

# Documentação Research Challenge – Recuperação de Informação

Francisco Teixeira Rocha Aragão - 2021031726

Lorenzo Carneiro Magalhães - 2021031505

franciscoaragao@protonmail.com

lorenzocarneirobr@gmail.com

Departamento de Ciência da Computação - UFMG

Belo Horizonte, Minas Gerais, Brasil

## Abstract

Máquinas de busca modernas, além de lidarem com a tarefa de indexar um enorme corpus de documentos, precisam também retornar rankings de qualidade para as queries do usuário. Desse modo, o presente trabalho foca nessa tarefa de produzir rankings de qualidade utilizando diferentes técnicas vistas na disciplina.

## CCS Concepts

• Information systems → Information retrieval.

## Keywords

ranking, recuperação, informação

### ACM Reference Format:

Francisco Teixeira Rocha Aragão - 2021031726 and Lorenzo Carneiro Magalhães - 2021031505. 2018. Documentação Research Challenge – Recuperação de Informação. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Recuperação de Informação - RC)*. ACM, New York, NY, USA, 3 pages. <https://doi.org/XXXXXXX.XXXXXXX>

## 1 Introdução

O trabalho de pesquisa proposto na disciplina de recuperação de informação implementa um sistema que, dado um corpus de documentos e as queries do usuário, retorna um ranking de documentos relevantes. Como a execução do trabalho está ligada a uma competição, o objetivo é retornar rankings com melhores resultados de nDCG. Dessa forma, as próximas seções explicam as estratégias utilizadas em cada submissão.

## 2 Primeira submissão

Identificação: submission.csv

Score: 0.37940

A primeira submissão foi feita seguindo abordagens simples vistas na disciplina, envolvendo pré-processamento dos textos, indexação e ranqueamento. Como era a primeira submissão, idealmente o bm25 seria um bom baseline de resultados já que é uma técnica clássica e com bom desempenho. Primeiramente, o corpus é lido e pré-processado com algumas funções simples, como aplicar *lower*

case, tokenização, remoção de stopwords e de tokens muito curtos (que possuem 1 caractere). Dessa forma, é salvo internamente duas estruturas para serem usadas como índice: a lista de IDs de cada entidade além dos tokens de cada documento para serem utilizados durante o ranqueamento. Assim, durante a leitura das queries, cada uma delas é pré-processada para então o algoritmo de Bm25 ser utilizado. Com isso, os scores são salvos e retornados posteriormente.

Vale destacar que essa submissão foi bem simples e utilizou conceitos clássicos da matéria, sem realizar qualquer tipo de treinamento. Foram utilizadas as bibliotecas nltk para pré-processamento, além de rank\_bm25 para realizar o ranqueamento com o algoritmo Bm25. O resultado foi interessante dado a simplicidade do método e do código desenvolvido.

## 3 Segunda submissão

Identificação: submission.csv

Score: 0.50639

A segunda submissão tentou explorar técnicas um pouco mais avançadas dado a estratégia aplicada no primeiro método. Dessa forma, agora foram utilizadas estratégias mais robustas, como o método rm3 e cross encoder para reranking. Essa abordagem vai de encontro as aulas que explicam como expansão de queries auxilia a aproximar as queries dos documentos além de facilitar o matching de vocabulário (o que acaba auxiliando o bm25), juntamente a técnica de reranking que é o "estado da arte" de recuperação de informação.

Assim, foi-se utilizado a abordagem com Rm3, que é um modelo para expansão de consultas. Ademais, também foi adotada o reranking com o modelo ms-marco-MiniLM-L12-v2 sendo utilizado, que é um modelo treinado com linguagem natural focado em tarefas de recuperação de informação e question answering. Com sua abordagem de Cross Encoder, o modelo recebe tanto a query quanto textos dos documentos como input, resultando em um bom entendimento sobre todas as informações. Vale destacar que essa parte de reranking é um pouco custosa e demorou um certo tempo, comparado ao bm25 puro da primeira submissão.

Desse modo, o processo foi feito da seguinte forma. Inicialmente, como estamos trabalhando com métodos mais robustos, a etapa de pré-processamento não foi feita para essa submissão pois idealmente os modelos deveriam ser capazes de entender bem os textos. Assim, inicialmente os modelos são carregados e seus parâmetros definidos (bm25 com  $k1 = 0.92$  e  $b=0.36$ , que foram utilizados pensando que o corpus não tem documentos muito longos, além de rm3 com até 10 termos adicionados nas consultas, sendo utilizados 50 documentos para coleta de informações para expansão). Os valores utilizados não são exatamente muito comuns mas também não são aleatórios, no

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

Recuperação de Informação - RC, Junho 23, 2025, Belo Horizonte, MG

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-XXXX-X/18/06

<https://doi.org/XXXXXXX.XXXXXXX>

geral foram valores idealmente ajustados ao problema e que também tiveram uma performance razoável durante os testes realizados.

Vale destacar que os modelos foram carregados sobre os arquivos de índice gerados sobre o utilitário Anserini. Como a biblioteca Pyserini foi utilizada para trabalhar com os modelos (rm3 e bm25), o formato do corpus deveria ser padronizado para a estrutura: {id: ..., contents: ...}. Assim, após esse processamento feito, a biblioteca Pyserini é chamada para realizar a criação do índice no formato correto. Dessa forma, a biblioteca que acabou cuidando de toda a tarefa de indexação, não sendo necessário maiores esforços nessa parte.

Após esse processamento inicial e padronização do corpus em um índice, o ranqueamento começa. O processo é bem simples no geral, com o bm25 sendo executado sobre a query, retornando K melhores documentos. Em sequência, o modelo de reranking é utilizado sobre essa amostra para então retornar os 100 melhores resultados para a query. Por fim, os resultados são interpolados a partir de um valor alpha, para então serem salvos e submetidos na competição. Essa estratégia foi utilizada pois como visto em aula e também percebido na prática, o modelo de reranking é muito custoso. Assim, sua utilização sozinha sobre todo o corpus é muito custoso, praticamente inviável, sendo melhor o seu uso sobre um espaço reduzido de documentos.

Como dito anteriormente, as bibliotecas utilizadas foram a pyserini para cuidar do corpus e dos métodos clássicos, além de sentence\_transformers para trabalhar com o modelo de cross encoder. O valor de K testado foi de 1000 documentos e alpha como 0.7 (dando mais peso para o modelo de reranking do que o bm25 + rm3). É válido comentar como essa abordagem mostrou-se muito melhor na prática do que utilizar apenas o bm25. Como muitas variáveis foram alteradas, não dá para dizer de momento o que foi mais impactante, se foi a nova estrutura de índice, o modelo de reranking ou a expansão de queries, porém toda essa junção de técnicas ajudou na melhoria dos resultados.

## 4 Terceira submissão

Identificação: submission3.csv

Score: 0.49218

A terceira submissão seguiu a mesma lógica da segunda, porém teve apenas alguns parâmetros alterados para testar diferentes ideias. Como a última submissão obteve um bom resultado no ranking da competição, a ideia era justamente tentar observar se a mudança nos parâmetros iria conseguir trazer bons resultados. Primeiramente, foi feito o pré-processamento do texto de modo semelhante ao que foi explicado na primeira submissão (já que esse processo não foi realizado na segunda submissão). Além disso, os parâmetros do Bm25 foram alterados para valores padrão, com  $k_1 = 1.2$  e  $b = 0.75$ . Ademais, o número de documentos para o reranking foi reduzido para apenas 500 (o que melhorou a performance de execução) e o peso do alpha foi alterado para 80% (ou seja, 80% do modelo de reranking e 20% do bm25). Nenhuma mudança foi feita na lógica do processo, somente a alteração dos parâmetros utilizados. Percebe-se que não houve muita diferença na prática, mesmo que houve uma piora dos resultados, não foi também algo muito expressivo.

## 5 Quarta submissão

Identificação: submission4.csv

Score: 0.48224

Agora na quarta submissão, seguindo a mesma ideia das submissões anteriores, nada foi alterado em relação a lógica de execução da tarefa. A única modificação foi a utilização do score completo do modelo cross-encoder para realizar o ranking dos 100 melhores documentos. O bm25 ainda é utilizado para gerar os melhores 1000 documentos, porém o score final é definido somente pela LLM. Isso foi feito para observar o quanto bom ficaria o modelo mais robusto sozinho, sendo um baseline de comparação com a terceira submissão. Ao observar os resultados, foi possível notar um pior score em comparação a submissão anterior, mostrando que embora o uso do modelo LLM seja útil, não pode ser utilizado cegamente para a tarefa.

## 6 Quinta submissão

Identificação: submission5.csv

Score: 0.526

Na quinta submissão, a ideia foi a mesma que as demais, no entanto utilizamos um modelo levemente (ou praticamente nada) inferior, mas que executava mais rapidamente, o que permitiu uma passagem de mais contexto para o modelo. O modelo utilizado foi o ms-marco-MiniLM-L6-v2, que executa 2 vezes mais rapidamente que o modelo anterior, de maneira que permitiu passar um modelo um contexto mais rico com 4000 documentos.

Ademais, também alteramos os parâmetros do bm25 para tentar adequar os algoritmos ao pequeno tamanho dos documentos. Os valores finais foram:  $\text{bm25}(k_1=0.8, b=0.2)$ .

## 7 Ideias não testadas

Por fim, vale destacar que outras estratégias foram testadas localmente, porém sem sucesso. Como exemplo, abordagens envolvendo modelos doc2query foram testadas, em que um trecho do documento é passado ao modelo e então é gerado um query que o representa, facilitando assim a etapa de matching com as queries. Entretanto, por motivos práticos, essa abordagem não seguiu adiante, sendo necessário maior poder computacional para cumprir a tarefa, que iria demorar muito em um ambiente computacional sem GPU. O mesmo problema também afeta abordagens learning to rank em que ocorre o aprendizado do modelo, que dependem de maior poder computacional para terem sucesso.

## 8 Execução

Todo o código foi sumariado em um notebook Jupyter, sendo separados para cada submissão feita. Vale destacar que é necessário fazer algumas etapas de instalação conforme mencionado no decorrer do documento, além das bibliotecas presentes no arquivo "requirements" enviado junto ao código. Vale destacar que é necessário a instalação do Java para correto funcionamento do código devido à presença do pyserini.

Adicionalmente, é importante alertar que rodar todas as células podem levar um tempo considerável.

## 9 Conclusão

Percebe-se ao fim da tarefa, como o problema é complicado e necessita de diferentes abordagens para ser solucionado com eficiência. Vale destacar que embora diferentes métodos tenham sido testados para cumprir a tarefa, novas abordagens ainda podem ser utilizadas, como exemplo, a utilização de enriquecimento das informações dos documentos com outros LLMs, ou utilização de novos métodos e heurísticas para ranquear os documentos.

## 10 Referências

### References

- [1] Elastic. *Escolha do valor de K1 e B do BM25*. Disponível em: <https://www.elastic.co/blog/practical-bm25-part-3-considerations-for-picking-b-and-k1-in-elasticsearch>. Acesso em: 24 jun. 2025.
- [2] Hugging Face. *Cross-Encoder models*. Disponível em: <https://huggingface.co/cross-encoder>. Acesso em: 24 jun. 2025.
- [3] SBERT – Sentence Transformers. *Modelos pré-treinados Cross-Encoder*. Disponível em: [https://www.sbert.net/docs/cross\\_encoder/pretrained\\_models.html](https://www.sbert.net/docs/cross_encoder/pretrained_models.html). Acesso em: 24 jun. 2025.
- [4] Pyserini – Castorini Group. *MS MARCO Passage Ranking – v1*. Disponível em: <https://castorini.github.io/pyserini/2cr/msmarco-v1-passage.html>. Acesso em: 24 jun. 2025.