and Ribeiro-Neto, B. *Recuperação de Informação - 2ed: Conceitos e Tecnologia das Máquinas de Busca*. Bookman Editora, 2013.

[7] Bakkali, S., Ming, Z., Coustaty, M., and Rusinol, M. Visual and textual deep feature fusion for document image classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (June 2020).

[8] Bengio, Y., Ducharme, R., and Vincent, P. A neural probabilistic language model. In *Advances in Neural Information Processing Systems* (2000), T. Leen, T. Dietterich, and V. Tresp, Eds., vol. 13, MIT Press.

[9] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. Enriching Word Vectors with Subword Information. *Transactions of the Association for Computational Linguistics 5* (06 2017), 135–146.

[10] Boteva, V., Gholipour, D., Sokolov, A., and Riezler, S. A full-text learning to rank dataset for medical information retrieval.

[11] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS) 33* (2020), 1877–1901.

[12] Burges, C. J. C. From ranknet to lambdarank to lambdamart: An overview. Tech. Rep. MSR-TR-2010-82, Microsoft Research, 2010.

[13] Cleverdon, C. The cranfield tests on index language devices. *Aslib Proceedings 19*, 6 (1967), 173–194.

[14] Cohan, A., Feldman, S., Beltagy, I., Downey, D., and Weld, D. S. Specter: Document-level representation learning using citation-informed transformers. In *ACL* (2020).

[15] Conneau, A., Kiela, D., Schwenk, H., Barrault, L., and Bordes, A. Supervised learning of universal sentence representations from natural language inference data. *arXiv preprint arXiv:1705.02364* (2017).

[16] Cormack, G. V., Clarke, C. L., and Buettcher, S. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval* (2009), pp. 758–759.

[17] COUTINHO, J. F. Avaliação de técnicas de combinação de embeddings para a análise de sentimentos de produtos escritos em português-br. Master's thesis, Universidade Federal de Pernambuco, 2022.

[18] de Oliveira Lima, J. A. Lexml. *Implantação e gestão de repositórios institucionais* (2013), 249.

[19] de Souza P. Moreira, G., Osmulski, R., Xu, M., Ak, R., Schifferer, B., and Oldridge, E. Nv-retriever: Improving text embedding models with effective hard-negative mining, 2024.

[20] Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Ranzato, M., Senior, A., Tucker, P., Yang, K., et al. Large scale distributed deep networks. *Advances in neural information processing systems 25* (2012).

[21] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1* (Minneapolis, Minnesota, 2019), Association for Computational Linguistics, pp. 4171–4186.

[22] Dumais, S. T. Latent semantic analysis. *Annual Review of Information Science and Technology (ARIST) 38* (2004), 189–230.

[23] Dwork, C., Kumar, R., Naor, M., and Sivakumar, D. Rank aggregation methods for the Web. In *Proceedings of the 10th International Conference on World Wide Web (WWW)* (2001), Association for Computing Machinery, pp. 613–622.

[24] Eberendu, A. C., et al. Unstructured data: an overview of the data of big data. *International Journal of Computer Trends and Technology 38*, 1 (2016), 46–50.

[25] Fan, W., Ding, Y., Ning, L., Wang, S., Li, H., Yin, D., Chua, T.-S., and Li, Q. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2024), KDD '24, Association for Computing Machinery, p. 6491–6501.

[26] Ferneda, E. *Introdução aos modelos computacionais de recuperação de informação.* Ciência moderna, 2012.

[27] Fox, E. A., and Shaw, J. A. Combination of multiple searches. In *Proceedings of the 2nd Text REtrieval Conference (TREC-2)* (1994), pp. 243–252.

[28] Gadjokov, V., Dragulev, V., Gove, N., and Schmid, H. Stairs user's manual. Tech. rep., International Atomic Energy Agency, 1976.

[29] Gan, W., Ye, Z., Wan, S., and Yu, P. S. Web 3.0: The future of internet. In *Companion Proceedings of the ACM Web Conference 2023* (2023), pp. 1266–1275.

[30] Ganitkevitch, J., Van Durme, B., and Callison-Burch, C. Ppdb: The paraphrase database. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies* (2013), pp. 758–764.

[31] Gao, T., Yao, X., and Chen, D. Simcse: Simple contrastive learning of sentence embeddings. *CoRR abs/2104.08821* (2021).

[32] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, H., and Wang, H. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997 2*, 1 (2023).

[33] Ghannay, S., Esteve, Y., Camelin, N., Dutrey, C., Santiago, F., and Adda-Decker, M. Combining continuous word representation and prosodic features for asr error prediction. In *Statistical Language and Speech Processing: Third International Conference, SLSP 2015, Budapest, Hungary, November 24-26, 2015, Proceedings 3* (2015), Springer, pp. 84–95.

[34] Ghannay, S., Favre, B., Estève, Y., and Camelin, N. Word embedding evaluation and combination. In *Proceedings of the Tenth International Conference*

on Language Resources and Evaluation (LREC'16) (Portorož, Slovenia, May 2016), N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, and S. Piperidis, Eds., European Language Resources Association (ELRA), pp. 300–305.

[35] Gövert, N., and Kazai, G. Overview of the initiative for the evaluation of xml retrieval (inex) 2002. In *INEX Workshop* (2002), Citeseer, pp. 1–17.

[36] Greene, D., and Cunningham, P. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proceedings of the 23rd International Conference on Machine Learning* (New York, NY, USA, 2006), ICML '06, Association for Computing Machinery, p. 377–384.

[37] Guimarães Pedronette, D. C., Pascotti Valem, L., and Latecki, L. J. Efficient rank-based diffusion process with assured convergence. *Journal of Imaging 7*, 3 (2021).

[38] Guo, J., Fan, Y., Pang, L., Yang, L., Ai, Q., Zamani, H., Wu, C., Croft, W. B., and Cheng, X. A deep look into neural ranking models for information retrieval. *Information Processing & Management 57*, 6 (2020), 102067.

[39] Harman, D. Overview of the first trec conference. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval* (1993), pp. 36–47.

[40] Harris, Z. S. *Distributional Structure*. Springer Netherlands, Dordrecht, 1981, pp. 3–22.

[41] Hartmann, N., Fonseca, E., Shulby, C., Treviso, M., Rodrigues, J., and Aluisio, S. Portuguese word embeddings: Evaluating on word analogies and natural language tasks. *arXiv preprint arXiv:1708.06025* (2017).

[42] Hasan, F., Roy, A., and Pan, S. Integrating text embedding with traditional nlp features for clinical relation extraction. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)* (2020), IEEE, pp. 418–425.

[43] Hersh, W., Buckley, C., Leone, T. J., and Hickam, D. Ohsumed: An interactive retrieval evaluation and new large test collection for research. In *SIGIR '94* (London, 1994), B. W. Croft and C. J. van Rijsbergen, Eds., Springer London, pp. 192–201.

[44] Incitti, F., Urli, F., and Snidaro, L. Beyond word embeddings: A survey. *Information Fusion 89* (2023), 418–436.

[45] Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., and Grave, E. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118* (2021).

[46] Järvelin, K., and Kekäläinen, J. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS) 20*, 4 (2002), 422–446.

[47] Joachims, T. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2002), Association for Computing Machinery, pp. 133–142.

[48] Johnson, J., Douze, M., and Jégou, H. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data 7*, 3 (2019), 535–547.

[49] Kalchbrenner, N., Grefenstette, E., and Blunsom, P. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188* (2014).

[50] Kanade, A., Maniatis, P., Balakrishnan, G., and Shi, K. Learning and evaluating contextual embedding of source code. In *Proceedings of the 37th International Conference on Machine Learning* (13–18 Jul 2020), H. D. III and A. Singh, Eds., vol. 119 of *Proceedings of Machine Learning Research*, PMLR, pp. 5110–5121.

[51] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2020), Association for Computational Linguistics, pp. 6769–6781.

[52] Keselj, V. Book review: Speech and language processing by daniel jurafsky and james h. martin. *Computational Linguistics 35*, 3 (2009).

[53] Khan, M. A., Yadav, N., Jain, M., and Goyal, S. The art of embedding fusion: Optimizing hate speech detection, 2023.

[54] Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. Skip-thought vectors. In *Advances in Neural Information Processing Systems* (2015), C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, Eds., vol. 28, Curran Associates, Inc.

[55] Kolomiyets, O., and Moens, M.-F. A survey on question answering technology from an information retrieval perspective. *Information Sciences 181*, 24 (2011), 5412–5434.

[56] Kowsari, K., Brown, D. E., Heidarysafa, M., Jafari Meimandi, K., Gerber, M. S., and Barnes, L. E. Hdltex: Hierarchical deep learning for text classification. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2017), pp. 364–371.

[57] Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683* (2017).

[58] Lakmal, D., Ranathunga, S., Peramuna, S., and Herath, I. Word embedding evaluation for Sinhala. In *Proceedings of the Twelfth Language Resources and Evaluation Conference* (Marseille, France, May 2020), N. Calzolari, F. Béchet, P. Blache, K. Choukri, C. Cieri, T. Declerck, S. Goggi, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, and S. Piperidis, Eds., European Language Resources Association, pp. 1874–1881.

[59] Lan, Z. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942* (2019).

[60] Landauer, T. K., Laham, D., Rehder, B., and Schreiner, M. E. How well can passage meaning be derived without using word order? a comparison of latent semantic analysis and humans. In *Proceedings of the 19th annual meeting of the Cognitive Science Society* (1997), pp. 412–417.

[61] Lavrenko, V., and Croft, W. B. Relevance-based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (2001), Association for Computing Machinery, pp. 120–127.

[62] Le, Q., and Mikolov, T. Distributed representations of sentences and documents. In *International conference on machine learning* (2014), PMLR, pp. 1188–1196.

[63] Learning, D. Ian goodfellow, yoshua bengio, aaron courville. *The reference book for deep learning models 1* (2016).

[64] Lee, J., Yoon, W., Kim, S., Kim, D., Kim, S., So, C. H., and Kang, J. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics 36*, 4 (2020), 1234–1240.

[65] Lee, K.-S., Park, Y.-C., and Choi, K.-S. Re-ranking model based on document clusters. *Information Processing & Management 37*, 1 (2001), 1–14.

[66] Lester, B., Pressel, D., Hemmeter, A., Choudhury, S. R., and Bangalore, S. Multiple word embeddings for increased diversity of representation. *CoRR abs/2009.14394* (2020).

[67] Levy, O., and Goldberg, Y. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)* (2014), pp. 302–308.

[68] Lew, M. S., Sebe, N., Djeraba, C., and Jain, R. Content-based multimedia information retrieval: State of the art and challenges. *ACM Trans. Multimedia Comput. Commun. Appl. 2*, 1 (feb 2006), 1–19.

[69] Lewis, D. Reuters-21578 Text Categorization Collection. UCI Machine Learning Repository, 1987. DOI: https://doi.org/10.24432/C52G6M.

[70] Li, X., Jia, P., Xu, D., Wen, Y., Zhang, Y., Zhang, W., Wang, W., Wang, Y., Du, Z., Li, X., Liu, Y., Guo, H., Tang, R., and Zhao, X. A survey of personalization: From rag to agent. *ACM Trans. Inf. Syst. 44*, 4 (May 2026).

[71] Liu, Q., Kusner, M. J., and Blunsom, P. A survey on contextual embeddings. *arXiv preprint arXiv:2003.07278* (2020).

[72] Liu, T.-Y. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval 3*, 3 (2009), 225–331.

[73] Liu, Y. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

[74] Liu, Y.-H., Chen, Y.-L., and Chang, P.-Y. A deep multi-embedding model for mobile application recommendation. *Decision Support Systems 173* (2023), 114011.

[75] Maia, M., Handschuh, S., Freitas, A., Davis, B., McDermott, R., Zarrouk, M., and Balahur, A. Www'18 open challenge: Financial opinion mining and question answering. In *Companion Proceedings of the The Web Conference 2018* (Republic and Canton of Geneva, CHE, 2018), WWW '18, International World Wide Web Conferences Steering Committee, p. 1941–1942.

[76] Manning, C. D. Introduction to information retrieval, 2008.

[77] Mikolov, T. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[78] Misra, R. News category dataset, 2022. Conjunto de notícias do *The Huffington Post*.

[79] Mitchell, J., and Lapata, M. Vector-based models of semantic composition. In *proceedings of ACL-08: HLT* (2008), pp. 236–244.

[80] Mitra, B., Nalisnick, E., Craswell, N., and Caruana, R. A dual embedding space model for document ranking. *arXiv preprint arXiv:1602.01137* (2016).

[81] Mozer, M. C. Inductive information retrieval using parallel distributed computation.

[82] Muennighoff, N. Sgpt: Gpt sentence embeddings for semantic search, 2022.

[83] Muennighoff, N., Su, H., Wang, L., Yang, N., Wei, F., Yu, T., Singh, A., and Kiela, D. Generative representational instruction tuning, 2024.

[84] Muennighoff, N., Tazi, N., Magne, L., and Reimers, N. Mteb: Massive text embedding benchmark, 2023.

[85] Mukherjee, P. Mental health text classification dataset (4-class). Kaggle, 2025. Conjunto derivado para classificação de saúde mental (categorias: Anxiety, Depression, Normal, Suicidal), combinando fontes públicas de Komati, Sarkar e Murarka et al. (2021). Licença CC BY 4.0.

[86] Nalisnick, E., Mitra, B., Craswell, N., and Caruana, R. Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web* (Republic and Canton of Geneva, CHE, 2016), WWW '16 Companion, International World Wide Web Conferences Steering Committee, p. 83–84.

[87] Nguyen, D. Q., Vu, T., and Nguyen, A. T. Bertweet: A pre-trained language model for english tweets. *arXiv preprint arXiv:2005.10200* (2020).

[88] Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., and Deng, L. MS MARCO: A human generated machine reading comprehension dataset. *CoRR abs/1611.09268* (2016).

[89] Nogueira, R., and Cho, K. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085* (2019).

[90] Patil, R., Boit, S., Gudivada, V., and Nandigam, J. A survey of text representation and embedding techniques in NLP. *IEEE Access 11* (2023), 36120–36146.

[91] Pedronette, D. C. G., Valem, L. P., Almeida, J., and da S. Torres, R. Multimedia retrieval through unsupervised hypergraph-based manifold ranking. *IEEE Transactions on Image Processing 28*, 12 (2019), 5824–5838.

[92] Pedronette, D. C. G., Valem, L. P., and da S. Torres, R. A bfs-tree of ranking references for unsupervised manifold learning. *Pattern Recognition 111* (2021), 107666.

[93] Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), pp. 1532–1543.

[94] Pereira-Ferrero, V. H., Lewis, T. G., Valem, L. P., Ferrero, L. G. P., Pedronette, D. C. G., and Latecki, L. J. Unsupervised affinity learning based on manifold analysis for image retrieval: A survey. *Computer Science Review 53* (2024), 100657.

[95] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations, 2018.

[96] Radford, A. Improving language understanding by generative pre-training.

[97] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. *OpenAI blog 1*, 8 (2019), 9.

[98] Raja, E., Soni, B., and Borgohain, S. K. Harnessing heterogeneity: A multi-embedding ensemble approach for detecting fake news in dravidian languages. *Computers and Electrical Engineering 120* (2024), 109661.

[99] Rajpurkar, P. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250* (2016).

[100] Reimers, N., and Gurevych, I. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* (2019), Association for Computational Linguistics, pp. 3982–3992.

[101] Robertson, S., Zaragoza, H., et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval 3*, 4 (2009), 333–389.

[102] Robertson, S. E., and Jones, K. S. Relevance weighting of search terms. *Journal of the American Society for Information science 27*, 3 (1976), 129–146.

[103] Robertson, S. E., Walker, S., Jones, S., Hancock-Beaulieu, M. M., Gatford, M., et al. Okapi at trec-3. *Nist Special Publication Sp 109* (1995), 109.

[104] Roca, S., Rosset, S., García, J., and Alesanco, Á. Evaluation of embeddings in medication domain for spanish language using joint natural language understanding. In *8th European Medical and Biological Engineering Conference: Proceedings*

of the EMBEC 2020, November 29–December 3, 2020 Portorož, Slovenia (2021), Springer, pp. 510–517.

[105] Rocchio, J. J. Relevance feedback in information retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, G. Salton, Ed. Prentice-Hall, 1971, pp. 313–323.

[106] Rouvier, M., Delecraz, S., Favre, B., Bendris, M., and Bechet, F. Multimodal embedding fusion for robust speaker role recognition in video broadcast. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)* (2015), pp. 383–389.

[107] Roy, D., Paul, D., Mitra, M., and Garain, U. Using word embeddings for automatic query expansion. *arXiv preprint arXiv:1606.07608* (2016).

[108] Salton, G. Some experiments in the generation of word and document associations. In *Proceedings of the December 4-6, 1962, fall joint computer conference* (1962), pp. 234–250.

[109] Salton, G. *Automatic Information Organization and Retrieval.* McGraw Hill Text, 1968.

[110] Salton, G., and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing & Management 24*, 5 (1988), 513–523.

[111] Salton, G., and Buckley, C. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science 41*, 4 (1990), 288–297.

[112] Salton, G., and Lesk, M. E. The smart automatic document retrieval systems—an illustration. *Communications of the ACM 8*, 6 (1965), 391–398.

[113] Salton, G., Wong, A., and Yang, C.-S. A vector space model for automatic indexing. *Communications of the ACM 18*, 11 (1975), 613–620.

[114] Sanderson, M., and Croft, W. B. The history of information retrieval research. *Proceedings of the IEEE 100*, Special Centennial Issue (2012), 1444–1451.

[115] Sanh, V. Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[116] Sayers, E. W., Bolton, E. E., Brister, J. R., Canese, K., Chan, J., Comeau, D., Connor, R., Funk, K., Kelly, C., Kim, S., Madej, T., Marchler-Bauer, A., Lanczycki, C., Lathrop, S., Lu, Z., Thibaud-Nissen, F., Murphy, T., Phan, L., Skripchenko, Y., Tse, T., Wang, J., Williams,

R., Trawick, B., Pruitt, K., and Sherry, S. Database resources of the national center for biotechnology information. *Nucleic Acids Research 50*, D1 (12 2021), D20–D26.

[117] Schwenk, H. Continuous space language models. *Computer Speech & Language 21*, 3 (2007), 492–518.

[118] Sharifani, K., and Amini, M. Machine learning and deep learning: A review of methods and applications. *World Information Technology and Engineering Journal 10*, 07 (2023), 3897–3904.

[119] Smolensky, P., and Bein, J. Application of the interactive activation model to document retrieval; cu-cs-405-88.

[120] Snoek, C. G. M., Worring, M., and Smeulders, A. W. M. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th Annual ACM International Conference on Multimedia* (New York, NY, USA, 2005), MULTIMEDIA '05, Association for Computing Machinery, p. 399–402.

[121] Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (2013), pp. 1631–1642.

[122] Souza, F., Nogueira, R., and Lotufo, R. Bertimbau: Pretrained bert models for brazilian portuguese. In *Intelligent Systems* (Cham, 2020), R. Cerri and R. C. Prati, Eds., Springer International Publishing, pp. 403–417.

[123] Speer, R., and Chin, J. An ensemble method to produce high-quality word embeddings (2016). *arXiv preprint arXiv:1604.01692* (2016).

[124] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research 15*, 1 (2014), 1929–1958.

[125] Taine, S. The medical literature analysis and retrieval system (medlars) of the us national library of medicine. *Methods of information in medicine 2*, 02 (1963), 65–69.

[126] Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., and Gurevych, I. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)* (2021).

[127] The pandas development team. pandas-dev/pandas: Pandas. https://github.com/pandas-dev/pandas, 2020. DOI: https://doi.org/10.5281/zenodo.3509134.

[128] Turian, J., Ratinov, L., and Bengio, Y. Word representations: a simple and general method for semi-supervised learning. In *Proceedings of the 48th annual meeting of the association for computational linguistics* (2010), pp. 384–394.

[129] Valem, L. P., and Guimarães Pedronette, D. C. Unsupervised similarity learning through cartesian product of ranking references for image retrieval tasks. In *2016 29th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)* (2016), pp. 249–256.

[130] Valem, L. P., and Pedronette, D. C. G. Unsupervised selective rank fusion for image retrieval tasks. *Neurocomputing 377* (2020), 182–199.

[131] Valem, L. P., and Pedronette, D. C. G. a. An unsupervised distance learning framework for multimedia retrieval. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval* (New York, NY, USA, 2017), ICMR '17, ACM, pp. 107–111.

[132] Vaswani, A. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

[133] Venezian, L. F. A., and Pedronette, D. C. G. Context-aware text re-ranking through unsupervised manifold learning. In *Anais do XLI Simpósio Brasileiro de Banco de Dados (SBBD)* (Porto Alegre, RS, Brasil, 2026), Sociedade Brasileira de Computação (SBC). Aceito para publicação na Trilha Principal.

[134] Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning* (2008), pp. 1096–1103.

[135] Voorhees, E. M. The evolution of cranfield. In *Information Retrieval Evaluation in a Changing World: Lessons Learned from 20 Years of CLEF*. Springer, 2019, pp. 45–69.

[136] Wachsmuth, H., Trenkmann, M., Stein, B., Engels, G., and Palakarska, T. A review corpus for argumentation analysis. In *Proceedings of the 15th International Conference on Intelligent Text Processing and Computational Linguistics* (Berlin Heidelberg New York, 2014), A. Gelbukh, Ed., Springer, pp. 115–127.

[137] Wadden, D., Lin, S., Lo, K., Wang, L. L., van Zuylen, M., Cohan, A., and Hajishirzi, H. Fact or fiction: Verifying scientific claims. In *EMNLP* (2020).

[138] Wang, A. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461* (2018).

[139] Wang, J., Huang, J. X., Tu, X., Wang, J., Huang, A. J., Laskar, M. T. R., and Bhuiyan, A. Utilizing bert for information retrieval: Survey, applications, resources, and challenges, 2024.

[140] Wang, L., Yang, N., Huang, X., Jiao, B., Yang, L., Jiang, D., Majumder, R., and Wei, F. Text embeddings by weakly-supervised contrastive pre-training, 2024.

[141] Wang, L., Yang, N., and Wei, F. Query2doc: Query expansion with large language models. *arXiv preprint arXiv:2303.07678* (2023).

[142] Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Advances in Neural Information Processing Systems (NeurIPS)* (2020), vol. 33, pp. 5776–5788.

[143] Wieting, J., Bansal, M., Gimpel, K., and Livescu, K. Towards universal paraphrastic sentence embeddings. *arXiv preprint arXiv:1511.08198* (2015).

[144] Wolf, T. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).

[145] Wu, J., Gan, W., Chen, Z., Wan, S., and Lin, H. Ai-generated content (aigc): A survey. *arXiv preprint arXiv:2304.06632* (2023).

[146] Xiao, S., Liu, Z., Zhang, P., Muennighoff, N., Lian, D., and Nie, J.-Y. C-Pack: Packed resources for general chinese embeddings, 2023.

[147] Yadan, O. Hydra - a framework for elegantly configuring complex applications. Github, 2019.

[148] Yuan, Y. Improving information retrieval by semantic embedding, July 2020.

[149] Zhang, Y., and Wallace, B. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820* (2015).

[150] Zhao, H., Lu, Z., and Poupart, P. Self-adaptive hierarchical sentence model. *arXiv preprint arXiv:1504.05070* (2015).

[151] Zhong, Q., Ding, L., Liu, J., Du, B., and Tao, D. Can chatgpt understand too? a comparative study on chatgpt and fine-tuned bert. *arXiv preprint arXiv:2302.10198* (2023).

[152] Zhu, Y. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *arXiv preprint arXiv:1506.06724* (2015).

[153] Zou, W. Y., Socher, R., Cer, D., and Manning, C. D. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (2013), pp. 1393–1398.

DADOS CURRICULARES

Nome: Luis Felipe Antunes Venezian

Nome em citações bibliográficas: VENEZIAN, L. F. A.

Currículo Lattes: http://lattes.cnpq.br/5328845037164145

Formação acadêmica

- 2023–2026: Mestrado em Ciência da Computação, Universidade Estadual Paulista "Júlio de Mesquita Filho" (UNESP), sob orientação do Prof. Dr. Daniel Carlos Guimarães Pedronette.
- 2020–2022: MBA em Data Science e Analytics, Escola Superior de Agricultura "Luiz de Queiroz" da Universidade de São Paulo (ESALQ/USP).
- 2017–2019: Graduação em Análise e Desenvolvimento de Sistemas, Faculdade de Tecnologia de Ourinhos (FATEC).
- 2015–2016: Curso Técnico em Eletrônica, Escola Técnica Estadual Jacinto Ferreira de Sá (ETEC).

Atuação profissional

Atua nas áreas de engenharia e análise de dados. Desde 2024, exerce a função de *Senior Analytics Engineer* no Nubank.

Produção bibliográfica selecionada

VENEZIAN, L. F. A.; PEDRONETTE, D. C. G. *Context-Aware Text Re-Ranking through Unsupervised Manifold Learning*. In: 41º Simpósio Brasileiro de Bancos de Dados (SBBD), 2026, São Carlos. Anais [...]. Porto Alegre: Sociedade Brasileira de Computação, 2026. p. 127–140.