Avaliação de *frameworks* para Recuperação de Documentos Legislativos: um Estudo de Caso na Câmara dos Deputados Brasileira

Flávio C. Rocha[1], Ellen Souza[1,4], Douglas Vitório[2], Nádia F. F. da Silva[3], André C. P. L. F. de Carvalho[4], Adriano L. I. Oliveira[2]

[1]Unidade Acadêmica de Serra Talhada (UAST)
Universidade Federal Rural de Pernambuco (UFRPE) – Serra Talhada – PE – Brasil

[2]Centro de Informática (CIn)
Universidade Federal de Pernambuco (UFPE) – Recife – PE – Brasil

[3]Instituto de Informática
Universidade Federal de Goiás (UFG) – Goiânia – GO – Brasil

[4]Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP) – São Carlos – SP – Brasil

{flavio.rocha, ellen.ramos}@ufrpe.br, {damsv,alio}@cin.ufpe.br,

nadia.felix@ufg.br, andre@icmc.usp.br

Abstract. *This work investigates information retrieval frameworks to deal with the difficulties present in the law-making process of the Brazilian Chamber of Deputies. Two open-source frameworks were chosen. In addition, different pre-processing techniques, including stemmers and n-gram language models, were evaluated. Two legislative corpora from the Chamber were used to build and validate the experiments. The results were compared to a baseline used by the Chamber of Deputies. The baseline showed the best result, achieving a Recall for 20 documents of 0.7376.*

Resumo. *Este trabalho investiga frameworks de recuperação de informações para lidar com as dificuldades existentes no processo de elaboração de leis da Câmara dos Deputados do Brasil. Dois frameworks de código aberto foram escolhidos. Além disso, foram avaliadas diferentes técnicas de pré-processamento, incluindo stemmers e modelos de linguagem n-gram. Dois corpora legislativos da Câmara foram usados para construir e validar os experimentos. Os resultados foram comparados com um baseline utilizado pela Câmara dos Deputados. O baseline apresentou o melhor resultado, alcançando um Recall para 20 documentos de 0,7376.*

1. Introdução

Um sistema de recuperação de informação tem o objetivo de atender a necessidade de informação dos usuários, apresentando os documentos relevantes referentes a uma consulta, também chamada *query*, caso eles existam [da Silva and Maia 2018]. A área legal foi uma das primeiras a adotar esse método, com a criação do primeiro sistema de

recuperação de informação para um domínio específico, tendo, inclusive, surgido uma subárea denominada Recuperação de Informação Legal [Maxwell and Schafer 2008].

Na Câmara dos Deputados do Brasil, há uma alta demanda na produção legislativa, sendo ela realizada, em sua maioria, de forma manual, o que gera um alto custo. Desde a década de 30, estima-se que mais de 144 mil proposições foram criadas e, durante todo o processo de sua criação, são gerados e anexados diversos documentos, desde o rascunho da proposta legislativa até a proposta final para a votação [Brandt 2020]. Nesse sentido, técnicas de PLN podem ser utilizadas para minimizar o trabalho manual relacionado à elaboração e evolução de leis.

O objetivo deste artigo é avaliar o desempenho de *frameworks* para a recuperação de documentos legislativos. Existem vários *frameworks* de recuperação de informação que podem ser facilmente implantados e utilizados. Sendo assim, foram escolhidos dois *frameworks* que utilizam a abordagem de *Exact Match*: CHERCHE [Sourty et al. 2022] e Haystack [Pietsch M. 2020], tendo em vista que essa é a abordagem atualmente utilizada na Câmara dos Deputados. Para comparar os resultados, o *pipeline* para a recuperação de documentos legislativos da Câmara dos Deputados do Brasil foi utilizado como *baseline*. Este pipeline avalia três variantes do algoritmo BM25 e 21 combinações de pré-processamento e foi apresentado em [Souza et al. 2021b].

O restante do artigo está organizado da seguinte forma: a Seção 2 apresenta os trabalhos relacionados; a Seção 3 detalha o método; a Seção 4 apresenta os resultados; e, por fim, a Seção 5 apresenta a conclusão e os trabalhos futuros.

2. Trabalhos Relacionados

Sistemas de recuperação de documentos estão presentes no cotidiano dos usuários, pois, por meio deles, é possível realizar buscas de forma eficiente por documentos, páginas da web, imagens, vídeos, entre outros [Yates et al. 2021]. A maioria dos sistemas de busca é geralmente classificada como *Exact Match* ou *Relevance Matching*. Enquanto o *Exact Match* tem foco na correspondência exata dos termos do documento com os termos da consulta, ou *query*, o *Relevance Matching* tem o foco na compreensão semântica, ou seja, refere-se a técnicas que visam abordar uma variedade de fenômenos linguísticos, levando em consideração intenções de contexto, para melhorar o acesso à informação [Li et al. 2014]. Os sistemas de busca geralmente utilizam corpora para processar e apresentar documentos relevantes. Corpora constituem conjuntos de dados linguísticos coletados de fontes variadas: falados, escritos, interpretados, entre outros. Esses dados são armazenados em formato eletrônico e codificados de acordo com padrões previamente estipulados [Bartholamei Junior et al. 2013].

O trabalho desenvolvido por [Souza et al. 2021b] descreve a criação do *pipeline* de recuperação de informação para textos legislativos utilizado pela Câmara dos Deputados do Brasil. Foram realizados experimentos com diversas técnicas de pré-processamento. Dois *stemmers* para a língua portuguesa foram comparados (Savoy e RSLP), além do uso da abordagem *n-gram* a nível de palavra. Três variantes do algoritmo BM25 também foram avaliados (BM25 Okapi, BM25+ e BM25L). A configuração que apresentou o melhor resultado utilizou o algoritmo Savoy como *stemmer*, o modelo *unigram* combinado com *bigram* e o BM25L para recuperação dos documentos. Ela alcançou um R@20 (*Recall* para vinte documentos) de 0,7356.

O trabalho de [Vitório et al. 2022] apresenta uma melhoria para o *pipeline* desenvolvido por [Souza et al. 2021b], utilizando a informação do *feedback* do usuário fornecido para *queries* similares àquela sendo processada no momento. Os resultados dos experimentos mostraram que a implementação desse método melhorou o desempenho dos algoritmos BM25L e BM25 Okapi em 2,18% e 2,10%, respectivamente.

O trabalho de [Chalkidis et al. 2021] relata sobre o uso de um sistema de recuperação de informação regulatória, que se trata de uma aplicação de recuperação de documento a documento, ou seja, dado como entrada um documento, o sistema retorna uma lista de documentos semelhantes àquele. Foram utilizados dois conjuntos de dados escritos na Língua Inglesa, com base nos relacionamentos entre as diretivas da União Europeia e a legislação do Reino Unido. Os autores compararam o BM25 e variações do modelo BERT. As métricas utilizadas para analisar o desempenho dos algoritmos foram *Recall* (R), *Normalized Discounted Cumulative Gain* (NDCG) e *R-Precision* (RP), tendo o modelo BERT treinado para classificação de um domínio especifico alcançado os melhores resultados.

No trabalho de [Wrzalik and Krechel 2021], foi criada uma base de dados com textos escritos em Alemão para recuperação de informação legal, com o objetivo de fornecer apoio a pesquisadores na área de recuperação de informação na Alemanha, visto que, até a publicação do trabalho, não havia um corpus com esse objetivo no país. Para avaliar o corpus, foram utilizados algoritmos baseados no BM25 e em redes neurais. O algoritmo BM25 ajustado com os parâmetros k1 = 0,90 e b = 0,98 obteve o melhor resultado: R@1000 de 0,829.

3. Método

O *pipeline* utilizado aqui foi baseado em [Souza et al. 2021b]. Como entrada, o processo recebe uma *query*, que corresponde a uma consulta legislativa realizada por um parlamentar ou por alguém designado por ele. O algoritmo, então, retorna as proposições legislativas ranqueadas com base na sua relevância para a consulta do parlamentar.

No total, foram avaliadas 21 configurações, combinando técnicas de pré-processamento básico, dois *stemmers* para a língua portuguesa, quatro modelos de linguagem baseados em *word n-grams*, e dois *frameworks*, como apresentado na Figura 1. Para a avaliação, foi utilizada a métrica R@20 (*Recall* para os 20 primeiros documentos).

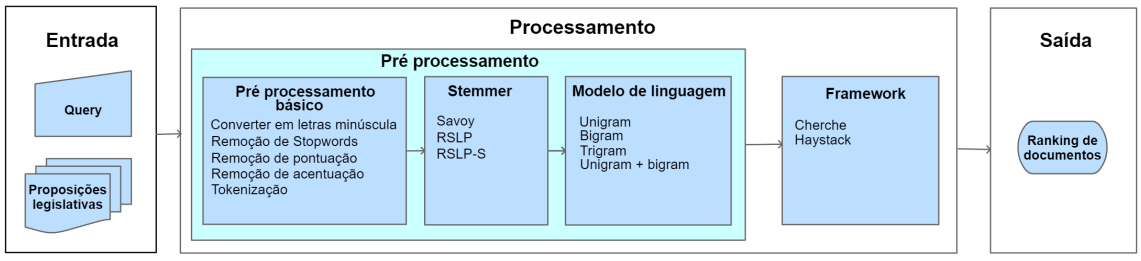

Figura 1. *Pipeline* dos experimentos.

3.1. Corpora

Para os experimentos, foram utilizados dois corpora da Câmara dos Deputados do Brasil: Corpus de Proposições Legislativas (*Bills Corpus*) e Corpus de Solicitações de Trabalho

(*Job Requests Corpus*), ambos apresentados em [Souza et al. 2021b] e detalhados nas subseções a seguir.

3.1.1. Proposições Legislativas

Este corpus é composto por três tipos de proposições legislativas[1]: PL (Projeto de Lei), PLC (Projeto de Lei Complementar) e PEC (Proposta de Emenda Constitucional), contendo um total de 48.555 documentos. Cada entrada do corpus possui informações como o texto da proposição e seu nome, bem como outros atributos que não foram utilizados neste trabalho.

3.1.2. Solicitações de Trabalho

Durante o processo de confecção de uma proposição legislativa, um dos órgãos da Câmara, denominado Consultoria Legislativa (Conle), é responsável por buscar proposições já criadas e fornecê-las aos parlamentares. Os parlamentares solicitam essa informação através de consultas, denominadas "solicitações de trabalho". As solicitações de trabalho são as *queries* no sistema de busca. O corpus utilizado neste estudo contém 295 *queries* construídas por parlamentares, das quais a Tabela 1 apresenta exemplos.

numero-proposicao-sileg	TxtAssunto
PEC XXXX/2019	Solicito a consultoria elaboração de Projeto de lei para garantir a estabilidade da mulher gravida quando esteja em cargo publico comissionado. Como não à possível impedir a exoneração que já preveja uma indenização no valor da media salarial do ultimo ano durante o período em que está gravida e período da licença maternidade.
PL XXXX/2019	art. XXX-X da CLT - insalubridade trabalhadora gestante ou lactante

Tabela 1. Exemplos de solicitação de trabalho.

3.2. Pré-processamento

Variando as técnicas de pré-processamento, foram construídas 21 configurações para os experimentos. Como pré-processamento básico, foram utilizadas a conversão do texto em letras minúsculas e a remoção de acentuação, pontuação e *stopwords*. Além disso, também foram avaliados algoritmos de Stemming e modelos de linguagem *n-gram*.

A técnica de Stemming é utilizada para redução de dimensionalidade, minimizando os problemas para a abordagem *Exact Match*, pois as palavras são reduzidas ao seu radical, melhorando a precisão dos algoritmos de recuperação de informação.

Os algoritmos de Stemming avaliados foram selecionados devido ao desempenho apresentado em [Souza et al. 2021a] e são eles: Savoy e RSLP. O Savoy [Savoy 2006] está disponível em diversas linguagens, inclusive em Português, e faz a remoção de inflexões associadas a adjetivos e substantivos, baseada em regras para o plural e na forma feminina. O RSLP (Removedor de Sufixos da Língua Portuguesa) abrange apenas a Língua Portuguesa e faz um processo de sucessivas remoções de sufixos semelhante ao algoritmo Porter [Porter 1980].

[1]https://www.camara.leg.br/propostas-legislativas/2309349

O modelo de linguagem utilizado foi o *n-gram*, o qual prevê a probabilidade de um determinado *n-gram* dentro de qualquer sequência de palavras. Um *n-gram* é uma sequência de partículas presentes em uma sequência de texto, as quais podem ser palavras, caracteres, entre outros.

A biblioteca Python NLTK foi utilizada para esta etapa de pré-processamento. E a Tabela 2 traz a lista de configurações construídas e avaliadas.

Configurações	
1. sem nenhum pré-processamento	12. apenas com unigram + bigram
2. apenas com a conversão para letras minúsculas	13. todo o pré-processamento básico + bigram
3. letras minúsculas + remoção de pontuação	14. todo o pré-processamento básico + trigram
4. letras minúsculas + remoção de pontuação + remoção de acentuação	15. todo o pré-processamento básico + unigram + bigram
5. todo o pré-processamento básico: letras minúsculas + remoção de pontuação + remoção de acentuação + remoção de stopwords	16. todo o pré-processamento básico + Stemming (RSLP) + bigram
6. apenas com Stemming (RSLP)	17. todo o pré-processamento básico + Stemming (RSLP) + trigram
7. apenas com Stemming (Savoy)	18. todo o pré-processamento básico + Stemming (RSLP) + unigram + bigram
8. todo o pré-processamento básico + Stemming (RSLP)	19. todo o pré-processamento básico + Stemming (Savoy) + bigram
9. todo o pré-processamento básico + Stemming (Savoy)	20. todo o pré-processamento básico + Stemming (Savoy) + trigram
10. apenas com bigram	21. todo o pré-processamento básico + Stemming (Savoy) + unigram + bigram
11. apenas com trigram	

Tabela 2. Combinações de pré processamento.

3.3. *Frameworks*

Para comparação, foram selecionados os *frameworks* CHERCHE e Haystack, visto que estes implementam algoritmos com abordagem de *Exact Match*, foram desenvolvidos em código aberto e são de fácil implementação. A melhor configuração do *pipeline* apresentado por [Souza et al. 2021b], a qual utiliza o algoritmo BM25L, foi utilizada como *baseline* para a comparação de resultados com as outras ferramentas.

O *Elasticsearch*[2] é um mecanismo de busca distribuído e de código aberto, o qual é implementado tanto no CHERCHE quando no Haystack. Ele é usado principalmente para indexar, pesquisar e analisar grandes volumes de dados em tempo real.

O CHERCHE é um *framework* para criar pipelines de busca neural usando algoritmos de recuperação de documentos e modelos de linguagem pré-treinados, como os *transformers*. Além disso, ele pode ser usado para tarefas de PLN, como QA (*Question Answering*), sumarização, tradução e re-ranqueamento de documentos relevantes em um ranking mais preciso. Nos experimentos, esse *framework* foi utilizado com algoritmos de *Exact Match*: BM25 Okapi e BM25L, além de uma implementação de busca usando apenas o *Elasticsearch*, na qual uma consulta simples é feita e a lista de documentos recuperados é organizada na exibição final.

O Haystack é um *framework* de código aberto para a construção de sistemas de pesquisa em grandes coleções de documentos. Com ele, é possível construir uma série de *pipelines* de busca, além de também possuir algoritmos que utilizam modelos neurais para determinadas tarefas de PLN, como QA, sumarização, re-ranqueamento, extrator de identidades nomeadas, entre outros. Para este trabalho, foi utilizado apenas o método de busca que utiliza o algoritmo BM25 Okapi, a única versão do BM25 implementada no Haystack. A implementação desse método utiliza o algoritmo BM25 na etapa de busca, juntamente com o *Elasticsearch* na etapa de recuperação dos documentos; ao contrário da implementação do CHERCHE, que utiliza o *Elasticsearch* tanto para realizar a consulta quanto para recuperar os documentos.

[2]https://www.elastic.co/pt/

Sendo assim, quatro modelos com base nos *frameworks* foram avaliados: CHERCHE com BM25 Okapi, CHERCHE com BM25L, CHERCHE com *Elasticsearch* e Haystack com BM25 Okapi.

3.4. Avaliação

Como forma de avaliar o desempenho dos modelos, foi utilizada a métrica R@20 (*Recall* para 20 documentos), a mesma utilizada no baseline [Souza et al. 2021b]. O *Recall* computa o percentual de documentos relevantes que foram recuperados dentre os documentos recuperados, neste caso dentre os 20 documentos. Vale ressaltar que, aumentando a quantidade de documentos recuperados, o Recall também tende a aumentar. Porém, para o usuário de um sistema de recuperação de informação, a apresentação de uma grande quantidade de documentos não é interessante.

Também é importante pontuar que, no corpus utilizado neste trabalho, cada query possuía apenas um documento relevante.

4. Resultados

A Tabela 3 traz os resultados alcançados pelos *frameworks* e pelo *baseline* para cada uma das 21 configurações de pré-processamento. Comparando os resultados do *baseline* do trabalho de [Souza et al. 2021b] com os *frameworks*, é notório que o *baseline* obteve o melhor desempenho, superando os *frameworks* em praticamente todas as configurações.

Configuração	CHERCHE - BM25 Okapi	CHERCHE - BM25L	CHERCHE - Elasticsearch	Haystack - BM25 Okapi	Baseline
pré-processamento básico					
1.	0,5322	0,0199	0,5152	0,5423	**0,6678**
2.	0,4745	0,0033	0,5288	0,5288	**0,6983**
3.	0,4745	0,0033	0,5288	0,5288	**0,7153**
4.	0,5457	0,0033	0,5932	0,5932	**0,7153**
5.	0,5661	0,0101	0,5898	0,5898	**0,7153**
Stemming					
6.	0,5016	0,0033	0,5627	0,5084	**0,6847**
7.	0,4372	0,0033	0,5254	0,5084	**0,6712**
8.	0,5457	0,0101	0,6033	0,5457	**0,7288**
9.	0,5627	0,0135	0,6169	0,5830	**0,7186**
n-gram					
10.	0,4576	0,0644	0,5016	0,5084	**0,5864**
11.	0,3627	0,2474	0,4813	0,4711	**0,4881**
12.	0,5016	0,0067	0,4813	0,4813	**0,6712**
pré-processamento básico + n-gram					
13.	0,4847	0,0813	0,5627	0,5627	**0,5898**
14.	0,3762	0,2474	**0,5389**	0,5322	0,4712
15.	0,5491	0,0067	0,5389	0,5389	**0,7051**
pré-processamento básico + n-gram + RSLP					
16.	0,5254	0,0644	0,5491	0,4745	**0,6305**
17.	0,4000	0,2440	**0,5186**	0,4542	0,4847
18.	0,5762	0,0101	0,5355	0,4644	**0,7322**
pré-processamento básico + n-gram + Savoy					
19.	0,5118	0,0745	0,5796	0,5322	**0,6237**
20.	0,3898	0,3864	**0,5694**	0,5152	0,4780
21.	0,5728	0,5559	0,5864	0,5423	**0,7356**

Tabela 3. Resultados dos experimentos dos *frameworks*.

Comparando apenas os *frameworks*, pode-se apontar que o CHERCHE utilizando *Elasticsearch* conseguiu superar os demais, sendo melhor, inclusive, que o *baseline* nas três configurações nas quais o *trigram* foi utilizado juntamente com outras técnicas de pré-processamento (configurações 14, 17 e 20). O Haystack apresentou um desempenho semelhante, superando o *baseline* em dois casos (14 e 20). Enquanto isso, o CHERCHE

utilizando o BM25L apresentou resultados bem inferiores aos demais. Esse fraco desempenho pode ser explicado por um problema na implementação do algoritmo BM25L nesse *framework*, o que prejudicou sua performance. Embora a mesma implementação tenha sido utilizada no *pipeline* presente em [Souza et al. 2021b], esse problema foi corrigido no algoritmo em funcionamento na Câmara dos Deputados.

Ao analisar as diversas configurações de pré-processamento, percebe-se que os melhores resultados alcançados pelos *frameworks* foram obtidos através da combinação apenas do pré-processamento básico com o Stemming, sem o uso de *n-gram* (configurações 8 e 9). Esse resultado difere dos obtidos pelo *baseline* no artigo de [Souza et al. 2021b], nos quais o uso de *n-gram* melhorou a performance do algoritmo de recuperação de informação. Isso mostra que *frameworks* e técnicas diferentes apresentam resultados também diferentes com o mesmo pré-processamento. Sendo assim, é essencial avaliar quais as técnicas de pré-processamento se adéquam melhor ao algoritmo a ser utilizado.

5. Conclusão e Trabalhos Futuros

Neste trabalho, investigou-se dois *frameworks* (CHERCHE e Haystack) para a tarefa de recuperação de documentos legislativos no cenário da Câmara dos Deputados do Brasil, comparando-os com o trabalho de [Souza et al. 2021b], o qual apresenta o *pipeline* utilizado atualmente na Câmara. Para os experimentos, foram construídas 21 configurações combinando diferentes técnicas de pré-processamento, dois algoritmos de Stemming (RSLP e Savoy) e tamanhos diferentes de *n-gram* (*unigram*, *bigram*, *trigram* e *unigram* + *bigram*).

Com base nos resultados, percebeu-se que nenhum dos *frameworks* superou o desempenho do *pipeline* apresentado por [Souza et al. 2021b].

Afora o *pipeline* supracitado, pode-se indicar o *framework* CHERCHE com o *Elasticsearch* como uma alternativa viável, caso necessite-se de um modelo já implementado e de rápida utilização. Porém, também pôde-se notar que a melhor configuração de técnicas de pré-processamento varia conforme o algoritmo utilizado. Portanto, o pré-processamento deve ser sempre avaliado antes da execução da recuperação de documentos legislativos.

Como trabalhos futuros, serão realizados experimentos com o texto das proposições totalmente segmentado, de forma similar ao trabalho de [Rosa et al. 2021], no qual essa segmentação melhorou o desempenho do algoritmo de recuperação de documentos. Também serão avaliados outros corpora legislativos, como o apresentado no trabalho de [Vitório et al. 2022], e outros no domínio legal da Língua Portuguesa. A utilização de outras métricas para avaliar o desempenho dos *frameworks*, bem como uma análise do tempo de processamento, também serão realizadas. Além disso, outros *frameworks* podem ser avaliados, juntamente com os algoritmos de busca que utilizam redes neurais presentes nos *frameworks*.

Referências

Bartholamei Junior, L. A. et al. (2013). Proposta de ordem sequencial e criação de sistemas informáticos para extração terminológica bilíngue em corpora paralelos-inglês/português-com vistas à tradução de texto das ciências médicas.

Brandt, M. B. (2020). Modelagem da informação legislativa: arquitetura da informação para o processo legislativo brasileiro.

Chalkidis, I., Fergadiotis, M., Manginas, N., Katakalou, E., and Malakasiotis, P. (2021). Regulatory compliance through doc2doc information retrieval: A case study in eu/uk legislation where text similarity has limitations. *arXiv preprint arXiv:2101.10726*.

da Silva, F. T. and Maia, J. E. B. (2018). Luppar: An information retrieval system for closed document collections. In *Anais do XV Encontro Nacional de Inteligência Artificial e Computacional*, pages 912–923. SBC.

Li, H., Xu, J., et al. (2014). Semantic matching in search. *Foundations and Trends® in Information Retrieval*, 7(5):343–469.

Maxwell, K. T. and Schafer, B. (2008). Concept and context in legal information retrieval. In *Legal Knowledge and Information Systems*, pages 63–72. IOS Press.

Pietsch M., Soni T., C. B. M. T. K. B. (2020). Haystack (version 0.5.0). In *GitHub*.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, 14(3):130–137.

Rosa, G. M., Rodrigues, R. C., Lotufo, R., and Nogueira, R. (2021). Yes, bm25 is a strong baseline for legal case retrieval. *arXiv preprint arXiv:2105.05686*.

Savoy, J. (2006). Light stemming approaches for the french, portuguese, german and hungarian languages. In *Proceedings of the 2006 ACM symposium on Applied computing*, pages 1031–1035.

Sourty, R., Moreno, J. G., Tamine, L., and Servant, F.-P. (2022). Cherche: A new tool to rapidly implement pipelines in information retrieval. In *Proceedings of SIGIR 2022*.

Souza, E., Moriyama, G., Vitório, D., de Carvalho, A. C., Félix, N., Albuquerque, H. O., and Oliveira, A. L. (2021a). Assessing the impact of stemming algorithms applied to brazilian legislative documents retrieval. In *Anais do XIII Simpósio Brasileiro de Tecnologia da Informação e da Linguagem Humana*, pages 227–236. SBC.

Souza, E., Vitório, D., Moriyama, G., Santos, L., Martins, L., Souza, M., Fonseca, M., Félix, N., Carvalho, A. C., Albuquerque, H. O., et al. (2021b). An information retrieval pipeline for legislative documents from the brazilian chamber of deputies. In *Legal Knowledge and Information Systems*, pages 119–126. IOS Press.

Vitório, D., Souza, E., Martins, L., da Silva, N. F., de Leon Ferreira de Carvalho, A. C. P., and Oliveira, A. L. (2022). Ulysses-RFSQ: A novel method to improve legal information retrieval based on relevance feedback. In *Intelligent Systems: 11th Brazilian Conference, BRACIS 2022, Campinas, Brazil, November 28–December 1, 2022, Proceedings, Part I*, pages 77–91. Springer.

Wrzalik, M. and Krechel, D. (2021). Gerdalir: A german dataset for legal information retrieval. In *Proceedings of the Natural Legal Language Processing Workshop 2021*, pages 123–128.

Yates, A., Nogueira, R., and Lin, J. (2021). Pretrained transformers for text ranking: Bert and beyond. In *Proceedings of the 14th ACM International Conference on web search and data mining*, pages 1154–1156.