IR 2025/1 RC
Entity Search Challenge

Showcase Presentations

June 25 2025

[0.35726] – [Arthur e Philipe]

Toolkit: PyTerrier

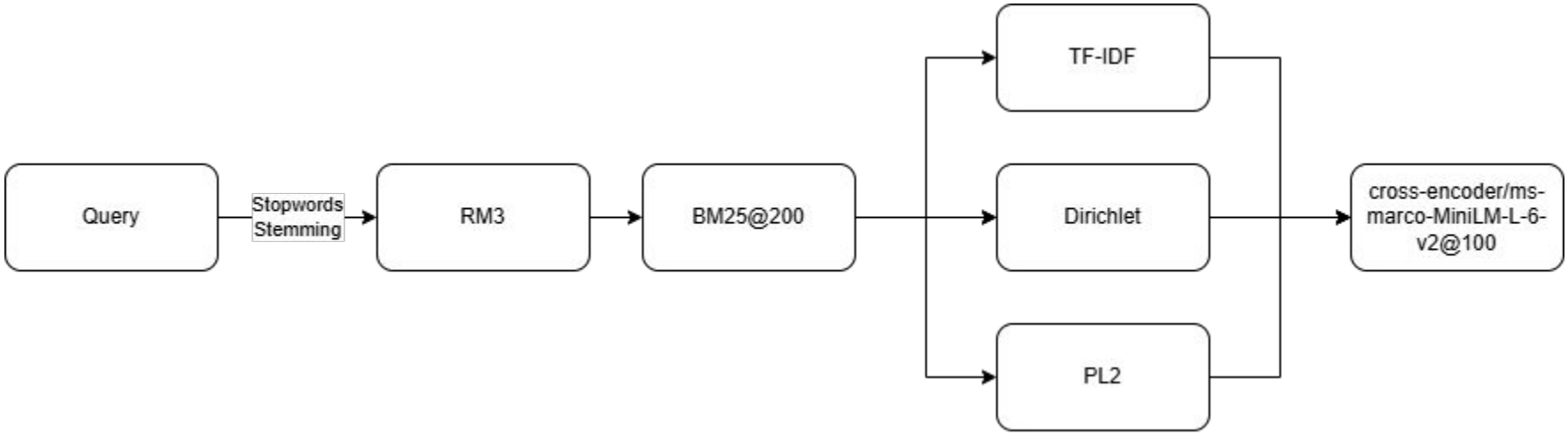
Index:

- Indexador PyTerrier (Porter + Stopword Removal), separando nos campos do corpus (separando em título, keywords e texto)

Pipeline:

- BM25 > RM3 > BM25 + TF + PL2 + Dirichlet > LTR (LightGBM) > Cross-Encoder (cross-encoder/ms-marco-MiniLM-L-6-v2)

[0.35726] – [Arthur e Philipe]



Score 0.37413 – Eduardo, Etelvina, Indra

- **Geração do Index com a biblioteca PyTerrier:**
  - Concatenação de título + keyword + texto respectivamente
  - Stemming e remoção de stop-words default da biblioteca (PorterStemmer)
  - Criação de um dataframe auxiliar para armazenar os offsets de cada índice para otimização
- **Filtragem Inicial:**
  - TF-IDF
- **Reranking dos 100 melhores documentos:**
  - cross-encoder/ms-marco-MiniLM-L12-v2 + Fine-tuning (com train_queries.csv e train_qrels.csv)

Score 0.37413 – Eduardo, Etelvina, Indra

- **O que aprendemos com tentativas anteriores:**
  - Remoção dos campos de título ou keyword piora o resultado final
  - TF-IDF performou ligeiramente melhor que o BM25
  - RM3 contribui para a melhoria nos resultados, mas não de forma significativa
  - O top-100 ranking produzido pelo TF-IDF já tem qualidade boa o bastante para ser enviado para o reranking

Score 0.37413 – Eduardo, Etelvina, Indra

- **Tentativas que não deram certo:**
  - MonoT5
  - Multi staging com MonoT5 + DuoT5 para top-20 documentos
  - Doc2Query
  - ms-marco-electra-base (ligeramente pior que o ms-marco-MiniLM-L12-v2)

Lucas Sacramento e Milena Moreira – 0.37546

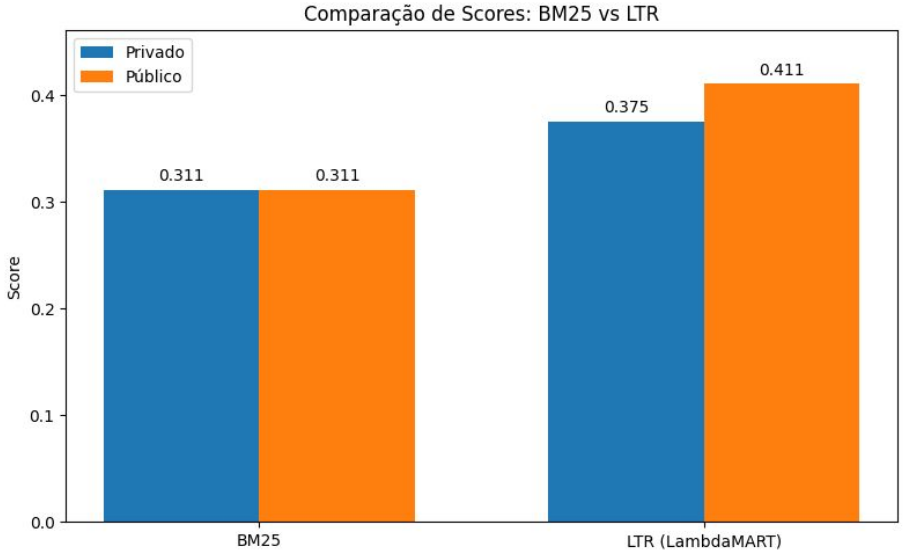
- **Estratégia: BM25 + LTR (LightGBM)**
- **Dados usados:** todos os dados disponibilizados para o desafio
- **Toolkit: Pyserini**
  - Pré-processamento: Remoção de stopwords e stemming
  - Indexação
  - BM25
- **Pipeline:**
  - Indexação
  - Recuperação do top-250 com BM25
  - Reranqueamento com LTR

Lucas Sacramento e Milena Moreira – 0.37546

- **Extração de Features**
  - Score do BM25
  - Tamanho do documento
  - N° de termos da query presente no documento
  - Outras features foram calculadas, mas pioraram ou não ajudaram o desempenho do modelo
- **Learning to Rank**
  - LightGBM – LambdaMART (objective = lambdarank) visto em aula
  - Treinado com os dados anotados (train_qrels.csv)
  - Ajuste manual de hiperparâmetros

Lucas Sacramento e Milena Moreira – 0.37546

- **Melhor submissão**
  - Score Privado: 0.37546
  - Score Público: 0.41060
  - Regularização aumentada (*lambda_l1*, *lambda_l2* = 3)
  - *learning_rate* = 0.03 com *n_estimators* = 7500
- Maior regularização melhorou a generalização do modelo
- Reranking supervisionado superou o BM25 isolado mesmo com poucas features



[0.39378] [Luciano]

Expansão de Consulta com SBERT e LTR Otimizado

Score Alcançado: 0.39378 (0.43883 - public)

Técnica Principal: A combinação de Expansão de Consulta Semântica (SBERT) com um Pipeline de Learning to Rank (LTR) otimizado por pesos de modelos de ranqueamento.

Como Funcionou:

- **Toolkit de Busca:** PyTerrier
- **Dados Explorados e Tratamento:** Tokenização, remove stopwords e stemming
- **Treinamento:** (75%/25% para treinamento/validação). Otimização de hiperparâmetros (incluindo pesos dos ranqueadores) via `pt.grid_scan`
- **Modelos de Ranking:** Recuperação inicial com modelos PyTerrier (BM25, TF-IDF, PL2, etc.). Expansão de Consulta com SBERT (`multi-qa-mpnet-base-dot-v1`). Ranking Final (LTR) com LightGBM, treinado para combinar pontuações.

Desafios e Limitações

Indexação Expandida

Problema: Embora a indexação com N-grams e keyphrases (gensim) visasse enriquecer a representação dos documentos, seu uso não resultou em diferenças significativas no score final em comparação com o índice básico.

Variações de Parâmetros BM25 Isoladas

Problema: A otimização isolada dos parâmetros específicos do BM25 (bm25.b, bm25.k_1, bm25.k_3) resultou na pontuação mais baixa entre as top 5 submissões.

Re-rankers Neurais Avançados (MonoT5 e DuoT5)

Problema: Apesar de promissores para capturar relações semânticas profundas, esses modelos não foram incluídos nas top 5 submissões.

Recuperação Densa (Dense Retrieval) como Solução Única

Problema: Embora explorada como uma linha de pesquisa avançada para similaridade semântica, não se destacou como a estratégia de melhor desempenho por si só ou nas combinações testadas que levaram aos melhores scores.

[0.39997] – [Arthur Codama]

- Bibliotecas utilizadas: Pyserini (Lucene), SentenceTransformers e LightGBM.
- Sem pré-processamento
- Indíces:
  - Esparso:
    - Título + keywords
    - Texto
  - Denso (“msmarco-MiniLM-L6-v3”)
    - Título, keywords e texto

[0.39997] – [Arthur Codama]

Pipeline de Busca

- Esparsa
  - BM25 com soma ponderada dos índices
- Densa
  - Similaridade de cosseno
- Híbrida
  - Reciprocal Rank Fusion com esparsa + densa
- LTR
  - LambdaMART (LambdaRank + GBDT)
  - Esparsa e Densa

$$RRFscore(d \in D) = \sum_{r \in R} \frac{1}{k + r(d)}$$

D - set of docs
R - set of rankings as permutation on 1..|D|
K - typically set to 60 by default

[0.39997] – [Arthur Codama]

Melhores resultados (Privado)

- Híbrida (0.40337)
  - BM25: B=0.3 e K1 = 1
  - RRF: K=60
- LTR (0.36223)
- Esparsa (0.35649)
  - BM25: B=0.3 e K1 = 0.6
- Densa (0.28855)

[0.40067] – [Jorge e Vitor]

Toolkit: PyTerrier

Index:

- Pré processamento: Padrão do PyTerrier (Porter + Stopword Removal)
- Documento: Título + Texto + Keywords

Baselines:

- TF-IDF: 0.29123
- BM25: 0.29283
- BM25 (pyserini): 0.31120
- QLD (pyserini): 0.30940
- RM3 (pyserini): 0.30699
- Rocchio (pyserini): 0.30549

[0.40067] – [Jorge e Vitor]

Estratégia final:

- Combinação de DPH com inL2
- Expansão de consultas com Bo1

[0.40067] – [Jorge e Vitor]

Estratégias que não funcionaram:

- Dense Retrieval com all-MiniLM-L6-v2
  - Muito demorado (+7 horas) e consome muito armazenamento (~38gb)
- Learning to Rank
  - Resultados não foram promissores

0.40327 - [Gabriel Lima Barros e Isabella Vignoli Gonçalves]

DCC

A estratégia foi expandir a consulta usando a técnica de **Pseudo-Relevance Feedback RM3** e, em seguida, realizar um refinamento da ordenação dos resultados com o modelo **DPH**

Uso de DPH (Primeira vez)	É utilizado para recuperar uma lista inicial de documentos com base em uma medição probabilística de relevância
Expansão de Consulta com RM3	Usa os documentos mais relevantes da recuperação inicial para gerar termos adicionais -> Melhorar o recall da consulta, incluindo **sinônimos** ou **termos relacionados** que não foram capturados inicialmente
Uso de DPH (Segunda vez)	Após a expansão da consulta, o DPH é reaplicado para refinar o ranking, considerando os novos termos da consulta

0.40327 - [Gabriel Lima Barros e Isabella Vignoli Gonçalves]

DCC

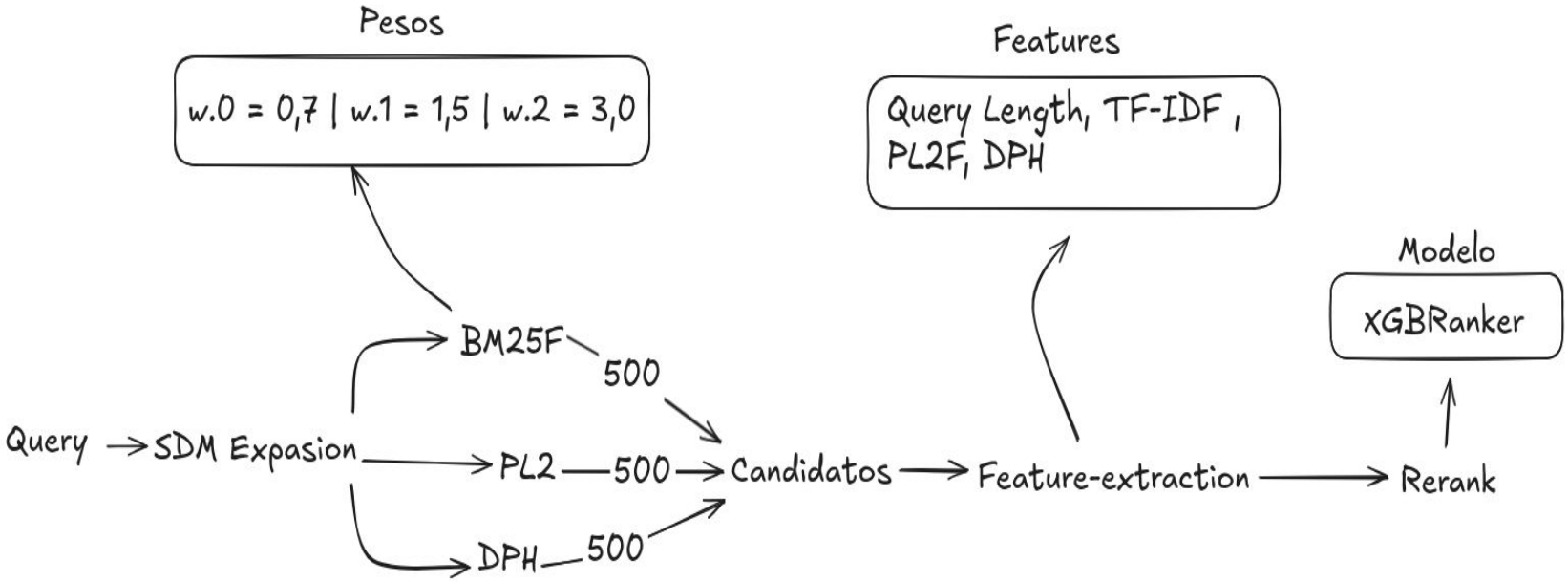
Pipeline é definido como a sequência DPH >> RM3 >> DPH

As consultas são transformadas aplicando o pipeline para gerar os resultados

Os resultados são ordenados por relevância e limite de 100 resultados

```
# Definir o modelo DPH e RM3
dph = pt.terrier.Retriever(index, wmodel="DPH")  # Recuperador DPH
rm3 = pt.rewrite.RM3(index)  # Aplicação de RM3 para expansão de consulta

# Definir o pipeline: DPH >> RM3 >> DPH
rm3_pipe = dph >> rm3 >> dph

# Processamento da consulta
results_df = rm3_pipe.transform(query_df)  # Aplicar o pipeline nas consultas

# Formatar os resultados para submissão
results_df = results_df[['qid', 'docno', 'score']].copy()  # Selecionar colunas relevantes
results_df = results_df.sort_values(['qid', 'score'], ascending=[True, False])  # Ordenação dos resultados

# Limitar os 100 melhores resultados por consulta
results_df = results_df.groupby('qid').head(100)

# Renomear colunas para o formato esperado
results_df = results_df[['qid', 'docno']].copy()  # Manter apenas as colunas 'qid' e 'docno'
results_df.columns = ['QueryId', 'EntityId']  # Renomear para os nomes esperados

# Salvar os resultados em CSV para submissão
submission_file_path = "C:/Users/gabri/Documents/RC/submissions/dphrm3.csv"
results_df.to_csv(submission_file_path, index=False)
```

[0.43100] – [Luis Antonio Duarte Sousa]



[0.41619] – [Giovana e Clara]

Index:

- Pré processamento: Remoção de stop worlds, steeming
- Documento: Título + Texto + Keywords
- Index: pyterrier

Ranks iniciais:

- TF-IDF: 0.29415
- BM25: 0.29302
- DFIC: 0.31547
- DPH: 0.32205

[0.41619] – [Giovana e Clara]

Estratégia final:

- Rank com DFIC e DPH, utilizando query expansion
- Normalização dos scores
- União dos rankings associando pesos a cada rankeador

[0.41619] – [Giovana e Clara]

Estratégias que não funcionaram:

- Learn to rank (Random forest e Xgboost)
- Utilização de mais rankeadores

Score 0.42878 – [Caio, Lucas e Victor]

Pré-processamento:

- Tokenização.
- Remoção de stopwords.
- Stemming.

Indexação e matching:

- Primeiro momento: Pyserini
- Depois: Pyterrier (Multiprocessing durante Indexação)

Score 0.42878 – [Caio, Lucas e Victor]

Recuperação supervisionada / Learning to Rank:

- LambdaMART com LightGBM.

Features usadas:

- BM25, TF-IDF e PL2.

Aperfeiçoamento:

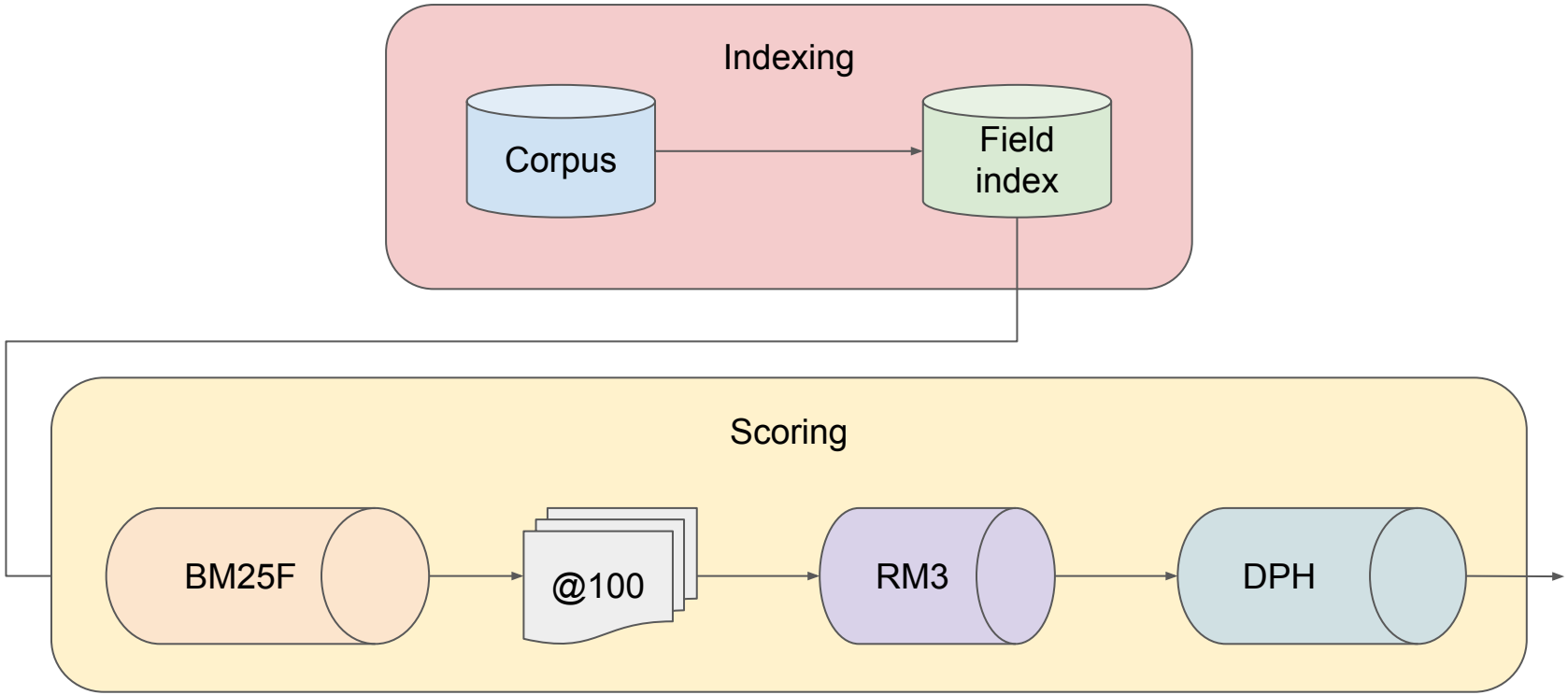
- Ponderar o texto por seu campo (Title, body, Keywords)
- BM25F na recuperação inicial (considera todos os campos de texto)

Score 0.42878 – [Caio, Lucas e Victor]

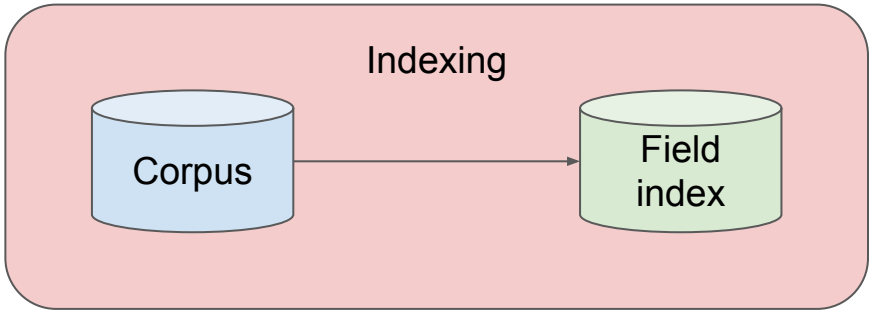
Tentativas de re-ranqueamento com modelos neurais (frustradas):

- CEDR (Contextualized Embeddings for Document Ranking
- MonoT5 (pointwise)
- DuoT5 (pairwise)

0.42025 – Gabriel, Maria Luiza e Mariano



0.42025 – Gabriel, Maria Luiza e Mariano



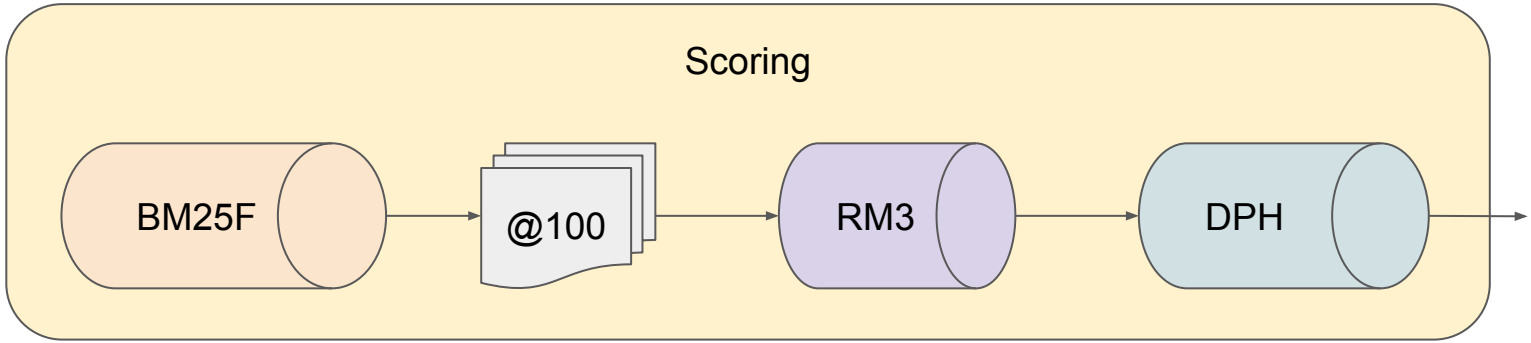
Pre-process:

- Lower-case
- Normaliza acentuação
- Remove aspas simples/dobradas e outras pontuações, como ‘!’ e ‘?’

PyTerrier index:

- **Steemer:** TerrierStemmer.porter
- **Stopwords:** TerrierStemmer.english
- **Tokenizer:** TerrierTokenizer.english
- **Fields:** “title”, “text” and “keywords”

0.42025 – Gabriel, Maria Luiza e Mariano



Parameter tuning for BM25F and RM3

Grid-search on train set → best local nDCG@100

Best local = best private ≠ best public

Weights

- `title.weight: 2.5,`
- `title.bias: 1.8,`
- `keywords.weight: 0.5,`
- `keywords.bias: 0.5,`
- `text.weight: 0.5,`
- `text.bias: 0.6,`
- `fb_docs=12,`
- `fb_terms=40,`
- `fb_lambda=0.65`

[0.46764] – [JVHeHe - João Vítor, Helio, Henrique]

Reranqueamento Híbrido com BM25 + FAISS + Cross-Encoder

- Estratégia combinada envolvendo recuperação esparsa e densa, seguida de reranqueamento supervisionado.

- **Objetivo:** aumentar cobertura e precisão na identificação de entidades relevantes, usando diferentes perspectivas de recuperação.

[0.46764] – [JVHeHe - João Vítor, Helio, Henrique]

Fluxo

1. Pré-processamento do corpus

- Título duplicado no conteúdo para reforço semântico (field boosting).

2. Indexação

- **BM25 (Pyserini)**: indexação esparsa com storePositions, storeDocvectors e storeRaw
- **FAISS**: embeddings com msmarco-MiniLM-L-6-v3, indexados com IVFPQ (nlist=4096, m=96, bits=8).

3. Recuperação Inicial

- Top-200 documentos por BM25 e FAISS, unificados por ID.

[0.46764] – [JVHeHe - João Vítor, Helio, Henrique]

Reranqueamento e Resultados

4. Reranking com Cross-Encoder (BAAI/bge-reranker-large)

- Par de entrada: (query, doc), com batches de 32.
- Classificação dos documentos combinados e seleção dos Top-100.

5. Resultados:

- Boa cobertura e precisão com reranker supervisionado robusto.
- Estratégia híbrida se mostrou mais efetiva do que abordagens puramente esparsas ou densas.

Denise: Best Result 0.49266

METHODOLOGY

Three Stage Pipeline

1: HYBRID RETRIEVAL

BM25 + Dense Vectors

(Elasticsearch + all-MiniLM-L6-v2)

2: QUERY EXPANSION

GPT-3.5-Turbo

(3 - 6 variations per query)

3: CROSS ENCODER RERANK

BGE Models

(300 - 1000 candidates)

KEY TECHNICAL COMPONENTS

- Hybrid Fusion: Reciprocal Rank Fusion (RRF) with k = 60
- BM25 Parameters: k1=2.0, b=0.75 with custom text analysis
- Score Combination: α=0.8 weighting toward Cross-Encoder scores
- Hard Negative Sampling: Improved Cross-Encoder training effectiveness

EXPERIMENTAL RESULTS & PERFORMANCE ANALYSIS

Initial evaluation based on the top 100 documents retrieved in the first stage.

Configuration	NDCG@100 (Average)	Recall@100 (Median)	Processing Time (Average)	Improvement (NDCG@100)
BM25 only	0.3896	0.3333	1.0s	0.0658
BM25 + Semantic	0.4697	0.5000	1.6s	0.0809
Full Pipeline (Best)	0.4841	0.5000	6.6s	0.0765

Key Findings:

- **Semantic Search Impact:** 50% recall improvement (0.33 → 0.50)
- **Query Expansion:** 3% NDCG gain but 400% time increase
- **Cross-Encoder Reranking:** Consistent 0.066-0.081 NDCG improvements
- **Performance-Efficiency Trade-off:** BM25+Semantic offers optimal balance

CONCLUSIONS & DEPLOYMENT RECOMMENDATIONS

Optimal Configuration	• BGE-Large reranker • 1000 candidates • 5 query expansion variants • 0.49266 NDCG@100
Production Recommendation	• BM25 + Semantic • 97% of best performance • 24% of computational cost • Optimal balance
Strategic Insights	• Hybrid Retrieval: Most impactful component with reasonable overhead • Recall vs. Ranking: Initial retrieval determines recall; reranking improves precision • Query Expansion: Significant computational bottleneck requiring optimization

REFERENCES

Reciprocal Rank Fusion (RRF)	Cormack, G. V., & Clarke, C. L. A. (2009). Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods. *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '09)*, pp. 758–759. https://doi.org/10.1145/1571941.1572114
Sentence Transformer all-MiniLM-L6-v2	Reimers, N., & Gurevych, I. (2019). **Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks.** *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP).* https://arxiv.org/abs/1908.10084 https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2
Cross Encoder BAAI/bge-reranker-large	Xiao, Y., Liang, Y., & BAAI Team. (2023). **BGE: Bridging the Gap Between Text and Semantic Embeddings with Multi-Stage Supervised Fine-Tuning.** *Preprint on arXiv*. https://arxiv.org/abs/2309.14267 https://huggingface.co/BAAI/bge-reranker-large

Score 0.49736 – Francisco e Lorenzo

RM3

BM25  Cross-encoder

BM25(k1=0.8, b=0.2)
RM3(fb_terms=10, fb_docs=50, original_query_weight=0.5)
Cross-encoder → ms-marco-MiniLM-L-6-v2 com 4000 docs de contexto
Interpolação 80%

Score 0.49736 – Francisco e Lorenzo

Biblioteca pyserini fez o trabalho sujo

Tentativas não bacanas:
- BM25 puro
- CrossEncoder puro
- Interpolação linear (não mudou muito no resultado)
- Remoção de stop words + stemming

Potenciais melhorias:
- Explorar hiperparâmetros
- Testar as combinações das coisas que fizemos
- Melhores modelos de CrossEncoder como o DeBERTa-v3
- SPLADE, RankGPT (Listwise re-ranker), DuoT5 (Pairwise re-ranker)

Referências: https://trec.nist.gov/pubs/trec32/papers/h2oloo.DN.pdf

[0.55626] – José, Luiz, Vinícius

- Toolkit fundamental: **PyTerrier**
- Pré-processamento:
  - Junção de título, body e *keywords* com “\n”.
- Processo de indexação “tradicional”:
  - Tokenizador e stopwords padrão do Terrier, Porter stemmer
- *Retrievers* iniciais:
  - **BM25** e **SPLADE** - junção via **Reciprocal Rank Fusion**
- *Reranker*:
  - Modelo baseado em LLM, ***mxbai-rerank-base-v2***

[0.55626] – José, Luiz, Vinícius

- **Primeiro *Retriever* inicial: SPLADE** (Sparse Lexical and Expansion Model)
  - Utiliza o BERT para gerar um vetor esparso “expandido” para cada documento e a query
  - Vetor resultante contém termos contidos no documento/query, assim como termos semanticamente similares
  - Modelo também atribui um peso para cada termo, indicando sua relevância
  - *f(q,d)* é obtido via produto escalar do vetor do documento e *query*.

Expansão de query

“d-day normandy invasion” → {"normandy": 247.7384, "d": 218.5087, "invasion": 179.3113, "day": 178.8987, "battle": 96.2996, "army": 94.3744, "norman": 85.2171, "war": 78.9227, "france": 66.4753, …. }

Expansão de documento

“H-hour (D-day) \n H-hour was the name given to the…” → {"normandy": 238, "h": 236, "d": 210, "hour": 198, "airborne": 194, "day": 161, "assault": 146, "hours": 132, "invasion": 129, "battle": 128, …}

[0.55626] – José, Luiz, Vinícius

- **Reranker: MixedBread AI Reranker (*mxbai-reranker-base-v2*):**
  - Estudo do estado-da-arte no benchmark **BEIR -** amplo benchmark heterogêneo em diversos domínios diferentes, para testagem de *zero-shot ranking* com a métrica *nDCG@10.*
  - Modelo escolhido: segundo melhor modelo no placar (*nDCG@10:* 55.5)
  - Utiliza como base a LLM **Qwen-2.5 0.5B**, e é treinado para a tarefa de ranqueamento via aprendizado por reforço, sendo usado *cross-encoding*.
  - Método de ranqueamento é **pointwise**, mas o modelo também é treinado utilizando *preference learning*

	nDCG@100	R@1000
BM25 + SPLADE	0.511088	0.839565
SPLADE	0.521023	0.817820
BM25	0.409834	0.750237