Research Challenge - Entity Search

Vinicius Faria*
vlcfaria@ufmg.br
Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais, Brasil

José Massucato*
josedmass@ufmg.br
Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais, Brasil

Luiz Rocha*
luizfgr@ufmg.br
Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais, Brasil

ABSTRACT

Este trabalho teve como objetivo principal avaliar as habilidades dos estudantes em engenharia de busca dado o conhecimento adquirido durante o curso de Recuperação da Informação. O desafio envolveu, dado uma base de conhecimentos textual de entidades, elaborar um sistema de busca que retorne, dado uma query do usuário, uma quantidade de documentos relevantes em ordem de relevância. Para isso, diversas formas de pré-processamento, indexação, busca inicial e *reranking* foram utilizadas.

1 INTRODUÇÃO

Como visto no decorrer do curso, a área de Recuperação de Informação possui diversos domínios de aplicação e diversas técnicas/métodos que podem ser aplicados com diferentes finalidades. Uma das áreas mais fundamentais nesse domínio é na construção de *pipelines* de busca onde, dado uma base de conhecimento e uma busca de um usuário, deve ser fornecido documentos relevantes à consulta, ordenados por ordem de relevância. Fazer um pipeline pode ser uma tarefa complexa, envolvendo diversas decisões de projeto, sendo necessário estabelecer um pré-processamento da base de conhecimento fornecida, o estabelecimento de um *retriever* inicial, com objetivo de obter um conjunto de documentos candidatos, juntamente com um *reranker* para refinar a ordenação dos documentos que serão fornecidos.

Esse trabalho visa criar um *pipeline* completo de busca em uma base de conhecimento de entidades nomeadas da *Wikipedia*, com métrica de avaliação principal o *nDCG@100*, que avalia a ordenação dos documentos relevantes fornecidos, comparando-a com a ordenação "ótima".

2 SUBMISSÕES

Para (quase) todas as submissões enviadas, foi utilizado a biblioteca PyTerrier [9] para a indexação do corpus, busca e reranking, dado a extensibilidade e flexibilidade da biblioteca, que trouxe juntamente consigo diversas implementações de técnicas mais recentes utilizadas na área. Com a biblioteca e o Corpus inicial em mãos, foram experimentadas diversas técnicas de pré-processamento do corpus, indexação, retrieval e *reranking*.

Antes de entrar em detalhes específicos de cada submissão, nota-se que, como diversas técnicas próximas do estado-da-arte foram utilizadas, as hipóteses de cada submissão estavam mais focadas em quais modelos/técnicas da criação do índice ou *reranking* a serem utilizadas. Detalhes como stemming e tokenização muitas vezes já são fixados dependendo da técnica a ser utilizada, logo esses detalhes não serão discutidos em detalhes em cada submissão. Apesar disso, esses detalhes foram explorados no decorrer do trabalho, porém tiveram resultados piores do que as top 5 submissões, que forneceram resultados bem altos. Algumas das técnicas mais "tradicionais" exploradas foram:

- Tokenização e lemmatization do corpus através da biblioteca spaCy [10]
- Pipeline utilizando LambdaMART com *field-based index* e *scoring*, utilizando features como BM25 do título, body e keywords
- *Query rewriting* com pseudo-relevância (RM3) e dependência sequencial.
- Tokenizador, stemming e stopwords padrão do sistema de busca *Anserini*, utilizando a biblioteca *Pyserini* [8]

Em particular, notou-se que o tokenizador *default*, *porter stemmer* e stopword list, empregados por padrão no Terrier se demonstraram os mais eficazes em questão da métrica *Recall@1k* do que tokenizadores e lemmatization mais sofisticados do spaCy, logo optou-se por utilizar ele para o restante dos testes. Além disso, *query rewriting*, mesmo com tunamento dos parâmetros do RM3, não se demonstrou muito efetivo, aumentando o *Recall@1k* em apenas 1% no dataset de treino.

A melhor métrica *nDCG@100* obtida usando técnicas "tradicionais", no dataset público do Kaggle foi utilizando o lambdaMART, utilizando features como BM25 do título, texto e keywords, tamanho da *query* e do documento, além de diversos outros modelos de *scoring* disponibilizados pelo Terrier (PL2, DPH, etc.). Apesar disso, apesar da técnica se apresentar melhor que o BM25 com um *nDCG@100* de 0.44, ainda houve uma diferença muito grande com o uso das técnicas mais avançadas a serem discutidas a seguir.

A ordem das submissões a seguir serão reportadas seguindo a efetividade da métrica *nDCG@100* do dataset **público** no Kaggle.

2.1 Submissão 5

- Identificação: SPLADE-submit-results.csv
- *nDCG@100*: 0.51081

Ao decorrer do trabalho, notou-se na ineficiência do retriever inicial utilizando BM25 em relação ao *Recall@1k*, mesmo com técnicas de *query rewriting* como o RM3. Em geral, utilizando tokenização, stemming e stopwords padrão do Terrier, o BM25 e BM25+RM3 obtinham 0.75 e 0.76 no *Recall@1k*, respectivamente. Visando melhorar o retriever inicial para um futuro *reranker*, a hipótese principal era encontrar algum método para obter uma representação semântica melhor da *query* e dos documentos em vez de matching exato ou pseudo-relevância.

Com o estudo da literatura, foram determinadas 3 técnicas candidatas ao retrieval incial: *doc2query* [12], *dense retrieval* e o *SPLADE* (Sparse Lexical and Expansion Model for First Stage Ranking) [5]. Testamos o uso de *dense retrieval* com o modelo *sentence-transformers/all-MiniLM-L6-v2*, que surpreendentemente trouxe resultados piores do que o BM25. Logo, optou-se pelo uso *SPLADE* como forma de preencher a lacuna lexical entre o documentos e a *query*, além da possibilidade do modelo generalizar melhor devido ao uso de um índice esparso comum.

Para obter o resultado fornecido, tanto os documentos quanto as *queries* foram expandidos através do *SPLADE*, com todos os valores textuais do documento juntadas, separados apenas por quebras de linha. Em resumo, o *SPLADE* utiliza o encoder do BERT [4] dado o documento/query de entrada, porém, em vez de utilizar um vetor denso como representação final, utiliza-se um Masked Language Model para prever palavras semanticamente relevantes dado o vetor denso. Com isso, é obtido um vetor *bag-of-words* contendo termos diretamente (palavras que estão no documento) ou semanticamente relevantes, atribuindo um peso conforme a relevância prevista de cada termo. Como exemplo, a query *"d-day normandy invasion"*, quando expandida, também conteria os termos: *war*, *battle* e *army*, que não estão presentes na query, mas são relevantes ao contexto dela.

Nessa submissão, foi utilizado apenas o retrieval inicial (sem reranking) utilizando o scoring padrão do *SPLADE* (equação 1), onde V é o vocabulário, e Q_i e D_i é o peso atribuído para a palavra i da query e do documento, respectivamente. O *nDCG@100* obtido, além da métrica *Recall@1k* de 0.81 obtida, consolidou o *SPLADE* como ótimo first retriever.

$$sim(q, d) = \sum_{i=1}^{V} Q_i D_i \tag{1}$$

2.2 Submissão 4

- Identificação: t5-CORRECTED.csv
- *nDCG@100*: 0.51294

Nessa submissão, queríamos ver o potencial do BM25 como retriever inicial com um *reranker* mais elaborado. Para isso, foi utilizado o *monoT5* [11], um reranker *pointwise* baseados em transformers *text-to-text*. Essa arquitetura codifica a query e os documentos conjuntamente (cross-enconding), e o modelo é treinado a prever "yes" e "no" dependendo da relevância do documento em relação a query, e a probabilidade definida para o token "true" é utilizada como o score para o documento. Como a técnica é *pointwise*, nem sempre a relação de relevância é tão boa quanto estratégias *pairwise* e *listwise*, mas esse passo inicial foi dado como suficiente.

Para obter a métrica fornecida, os campos textuais do documento foram transformados em um só novamente, separados por quebra de linha (título \n body \n keywords) e utilizados o tokenizador e stopwords padrão do Terrier, assim como o Porter stemmer. O modelo do monoT5 utilizado foi 'castorini/monot5-base-msmarco', treinado no MS-MARCO [1]. O retrieval inicial foi feito com o BM25, e os 1000 primeiros documentos foram rerankeados com o monoT5.

Deve-se salientar que também foi utilizado o duoT5 [13], que executa reranking *pairwise* nos top 100 documentos após o monoT5, porém este apresentou resultados piores do que quando só o mono é utilizado. Concluiu-se que o duoT5 é mais sensível ao contexto/domínio inicial do treinamento.

2.3 Submissão 3

- Identificação: SPLADE-MONOT5-submit-results.csv
- *nDCG@100*: 0.51644

Nessa submissão, queríamos fazer a mesma coisa que fizemos na submissão 4, porém agora com o *SPLADE* como retriever inicial. Utilizou-se o mesmo índice e as mesmas técnicas de expansão da query da submissão 5. Os resultados foram similares aos da submissão 3, visto que o modelo do monoT5 utilizado foi o mesmo, sendo o único fator diferente uma seleção inicial do *SPLADE*, que tem *Recall@1k* maior.

2.4 Submissão 2

- Identificação: splade-colbert.csv
- *nDCG@100*: 0.52275

Inicialmente, queríamos testar outros retrievers iniciais para comparar com o *SPLADE*, em específico o ColBERT [6]. O ColBERT (Contextualized Late Interaction over BERT) é um retriever inicial com resultados estado-da-arte conhecido por ser eficiente (em questão de tempo) através do seu mecanismo de "late interaction": o documento e a query são processados de forma independente, gerando um embedding (vetor denso) por token, e, por fim, é utilizado a operação *MaxSim* para cada token i da query, que seleciona o valor do maior *dot product* resultante entre o embedding do token da query E_{q_i} com todos os outros embeddings dos tokens do documento E_{d_j}. O score final é então a soma de todos os *MaxSim* entre os tokens da query (equação 2).

$$sim(q, d) = \sum_{i \in [|E_q|]} max_{j \in [|E_d|]} E_{q_i} \cdot E_{d_j}^T \tag{2}$$

O ColBERT pode ser visto, a grosso modo, como um "bag-of-words" de embeddings contextualizados, e com isso ele evita operações custosas como *cross-encodings* entre o documento e a query, o que é inviável para o retrieval inicial. Por outro lado, manter um embedding por token é extremamente custoso em questão de memória, e o ColBERT não conseguiu ser executado no corpus inteiro, o que exigiria por volta de 100GB de memória.

Para combater o tamanho grande do corpus, utilizou-se uma "filtragem" inicial do corpus: para as queries testadas, extraiu-se os primeiros 1000 documentos selecionados pelo *SPLADE* e o BM25 (usando as técnicas de indexação das submissões anteriores), gerando um corpus de 339k documentos. A partir desse corpus, os campos textuais foram juntados com quebras de linha (\n) novamente, e então indexados utilizando o ColBERT e feito o retrieval inicial novamente, fornecendo o resultado visto acima. A indexação e retrieval por parte do ColBERT foi feita utilizando a biblioteca RAGatouille [2].

Reconhecemos que essa forma não é uma maneira muito formal de rerankear documentos, sendo mais apropriado passar, para cada query, a lista de documentos obtidos pelo SPLADE e BM25 apenas daquela query em específico. De qualquer forma, a hipótese desse teste era só verificar a eficácia e qualidade do retrieval inicial do ColBERT dados os limites de memória, e, apesar de ser feita uma espécie de rerankeamento "cego" entre diversas queries, o ColBERT se demonstrou efetivo.

2.5 Submissão 1

- Identificação: SPLADE-mxbai-submit-results.csv
- *nDCG@100*: 0.54743

Com a efetividade e simplicidade do *SPLADE* como initial retrieval comparado com o BM25 e o ColBERT, decidiu-se fazer um pipeline de busca mais complexo, mirando em um retriever com *Recall*@1k alto que então seria passado para um reranker

ainda mais elaborado e generalizável que o monoT5. Para isso, foi pesquisado o atual estado da arte do benchmark BEIR [15], um benchmark altamente heterogêneo elaborado para análise da generalização (zero-shot) de sistemas de busca, sendo um dos domínios do benchmark o próprio *Entity Retrieval*, que estamos fazendo. Observou-se que os resultados anteriores eram muito parecidos com a métrica *nDCG*@10 reportadas para pipelines similares aos elaborados nas submissões anteriores.

Analisando o atual estado na arte no BEIR, optou-se pelo uso do modelo *mxbai-rerank-base-v2* [7] como reranker, que atualmente possui o segundo maior *nDCG*@10 do BEIR, perdendo apenas para o seu modelo maior, *mxbai-rerank-large-v2*. O modelo utiliza uma LLM pequena, treinada a partir do *Qwen2.5-0.5B* [14] (500M parâmetros) utilizando aprendizado por reforço para aprender a realizar scorings de relevância para pares de documento-query. O treinamento do modelo foi separado em 3 estágios: primeiro, o modelo é treinado para fazer classificações binárias de relevância, para, em seguida, ser treinado a entender mais profundamente relações semânticas entre documentos e queries, e, finalmente, é feito *preference learning*, em que o modelo é treinado a rankear os documentos mais relevantes mais alto. Os autores afirmam que o paper técnico ainda está sendo feito, mas acreditamos que, apesar do rankeamento utilizado ser *pointwise*, o modelo foi treinado de certa forma para aprender implicitamente a ordenar "globalmente" os documentos passados através de *preference learning*, o que o favorece em relação a outras estratégias de reranking atuais como o RankT5 [17], ListT5 [16] e duoT5 [13].

Com o reranker estabelecido, como ele é considerado *pointwise*, e portanto não é sensível ao ranqueamento inicial, o retriever inicial foi elaborado para ter o máximo possível de *Recall*@1*k*. O *SPLADE*, como visto anteriormente, é ótimo em relação à essa métrica, porém também deve ser notado que seu vocabulário é fixo, e ele pode julgar de forma errada os pesos de cada termo. Com esse detalhe em mente, achou-se interessante ainda manter de alguma forma o score da query original feita pelo usuário. Logo, para o retrieval inicial, foram utilizados dois índices: o índice expandido do *SPLADE* (submissão 5), em que o retrieval é feito com o scoring do *SPLADE* (eq. 1), e o índice original (submissão 4) em que o retrieval é feito com o BM25. O retrieval do SPLADE e do BM25 retornam até 1000 documentos cada, e então é utilizado Reciprocal Rank Fusion [3] para combinar os resultados de ambos retrievers, fornecendo novamente 1000 resultados. A fusão de ambos os retrievers se demonstrou extremamente eficaz, aumentando o *Recall*@1*k* para 0.85 no dataset de treino e fornecendo mais documentos relevantes para o *reranker*.

Finalmente, os 1000 resultados da fusão são passados para o reranker estabelecido, fornecendo a métrica visualizada acima, que é inclusive bem próxima do resultado do modelo no BEIR (55.5). Deve-se destacar que esse modelo de reranking, apesar de extremamente eficaz, é bem demorado, demorando cerca de 2 minutos para extrair os top 100 documentos em uma GPU T4. Apesar disso, o pipeline completo desenvolvido foi considerado muito eficaz para a métrica do trabalho. Uma visualização final do pipeline está presente na imagem 1.

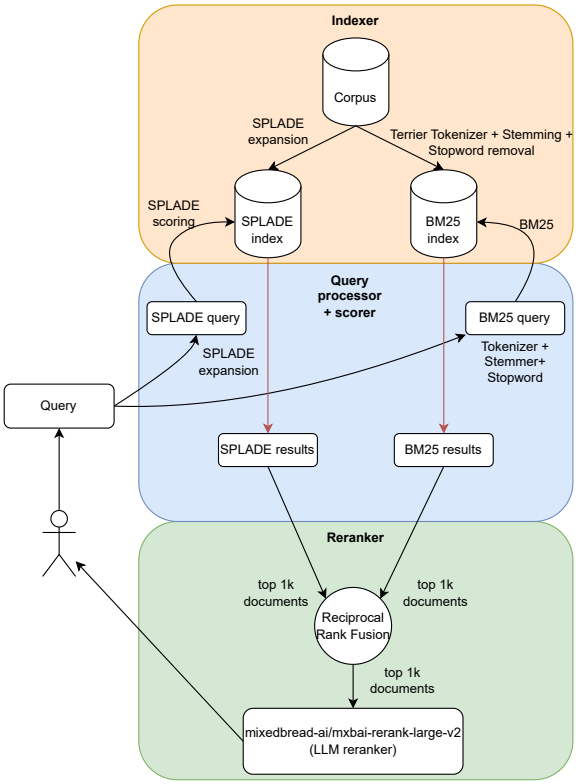

Figure 1: Pipeline final da quinta submissão. São utilizados o *SPLADE* e o BM25 como retrievers iniciais, cujos resultados são mesclados e então passados para o reranker *mxbai-rerank-base-v2*

3 CONCLUSÃO

Ao decorrer deste trabalho, foram testados diferentes técnicas e *pipelines* de busca de forma a otimizar a métrica *nDCG@100* a partir de uma base de entidades nomeadas da *Wikipedia*, e uma série de *queries*. Inicialmente, fez-se um estudo utilizando técnicas mais "tradicionais" da área, como a escolha de diferentes pré-processamentos, tokenizadores, *stemmers/lemmatizers*, retrievers iniciais e features para reranqueamento com técnicas de *Learning to Rank*. Com um estudo inicial com tais técnicas, migrou-se para métodos mais recentes com a popularização de *Transformers* como o *BERT* e o *T5*, tanto para retrieval inicial quanto reranqueamento.

Dado que não foi feito *fine-tuning* para nenhum dos modelos, o *SPLADE* se demonstrou extremamente efetivo para expansão de documentos e *queries* nesse contexto, pois generalizou bem para nossos dados e produz um índice esparso, o que é muito barato em questão de memória e processamento para buscas futuras. Para retrieval inicial foi então utilizado o índice produzido pelo *SPLADE* assim como o índice "comum" (com tokenização, stemming e stopword removal do Terrier) para conciliar a expansão semântica do *SPLADE* com o

vocabulário original do documento e da query, mesclando os resultados obtidos pelo scorer do *SPLADE* com o BM25 via *Reciprocal Rank Fusion* para maximizar o *Recall@1k*. Com uma cobertura grande de nos 1000 primeiros documentos, empregou-se um reranker *point-wise* estado-da-arte no benchmark do BEIR, utilizando o modelo *mxbai-rerank-base-v2*, baseado na LLM Qwen2.5-0.5B e treinado para *reranking* via aprendizado por reforço. Obteu-se então, um pipeline muito efetivo para o trabalho proposto, visto que cada pedaço do pipeline conseguiu resultados excepcionais nas métricas desejadas: *Recall* para o retriever, *nDCG* para o reranker.

REFERENCES

[1] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. arXiv:1611.09268 [cs.CL] https://arxiv.org/abs/1611.09268

[2] Benjamin Clavié. 2024. RAGatouille: Bridging the gap between state-of-the-art Retrieval research and RAG applications. https://ragatouille.clavie.eu Copyright © 2024 Benjamin Clavié.

[3] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. 2009. 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* (Boston, MA, USA) *(SIGIR '09)*. Association for Computing Machinery, New York, NY, USA, 758–759. https://doi.org/10.1145/1571941.1572114

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL] https://arxiv.org/abs/1810.04805

[5] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. arXiv:2107.05720 [cs.IR] https://arxiv.org/abs/2107.05720

[6] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. arXiv:2004.12832 [cs.IR] https://arxiv.org/abs/2004.12832

[7] Xianming Li, Aamir Shakir, Rui Huang, Julius Lipp, and Jing Li. 2025. ProRank: Prompt Warmup via Reinforcement Learning for Small Language Models Reranking. *arXiv preprint arXiv:2506.03487* (2025).

[8] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2021)*. 2356–2362.

[9] Craig Macdonald and Nicola Tonellotto. 2020. Declarative Experimentation inInformation Retrieval using PyTerrier. In *Proceedings of ICTIR 2020.*

[10] Ines Montani, Matthew Honnibal, Matthew Honnibal, Adriane Boyd, Sofie Van Landeghem, and Henning Peters. 2023. explosion/spaCy: v3.7.2: Fixes for APIs and requirements.

[11] Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. arXiv:2003.06713 [cs.IR] https://arxiv.org/abs/2003.06713

[12] Rodrigo Nogueira, Wei Yang, Jimmy Lin, and Kyunghyun Cho. 2019. Document Expansion by Query Prediction. arXiv:1904.08375 [cs.IR] https://arxiv.org/abs/1904.08375

[13] Ronak Pradeep, Rodrigo Nogueira, and Jimmy Lin. 2021. The Expando-Mono-Duo Design Pattern for Text Ranking with Pretrained Sequence-to-Sequence Models. arXiv:2101.05667 [cs.IR] https://arxiv.org/abs/2101.05667

[14] Qwen Team. 2024. Qwen2.5: A Party of Foundation Models. https://qwenlm.github.io/blog/qwen2.5/

[15] Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. arXiv:2104.08663 [cs.IR] https://arxiv.org/abs/2104.08663

[16] Soyoung Yoon, Eunbi Choi, Jiyeon Kim, Hyeongu Yun, Yireun Kim, and Seung won Hwang. 2024. ListT5: Listwise Reranking with Fusion-in-Decoder Improves Zero-shot Retrieval. arXiv:2402.15838 [cs.IR] https://arxiv.org/abs/2402.15838

[17] Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2022. RankT5: Fine-Tuning T5 for Text Ranking with Ranking Losses. arXiv:2210.10634 [cs.IR] https://arxiv.org/abs/2210.10634