Avaliação de Modelos de Recuperação de Informação Utilizando Biblioteca de Busca por Similaridade FAISS

José Carlos Ferreira Neto[1]

[1]Programa de Pós-Graduação em Engenharia de Sistemas e Automação – Universidade Federal de Lavras (UFLA)

Abstract. *Information Retrieval (IR) is an area of computer science that explores techniques of representation, storage, organization and access to information. RI models seek to satisfy user needs by retrieving documents that meet their request. This work presents a comparison of IR modeling techniques, combining vector representation generated through TF-IDF and BERT. These strategies are evaluated with the Cystic Fibrosis (CF) collection.*

Resumo. *A Recuperação de Informação (RI) é uma área da computação que explora técnicas de representação, armazenamento, organização e acesso a informação. Os modelos de RI buscam satisfazer as necessidades do usuário, recuperando documentos que atendem a sua solicitação. Este trabalho apresenta uma comparação de técnicas de modelagem de RI, combinando representação vetorial gerada através do TF-IDF e do BERT. Estas estratégias são avaliadas com a coleção Cystic Fibrosis.*

1. Introdução

A Recuperação de Informação (RI) é uma área da computação que explora técnicas de representação, armazenamento, organização e acesso a informação. Sendo que, os objetivos principais desta área são propor e gerar boas técnicas de indexação de texto e de mecanismos de se realizar consultas, em busca de alguma informação, em uma coleção de documentos [Baeza-Yates et al. 1999].

Os modelos de RI buscam satisfazer as necessidades do usuário, recuperando documentos que atendem a sua solicitação. Para isso, estes modelos atribuem uma pontuação numérica para cada um dos documentos e os classifica de acordo com esta pontuação. Essa pontuação indica o grau de relevância do documento em relação a consulta do usuário [Singhal et al. 2001].

Este trabalho apresenta uma comparação de algumas técnicas de modelagem aplicadas ao coleção *Cystic Fibrosis*. Esta coleção conta com 1.239 documentos publicados entre os anos de 1974 a 1979, e outros dados referentes a cada uma destas publicações.

A seção 2 aborda sobre as técnicas de representação vetorização dos textos utilizadas. A seção 3 apresenta as combinações das técnicas utilizadas para compor cada uma das estratégias. A seção 4 apresenta as características do *hardware* e das ferramentas utilizadas. A seção 5 apresenta os resultados obtidos. Por fim, a seção 6 aponta as conclusões sobre os resultados obtidos.

2. Representação Vetorial

As máquinas de busca podem ser definidas, segundo [Croft et al. 2010] como sendo a utilização de técnicas de recuperação de informação (RI) para coleções de texto.

Para o desenvolvimento de um sistema de máquina de busca é necessário definir o tipo de representação vetorial a ser aplicado aos documentos da coleção. Várias são as formas de se gerar estas representações, desde estratégias relacionado a frequência dos termos em um documento até estratégias baseadas em aprendizado de máquina.

O *Term Frequency – Inverse Document Frequency* (TF-IDF) é uma técnica dentro da modelagem vetorial, que consiste em produzir uma representação vetorial de um texto, e esta estratégia é baseada em pesos. O componente TF baseia-se na frequência de cada palavra em cada documento da coleção, e o componente IDF está relacionado com a quantidade de documentos que uma palavra aparece [Croft et al. 2010].

O documento pode ser representado pelo vetor $\vec{d_j} = (w_{1,j}, w_{2,j}, ..., w_{i,j})$, sendo que o componente tf do termo k_i no documento d_j pode ser obtido por:

$$tf_{i,j} = \begin{cases} 1 + log f_{i,j} & se\ f_{i,j} > 0 \\ 0 & caso\ contrário \end{cases} \tag{1}$$

Onde $f_{i,j}$ representa a frequência de ocorrência de um termo k_i no documento d_j. Já o IDF pode ser obtido por:

$$idf_i = log \frac{N}{n_i} \tag{2}$$

Onde N é o total de documentos em uma coleção e n_i o número de documentos que o termo k_i aparece. Portanto, a combinação dos componentes TF e IDF produz pesos conforme a equação a seguir:

$$w_{i,j} = \begin{cases} (1 + log f_{i,j}) \times log \frac{N}{n_i} & se\ f_{i,j} > 0 \\ 0 & caso\ contrário \end{cases} \tag{3}$$

Outra forma de modelagem é a modelagem semântica. Uma das técnicas desse tipo de modelagem, sendo esta mais recente, é utilizar o modelo BERT (*Bidirectional Encoder Representations from Transformers*) para produzir vetores para as palavras captando o seu sentido no contexto a qual esta está inserida.

O BERT é um modelo pré-treinado desenvolvido pela Google utilizado para tarefas de Processamento de Linguagem Natural (PLN). Os dados de pré-treinamento utilizados extraídos do *BooksCorpus* (800 milhões de palavras) e do *Wikipedia* (2.500 milhões de palavras) [Devlin et al. 2018].

Entretanto, o BERT é restrito a vetorização de palavras, por conta disso, pensando em escalar o mesmo conceito e gerar vetores (*embeddings*) para toda uma sentença, em [Reimers and Gurevych 2019] é proposto uma modificação deste modelo, dando origem ao *sentece*-BERT. Os autores explicam que, o *sentece*-BERT utiliza um codificador que pode converter passagens mais longas de texto em vetores. Este sistema é bastante útil para busca semântica de documentos, uma vez que sentenças semelhantes semanticamente estão próximas umas das outras no espaço vetorial.

3. Estratégias Adotadas

Para proceder com a comparação dos modelos de busca, oito combinação de abordagens foram utilizadas, isto é, oito estratégias de modelos de busca foram implementados. A Figura 1 apresenta um fluxograma completo com todas as abordagens que foram utilizadas para compor as estratégias adotas.

Todas as estratégias passam pelas etapas elencadas na Figura 1 com diferença nas abordagens utilizadas. As estratégias são:

1. Não há pré-processamento, vetorização via BERT e indexação *Flat*;
2. Não há pré-processamento, vetorização via BERT e indexação *IVFFlat*;
3. Normalização das palavras, vetorização via TF-IDF e indexação *Flat*;
4. Normalização das palavras, vetorização via TF-IDF e indexação *IVFFlat*;
5. Remoção de *stopwords*, vetorização via BERT e indexação *Flat*;
6. Remoção de *stopwords*, vetorização via BERT e indexação *IVFFlat*;
7. Normalização das palavras, remoção de *stopwords*, vetorização via TF-IDF e indexação *Flat*;
8. Normalização das palavras, remoção de *stopwords*, vetorização via TF-IDF e indexação *IVFFlat*;

Na etapa de pré-processamento duas abordagens foram utilizadas, a normalização e a remoção das *stopwords*. A aplicação da primeira se dá por converter todos os caracteres em letras minúsculas. Já para a segunda, toda palavra presente no texto e no conjunto de *stopwords* foram removidas dos documentos. O conjunto de *stopwords* empregado está disponível na biblioteca NLTK.

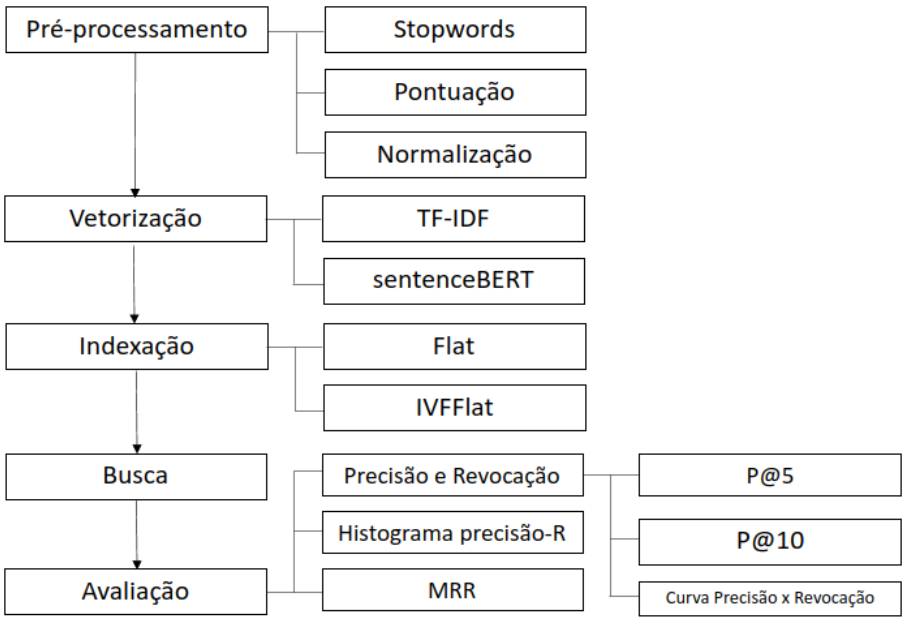

Figura 1. Abordagens Adotadas nas Estratégias Aplicadas

Duas abordagens foram adotadas para a vetorização dos documentos: TF-IDF e *sentence*-BERT. O TF-IDF foi aplicado utilizando a biblioteca *Scikit-Learn*. Para a

geração dos *embeddings* via BERT, utilizou-se o modelo pré-treinado *multi-qa-mpnet-base-dot-v1*[1]. Este modelo foi ajustado para busca semântica utilizando 215 milhões de pares de pergunta e resposta de diversas fontes, este suporta sentenças de até 512 palavras, gerando um *embedding* de 768 dimensões.

Dois tipos de indexação foi aplicados, o *Flat* e o *IVFFlat*, ambas disponíveis na biblioteca FAISS.

A busca utilizando indexação *Flat* é realizada calculando a similaridade entre todos os documentos e a consulta. Já a indexação do tipo *IVFFlat* agrupa os documentos com base na distância euclidiana, 10 agrupamentos foram gerados neste trabalho. Na busca utilizando *IVFFlat* a consulta é comparada aos centroides de cada um destes agrupamentos, e o cálculo de similaridade é realizado apenas com os documentos presentes no agrupamento em que a consulta foi atribuído, reduzindo assim o espaço de busca. O resultado das buscas utilizando ambas abordagens ordena os documentos relevantes de acordo com a similaridade encontrada.

A função de ranqueamento para definir a relevância de um documento dado uma consulta será obtida utilizando a similaridade do cosseno, conforme a equação a seguir:

$$sim(\vec{q}, \vec{d_j}) = \frac{\vec{q} \bullet \vec{d_j}}{|\vec{q}| \times |\vec{d_j}|} \tag{4}$$

As estratégias adotadas são avaliadas através da precisão, revocação, precisão em 5 e 10 documentos recuperados (P@5 e P@10), curva de precisão vs. revocação, histograma da precisão-R e *Mean Reciprocal Rank*.

4. Experimentos

Todos as bibliotecas mencionadas na seção 3 estão disponíveis para linguagem Python, e portanto, todos os experimentos foram executados utilizando esta linguagem. A biblioteca FAISS, que disponibiliza as opções de indexação e busca está disponível para ser executada via processamento de CPU ou GPU, neste trabalho utilizou-se o processamento via CPU. Assim sendo, a configuração da máquina utilizada é: processador (CPU) Ryzen 5 3600, 6-CORE, 12-THREADS, velocidade de 3.6GHz e duas memórias (RAM) DDR4 de 16GB e velocidade de 2666MHz.

Todos os códigos utilizados para executar os experimentos mencionados neste trabalho podem ser acessados no GitHub[2].

5. Resultados

Os vetores produzidos pelo *sentence*BERT possuem 768 dimensões, enquanto que os vetores produzidos pelo TF-IDF possuem 13.171 dimensões para os documentos sem pré-processamento e 13.141 dimensões para os documentos com remoção das *stopwords*.

Dentre os 1.239 documentos, 79 foram truncados para os documentos que não receberam pré-processamento (6.4%) e 25 para os documentos em que as *stopwords* foram removidos (2%).

[1]Disponível em: https://huggingface.co/sentence-transformers/multi-qa-mpnet-base-dot-v1

[2]Códigos disponíveis em: https://github.com/jcfneto/ir

A Figura 2 apresenta o *boxplot* dos resultados de precisão das estratégias adotadas. Com esse gráfico é possível visualizar o resumo estatístico da precisão para todas as consultas. A precisão média variou entre 20.1% para indexação do tipo *IVFFlat*, vetorização com TF-IDF e remoção de *stopwords* para 27.1% com indexação *Flat*, vetorização com BERT sem pré-processamento.

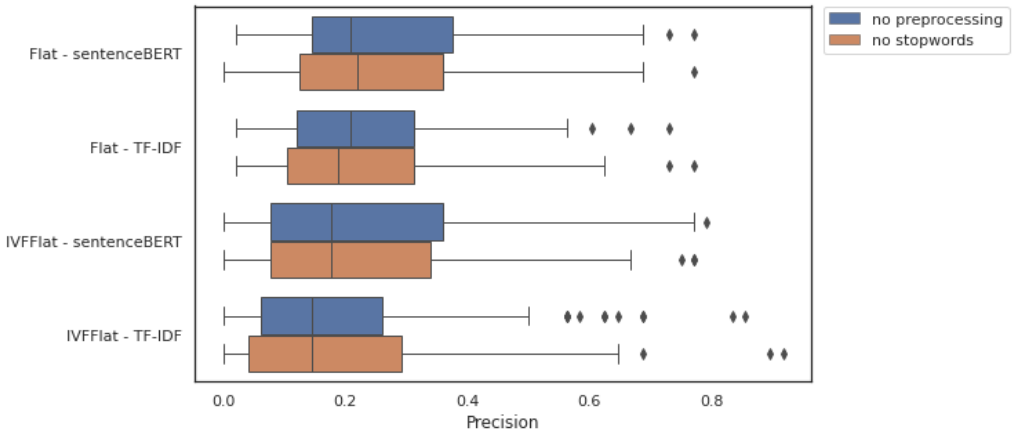

Figura 2. Precisão

Comparativamente, utilizou-se teste-T bilateral para as médias de duas amostras independentes para identificar se há diferença estatística para os experimentos executados com e sem remoção de *stopwords*. A hipótese nula para este teste é de que as duas amostras independentes possuem valores médios idênticos. Para todos os casos, o valor-P observado foi maior do que 5% (intervalo de confiança de 95%), isto significa que, não podemos rejeitar nossa hipótese nula.

A Figura 3 apresenta os resultados de revocação. O teste-T foi implementado com os resultados obtidos, e semelhante ao observado com a precisão, não foi possível perceber diferença estatística (p-Valor $>$ 5%) entre as estratégias utilizando ou não a remoção de *stopwords*.

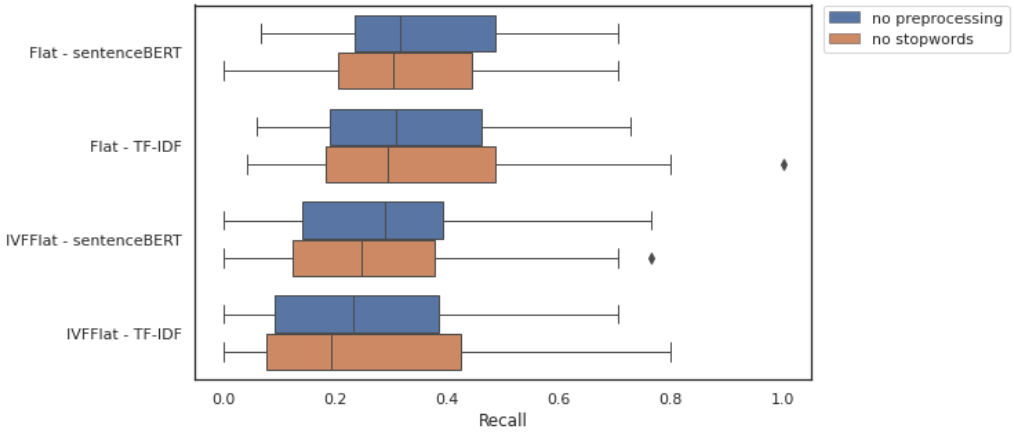

Figura 3. Revocação

As curvas precisão vs revocação são apresentada na Figura 4. Este gráfico é gerado a partir da precisão média obtida para os 11 níveis de revocação (de 0% a 100%). O

gráfico da esquerda apresenta os resultados para as estratégias com a geração dos vetores com os documentos sem pré-processamento, enquanto o gráfico da direita apresenta os resultados para as estratégias que utilizaram os textos com remoção de *stopwords*. É possível observar que, os valores de precisão das estratégias nos níveis finais de revocação ($> 60\%$) invertem nos dois gráficos. Enquanto a estratégia representada pela linha preta finaliza com 100% de revocação e precisão maior que 10% no gráfico a esquerda, no gráfico a direita a mesma estratégia finaliza com precisão inferior a 10%. Esse mesmo padrão pode ser observado nas outras estratégias.

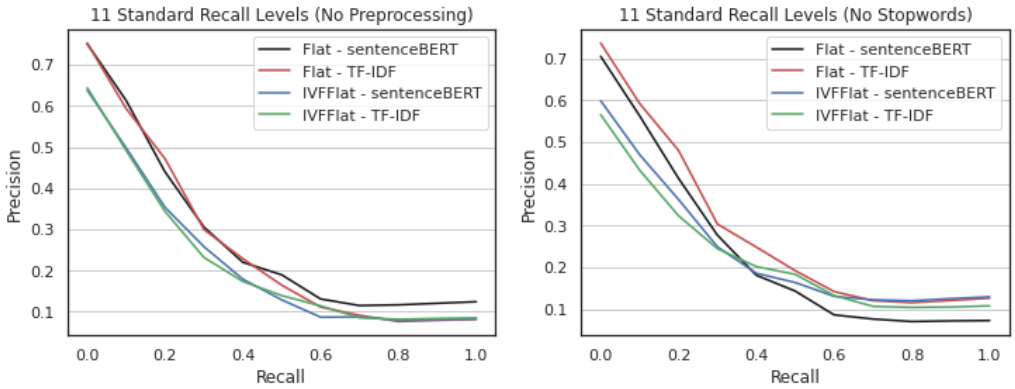

Figura 4. Precisão vs Revocação

As Figuras 5 e 6 apresentam o gráfico de violino para os resultados da precisão em 5 e 10 documentos, respectivamente. Este gráfico exibe um resumo estatístico e a distribuição dos resultados. Na maioria dos casos é possível ver que, não há diferença dos quartis para as distribuições em azul (sem processamento) e cinza (sem *stopwords*), o que é comprovado pelos resultados do teste-T, onde em todos os casos p-Valor $> 5\%$.

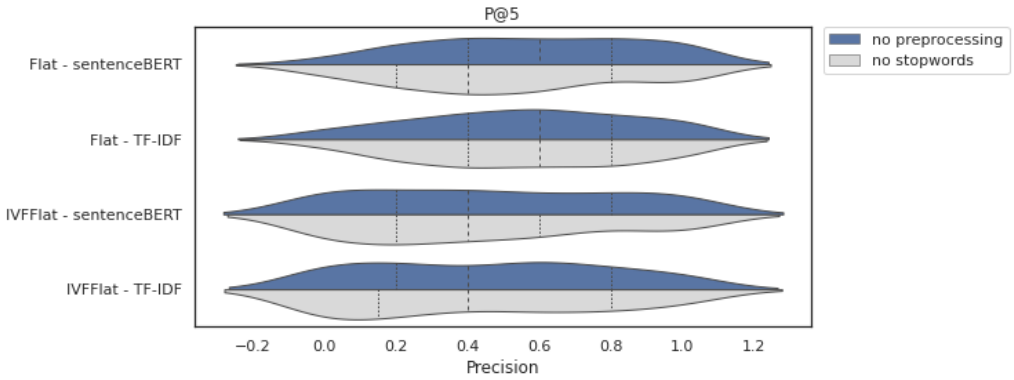

Figura 5. Precisão em 5 Documentos

O MRR é apresentado na Figura 7. É possível notar que, com a utilização dos documentos com remoção de *stopwords* menos documentos relevantes foram recuperados nas primeiras posições e mais consultas ficaram sem documentos relevantes dentro das cinco primeiras posição. Isto foi observado em todas as estratégias, exceto para a estratégia que utiliza indexação *Flat* e vetorização TF-IDF.

Levando em consideração os resultados médios apresentados até o momento, duas estratégias se mostram mais robusta dado as condições do experimento. A primeira é a

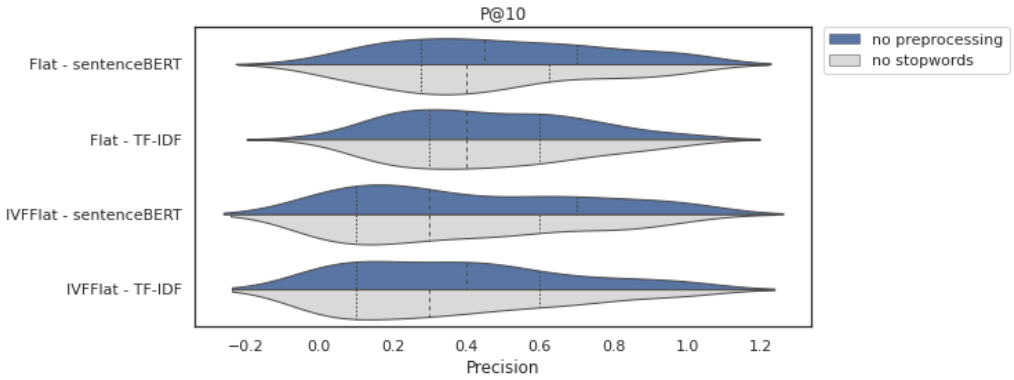

Figura 6. Precisão em 10 Documentos

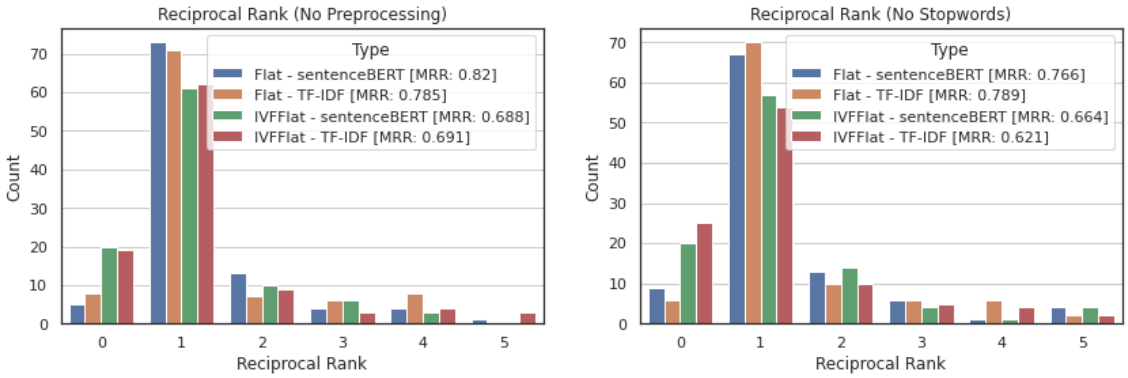

Figura 7. *Mean Reciprocal Rank* (MRR)

indexação *Flat* com vetorização BERT sem pré-processamento e a segunda é o conjunto de indexação *Flat* com vetorização TF-IDF com remoção de stopwords. A Figura 8 apresenta o histograma da precisão-R para as 20 primeiras consultas. Observa-se que, a primeira estratégia possui precisão superior a segunda em 50% das consultas, a segunda em 30% das consultas e em 20% existe um empate.

O último fator avaliado foi o tempo de execução das 1.239 consultas para cada uma das estratégias. A Tabela 1 apresenta os tempos em segundos. Nota-se que, as estratégias utilizando indexação *IVFFlat* foram em média 6.5 vezes mais rápidas em relação a indexação *Flat*. Em relação ao tipo de vetorização, o ganho em velocidade produzido pela indexação *IVFFlat* em relação a *Flat* foi maior na vetorização TF-IDF, com um ganho de 7.1x, enquanto que na vetorização via BERT o ganho foi de 5.8x.

Tabela 1. Tempo Total de Execução das Consultas

Preprocessing	Flat - BERT	Flat - TF-IDF	IVFFlat - BERT	IVFFlat - TF-IDF
no preprocessing	0.072673	1.272436	0.013532	0.198257
no stopwords	0.073614	1.295599	0.012794	0.166142

6. Conclusão

Dentre todas as estratégias utilizadas, a que obteve os melhores resultados no geral foi a estratégia que combinou indexação *Flat*, com representação vetorial via *sentence*BERT.

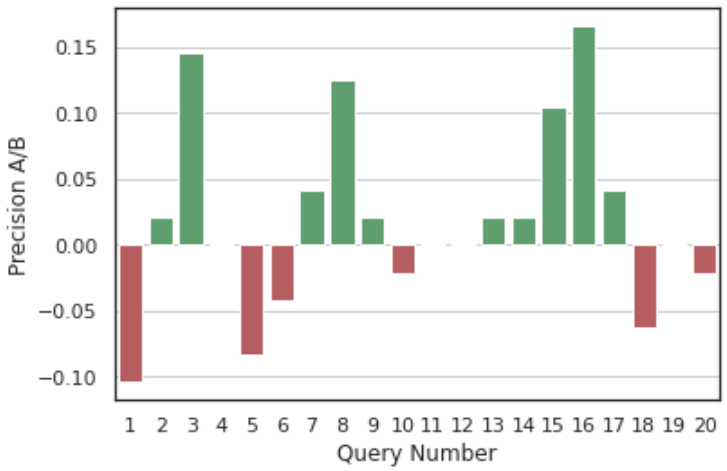

Figura 8. Tempo Total de Execução das Consultas

Porém, o tempo de busca nesse tipo é diretamente proporcional ao tamanho da coleção. Isto é, utilizar esse tipo de busca é interessante quando precisão é mais importante que velocidade e/ou quando o tamanho da coleção é pequeno.

Como foi observado, estatisticamente não houve, para os experimentos aqui realizados, diferença nos resultados aplicando a remoção de *stopwords* e não aplicando qualquer técnica de pré-processamento.

A biblioteca FAISS se mostrou ser uma boa ferramente para indexação e busca de documentos. Sendo sua maior qualidade a facilidade de implementação, requerendo poucas linhas de código para o desenvolvimento desta etapa.

Referências

Baeza-Yates, R., Ribeiro-Neto, B., et al. (1999). *Modern information retrieval*, volume 463. ACM press New York.

Croft, W. B., Metzler, D., and Strohman, T. (2010). *Search engines: Information retrieval in practice*, volume 520. Addison-Wesley Reading.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Singhal, A. et al. (2001). Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43.