Análise da Eficácia de Fine-Tuning de Embeddings no Contexto Jurídico Brasileiro

Matheus Vargas
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brasil
mvargas@cos.ufrj.br

***Abstract*—Um sistema de recuperação da informação preciso é de grande relevância no contexto jurídico, tanto para seres humanos interessados em jurisprudência, quanto para modelos de linguagem no contexto de *Retrieval Augmented Generation*. Entretanto, modelos de embedding genéricos podem gerar representações não muito úteis para campos especializados, como é o caso do Direito. Este artigo apresenta uma metodologia de fine-tuning para especializar um modelo de representação de texto para o acervo da Procuradoria Geral do Município do Rio de Janeiro (PGM-Rio). Construímos um conjunto de dados de treinamento com base nos metadados fornecidos, e utilizamos as funções de perda *Multiple Negatives Ranking Loss* (MNRL) e *Matryoshka Representation Learning* (MRL) para ajustar o modelo `gte-multilingual-base`. A avaliação, definida por duas tarefas de similaridade semântica de diferentes dificuldades, demonstra que o nosso modelo (`gte-finetune-pgm`) alcança um ganho de performance relativo de até 28,4% na métrica NDCG@20. Adicionalmente, demonstramos que as representações de baixa dimensão geradas via MRL superam as baselines de alta dimensão, oferecendo um trade-off eficiente entre qualidade e custo computacional. Concluímos que o fine-tuning específico ao domínio é uma estratégia eficaz para aprimorar a recuperação de informação jurídica.**

I. Introdução

A crescente digitalização de processos no setor jurídico gerou um volume massivo de documentos, tornando a busca e a recuperação de informações uma tarefa desafiadora, principalmente em bases de dados de grandes proporções, como é o caso do acervo da Procuradoria Geral do Município do Rio de Janeiro (PGM-Rio). A busca por similaridade através de modelos de embeddings é uma solução escalável e poderosa, por criar representações textuais que capturam a semântica do texto. Porém, modelos de embedding de propósito geral podem criar representações pouco úteis em domínios específicos [1].

Este trabalho aborda diretamente esse problema. Propomos e avaliamos uma metodologia para especializar um modelo de embedding para o domínio jurídico da PGM-Rio. Para isso, realizamos o fine-tuning do modelo `Alibaba-NLP/gte-multilingual-base` utilizando uma estratégia de treinamento contrastivo baseada nos metadados dos próprios documentos. A metodologia emprega a função de perda *Multiple Negatives Ranking Loss* (MNRL) para otimizar a distinção entre documentos, uma estratégia de seleção de negativos difíceis (*hard negatives*) para aprimorar a capacidade de discernimento do modelo, e a técnica de *Matryoshka Representation Learning* (MRL) para permitir a geração de embeddings de dimensionalidade variável, oferecendo flexibilidade entre custo computacional e qualidade.

A avaliação, que envolveu a seleção de documentos aleatórios fixos e o uso de diferentes métricas consagradas da área de Recuperação da Informação, demonstra que o nosso modelo ajustado é superior aos modelos genéricos na tarefa de similaridade semântica, até mesmo em dimensões reduzidas. Em particular, na métrica de *Normalized Discounted Cumulative Gain* (NDCG) o `gte-finetune-pgm` alcançou ganhos de até 28,4%.

O restante deste artigo está organizado da seguinte forma: a Seção II detalha a base de dados e seu tratamento. A Seção III apresenta a metodologia de fine-tuning e avaliação. A Seção IV expõe e analisa os resultados obtidos. Finalmente, a Seção V conclui com uma discussão sobre as limitações do estudo.

II. Base de dados

A. Descrição da base de dados original

O conjunto de dados utilizado neste trabalho foi cedido pela Procuradoria Geral do Município do Rio de Janeiro (PGM-Rio). Ele consiste em um banco de dados relacional que armazena os textos e metadados relevantes aos documentos legais de diferentes processos.

Os metadados das tabelas foram fundamentais para criar uma relação de similaridade de referência entre os textos. Em específico, as colunas de metadados utilizadas para esse propósito foram:

- `assunto_principal`: Assunto principal do processo. ”Verba estipendial”, ”Internação”, ”Responsabilidade Civil” e ”IPTU” são alguns exemplos de valores referentes à essa coluna.
- `especializada`: Procuradoria especializada responsável pelo processo. Exemplos: ”PUMA - Procuradoria de Urbanismo e Meio Ambiente” e ”PAS - Procuradoria de Atenção a Saúde”.
- `tipo`: Tipo do documento. Representado por um código númerico que identifica o propósito do documento, como intimação, acórdão e decisão.
- `tipo_processo`: Tipo do processo. ”Cumprimento de sentença”, ”Reclamação trabalhista” e ”Mandado de segurança” são alguns dos valores possíveis.

O banco de dados demanda cerca de 8 TB de armazenamento, sendo a maioria ocupada pelos textos completos dos processos, nos formatos HTML ou PDF.

B. Tratamento dos dados

Os documentos originais, em formato PDF e HTML, não são diretamente compatíveis com modelos de linguagem baseados em texto. Portanto, desenvolvemos uma pipeline de pré-processamento para extrair o conteúdo textual.

Muitos dos arquivos são parcialmente ou completamente escaneados, necessitando de modelos de Reconhecimento de Caracter Óptico (OCR, da sigla em inglês) para a sua extração total. O resultado da extração de tais documentos foi misto, e depende tanto do modelo de OCR utilizado quanto da qualidade da imagem; empiricamente, avaliamos que o modelo Tesseract [2] produz transcrições melhores do que as do PaddleOCR [3] para imagens que julgamos mais fáceis, isto é, imagens com poucos artefatos e em alta resolução, enquanto que o contrário ocorria para as imagens "difíceis".

Para garantir a alta fidelidade do texto de entrada e isolar a análise do impacto do fine-tuning, optamos por focar exclusivamente em documentos totalmente digitais, descartando aqueles que exigiriam OCR.

Documentos duplicados, muito pequenos ou com metadados muito raros também foram filtrados.

Após a aplicação de todos os filtros, chegamos ao número final de 261.325 documentos, referentes a 26.286 processos diferentes. Para gerenciar os documentos processados e viabilizar as operações de busca por similaridade vetorial de forma eficiente, migramos os dados para um banco de dados PostgreSQL, utilizando a extensão `pgvector`, que permite o armazenamento nativo e a indexação dos embeddings.

III. Metodologia

A metodologia de tratamento e seleção de dados está ilustrada no fluxograma da Figura 1. As seções a seguir detalham cada etapa deste processo, além do treinamento do modelo.

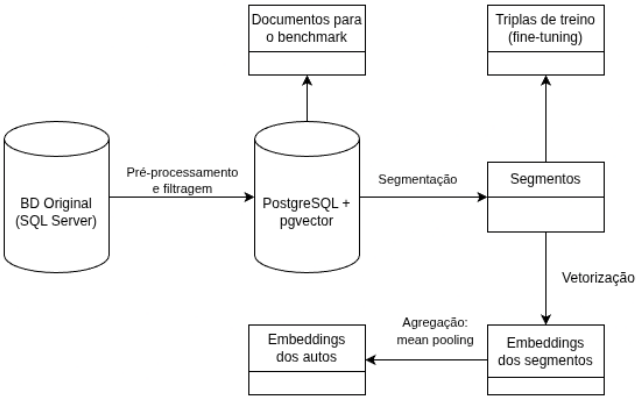

Fig. 1. Fluxograma da pipeline de processamento de dados. O processo inicia com a extração e filtragem dos dados, seguido segmentação, vetorização dos segmentos e criação das triplas de treino, e finalmente a agregação para obter os embeddings dos autos via *mean pooling*.

A. Segmentação

Os modelos de arquitetura Transformers utilizados possuem limites de tokens processados por vez, e alguns dos documentos que possuímos são enormes.

Em particular, há um documento da base de dados filtrada que é composto por 5.239.935 caracteres. Até os modelos da família Gemini 2.5 da Google, reconhecidos por terem uma larga janela de contexto de 1 milhão de tokens [4] não seriam capazes de ingerir essa quantidade de texto se assumirmos que um token equivale a quatro caracteres, que é a aproximação fornecida pela Google para seus modelos [5].

Tendo essa limitação em vista, dividimos os documentos em segmentos de 1024 tokens, utilizando o tokenizador do GPT-2, com o segmentador recursivo do Langchain, que respeita limites semânticos como parágrafos e pontuação ao custo de não segmentar os textos no valor especificado exatamente.

Esse processo resultou na criação de 2.145.959 segmentos.

B. Modelos

Os modelos foram escolhidos com base no ranqueamento do MMTEB [6], que é um benchmark que avalia a qualidade dos embeddings produzidos pelos diferentes modelos, com base em um conjunto diverso de tarefas.

Em particular, os dois modelos escolhidos se destacam pelos requisitos computacionais modestos, boa qualidade e janela de contexto relativamente grande:

- `jinaai/jina-embeddings-v3`: Baseado na arquitetura do XLM-RoBERTa, possui 572 milhões de parâmetros, janela de contexto de 8192 tokens, dimensão de representação adaptável de 1024 tokens, além de adaptadores LoRA para diferentes tarefas [7].
- `Alibaba-NLP/gte-multilingual-base`: Adaptação do BERT, possui 305 milhões de parâmetros, janela de contexto de 8192 tokens, dimensão de representação adaptável de 768 tokens [8].

Para o objetivo principal desse artigo, que é o fine-tuning de um modelo, optamos por ajustar apenas o `gte-multilingual-base` puramente pela quantidade menor de parâmetros em relação ao modelo da Jina AI, o que diminui o consumo de memória e acelera o treinamento. Vamos nos referir a esse modelo ajustado como `gte-finetune-pgm` a partir de agora.

Esses 3 modelos foram utilizados para transformar em vetor cada um dos segmentos gerados.

C. Estratégia de Fine-Tuning

Para ajustar o modelo de embedding ao domínio jurídico da PGM-Rio, adotamos uma estratégia de fine-tuning utilizando a biblioteca `sentence-transformers` [9]. O treinamento foi feito com a função de perda *Multiple Negatives Ranking Loss* (MNRL) [10], uma abordagem eficiente que utiliza os exemplos dentro de um mesmo lote como negativos para cada âncora. Mais especificamente, essa perda considera como um exemplo positivo apenas o elemento rotulado como tal, e todos os outros exemplos, incluindo as outras âncoras e seus respectivos pares, como negativos. Adicionalmente, foi utilizada a

técnica de *Matryoshka Representation Learning* (MRL) [11], que permite que o modelo treinado produza embeddings de dimensões variadas sem a necessidade de retreinamento, o que habilita ao modelo à produção de embeddings de qualidade não muito pior com dimensões reduzidas, o que flexibiliza a relação entre qualidade de embedding e custo computacional.

O sucesso do fine-tuning contrastivo, como o que utiliza a perda MNRL, depende da qualidade dos exemplos de treinamento. Por isso, desenvolvemos uma metodologia baseada em heurísticas para construir um conjunto de dados de 10.000 triplas no formato (*âncora, positivo, negativo*), onde cada elemento é um documento da nossa base de dados.

1) Construção do Conjunto de Dados de Treinamento: O processo de construção das triplas de treinamento foi guiado pelos metadados dos documentos, com o objetivo de ensinar ao modelo as nuances de similaridade semântica relevantes para o nosso domínio. Novamente, aqui há o problema da janela de contexto limitada com relação ao tamanho do documento. Por conta disso, uma aproximação dos documentos foi criada na forma dos seus dois primeiros e dois últimos segmentos, limitando o total de tokens por documento em 4096 ($1024 \times 4 = 4096$).

a) Seleção das Âncoras: Os documentos âncora foram selecionados através de uma amostragem estratificada. Primeiramente, identificamos grupos de documentos que compartilham simultaneamente quatro metadados: `especializada`, `tipo`, `assunto_principal` e `tipo_processo`. Apenas grupos com mais de 100 documentos foram considerados, para garantir representatividade. Em seguida, amostramos um número balanceado de documentos de cada um desses grupos para compor o conjunto final de 10.000 âncoras.

b) Definição de Pares Positivos: Para cada âncora, um documento positivo foi selecionado com base em uma noção de similaridade semântica. A nossa heurística principal define um par positivo como dois documentos que, embora distintos, pertencem ao mesmo grupo de quatro metadados (i.e., compartilham os mesmos valores para `especializada`, `tipo`, `assunto_principal` e `tipo_processo`). Caso nenhum positivo fosse encontrado sob esta regra mais estrita, o requisito era diminuído para buscar um documento que compartilhasse apenas `especializada` e `tipo` com a âncora.

c) Seleção de Negativos Difíceis (Hard Negatives): A seleção de negativos é crucial para um fine-tuning eficaz. Em vez de selecionar negativos aleatoriamente, o que resultaria em exemplos "fáceis" de distinguir, implementamos uma estratégia de seleção de negativos difíceis (*hard negatives*). O objetivo é forçar o modelo a aprender a diferenciar documentos que são superficialmente similares, mas semanticamente distintos. Para introduzir variedade e complexidade, escolhemos três regras para a seleção do negativo, garantindo que ele fosse similar à âncora em alguns aspectos, mas diferente em outros:

- **Regra 1:** O negativo compartilha a mesma `especializada`, mas possui um `tipo` de documento diferente.
- **Regra 2:** O negativo compartilha a mesma `especializada` e `tipo_processo`, mas difere no `tipo`.
- **Regra 3:** O negativo compartilha a mesma `especializada` e `assunto_principal`, mas difere no `tipo`.

Essa abordagem cria um desafio de aprendizado mais complexo, incentivando o modelo a focar nas distinções semânticas mais sutis.

2) Configuração e Parâmetros do Treinamento: A função de perda, como mencionado, foi a `MatryoshkaLoss` encapsulando a `MultipleNegativesRankingLoss`. Utilizamos a perda do Matryoshka para que o modelo final seja capaz de gerar embeddings com as seguintes dimensões: `[768, 512, 256, 128, 64]`.

O treinamento de um modelo de linguagem da escala do `gte-multilingual-base` é custoso. Tendo isso em vista, escolhemos alguns parâmetros para otimizar o uso de memória, como acumulação de gradiente, treinamento com precisão mista (BF16) e *gradient checkpointing*.

Os principais parâmetros foram:

- **Otimizador:** Utilizamos o otimizador `AdamW` com a implementação *fused* do PyTorch.
- **Taxa de Aprendizagem:** 2×10^{-5} adaptável, com um agendador do tipo cosseno para ajustar a taxa.
- **Lote de Treinamento (Batch):** O tamanho do lote foi 16. Para simular um lote maior, utilizamos 16 passos de acumulação de gradiente, resultando em um lote efetivo de 256 amostras.

Foram utilizadas quatro épocas de treinamento.

D. Geração e Agregação de Embeddings

Observe que os segmentos foram gerados apenas como um artifício para evitar a limitação imposta pela janela de contexto; a tarefa de similaridade semântica é sobre os documentos, e não sobre os segmentos.

O passo seguinte é, então, gerar uma representação vetorial única para cada documento a partir dos segmentos. Para alcançar esse objetivo, adotamos um processo de duas etapas:

1) **Vetorização dos Segmentos:** Primeiramente, cada um dos 2.145.959 segmentos gerados foi processado individualmente pelos modelos de embedding para produzir uma representação vetorial.
2) **Agregação por Documento via *Mean Pooling*:** Em seguida, aplicamos a estratégia de *mean pooling* para consolidar os vetores dos documentos. Esta técnica consiste em calcular a média aritmética de todos os embeddings dos segmentos que pertencem a um mesmo documento. O resultado é um único vetor que representa a semântica agregada do documento inteiro.

A abordagem de *mean pooling* foi escolhida por ser uma estratégia computacionalmente eficiente e amplamente utilizada na literatura como um método robusto e eficaz para obter representações de documentos a partir de suas partes.

Ao final deste processo, cada um dos 261.325 documentos do nosso acervo passou a ser representado por um

único vetor, para cada um dos modelos de embedding, o `jina-embeddings-v3`, o `gte-multilingual-base` e o `gte-finetune-pgm`.

E. Avaliação

Para avaliar a qualidade dos embeddings na tarefa de similaridade semântica, foi criado um benchmark com esse propósito.

Selecionamos 1000 documentos aleatórios para compor a base de dados do benchmark. O processo de avaliação consiste em, para cada um desses documentos, criar uma ordenação da base de dados completa e, a partir dessa ordenação, computar métricas relevantes e informativas para a tarefa de similaridade semântica. As métricas escolhidas foram:

- Precision@k (P@k): Fração de documentos relevantes encontrados nos top-k resultados. Mede a precisão imediata da busca.
- Mean Reciprocal Rank @ k (MRR@k): A média do inverso da posição do primeiro documento relevante encontrado (limitado a k). Avalia quão rapidamente o usuário encontra o primeiro resultado útil.
- Normalized Discounted Cumulative Gain @ k (NDCG@k): Mede a qualidade do ranqueamento, atribuindo pesos maiores para documentos relevantes nas primeiras posições.

IV. Resultados e Análise

Para avaliar o desempenho dos modelos na tarefa de similaridade semântica de documentos, conduzimos os experimentos no benchmark de 1000 documentos descrito anteriormente. A avaliação foi dividida em dois cenários distintos para medir a capacidade dos modelos em capturar diferentes níveis de compreensão:

- **Similaridade Fraca:** Um documento é considerado relevante a outro se ambos compartilharem os mesmos metadados de `especializada` e `tipo`. Este é um critério mais abrangente, testando a capacidade do modelo de agrupar documentos por temas e propósitos gerais.
- **Similaridade Forte:** A relevância é definida por uma correspondência exata dos quatro metadados: `especializada`, `tipo`, `assunto_principal` e `tipo_processo`. Este é um critério mais estrito, avaliando a habilidade do modelo em identificar documentos quase idênticos em seu contexto jurídico.

Os modelos comparados foram o `gte-multilingual-base` e o `jina-embeddings-v3`, que servem como baselines, e o nosso modelo `gte-finetune-pgm` em diversas dimensões, graças à técnica Matryoshka. As métricas utilizadas foram Precision@20, MRR@20 e NDCG@20.

A. Resultados em Similaridade Fraca

A Tabela I apresenta os resultados para o cenário de similaridade fraca.

A análise da Tabela I revela uma superioridade expressiva do modelo `gte-finetune-pgm` em todas as métricas e dimensões quando comparado às baselines. O nosso modelo na sua dimensão máxima (768d) alcança um NDCG@20 de 0.6575, representando um ganho relativo de aproximadamente **23.2%** sobre o modelo `gte-multilingual-base` original. Notavelmente, a versão de apenas 64 dimensões do nosso modelo já supera as baselines com folga, atingindo um NDCG@20 de 0.6412. Este resultado demonstra a eficácia do fine-tuning em especializar o modelo para a semântica do domínio, permitindo que suas representações muito mais compactas superem representações vetoriais com alta dimensionalidade de modelos não ajustados.

TABLE I
RESULTADOS DE AVALIAÇÃO NO CENÁRIO DE SIMILARIDADE FRACA. DOCUMENTOS RELEVANTES COMPARTILHAM `ESPECIALIZADA` E `TIPO`.

Modelo	Dimensões	Precision@20	MRR@20	NDCG@20
gte-multilingual-base	768	0.5145	0.7004	0.5338
jina-embeddings-v3	1024	0.4992	0.6795	0.5169
gte-finetune-pgm	64	0.6257	0.7743	0.6412
gte-finetune-pgm	128	0.6340	0.7871	0.6510
gte-finetune-pgm	256	0.6404	0.7854	0.6552
gte-finetune-pgm	512	**0.6422**	0.7877	0.6570
gte-finetune-pgm	768	0.6413	**0.7909**	**0.6575**

B. Resultados em Similaridade Forte

A Tabela II mostra o desempenho dos modelos no cenário de similaridade forte, uma tarefa consideravelmente mais desafiadora.

TABLE II
RESULTADOS DE AVALIAÇÃO NO CENÁRIO DE SIMILARIDADE FORTE. DOCUMENTOS RELEVANTES COMPARTILHAM OS QUATRO METADADOS PRINCIPAIS.

Modelo	Dimensões	Precision@20	MRR@20	NDCG@20
gte-multilingual-base	768	0.2460	0.3973	0.2570
jina-embeddings-v3	1024	0.2414	0.3937	0.2536
gte-finetune-pgm	64	0.3023	0.4754	0.3156
gte-finetune-pgm	128	0.3090	0.4767	0.3217
gte-finetune-pgm	256	0.3165	0.4765	0.3276
gte-finetune-pgm	512	0.3167	0.4788	0.3287
gte-finetune-pgm	768	**0.3173**	**0.4837**	**0.3301**

Conforme esperado, as pontuações absolutas são menores neste cenário, refletindo a sua maior dificuldade. No entanto, a vantagem do modelo `gte-finetune-pgm` se torna ainda mais acentuada. O ganho relativo de desempenho do nosso modelo de 768 dimensões sobre a baseline `gte-multilingual-base` atinge **28.4%** na métrica NDCG@20. Este resultado sugere que o processo de fine-tuning foi bem-sucedido em ensinar ao modelo as distinções finas necessárias para diferenciar documentos com alto grau de sobreposição semântica. Mais uma vez, a versão de 64 dimensões do nosso modelo demonstra um desempenho robusto, superando as baselines por uma margem significativa.

C. Análise Consolidada e o Efeito Matryoshka

Os resultados consolidados demonstram que o fine-tuning foi bem-sucedido. O modelo `gte-finetune-pgm` superou

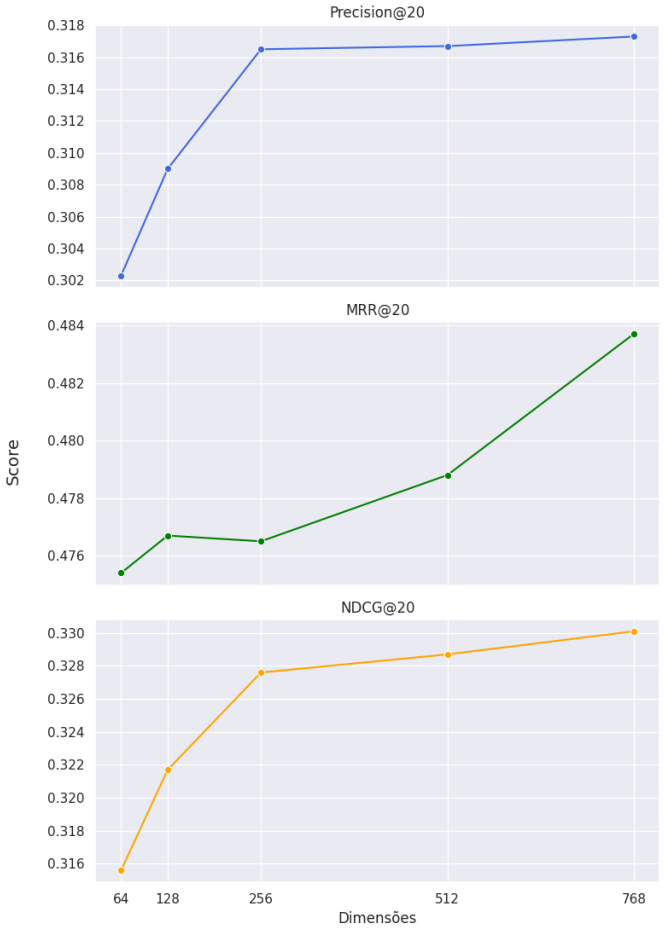

Fig. 2. Gráfico de score da métrica por dimensões do embedding na tarefa de similaridade forte. Em geral, quanto mais dimensões são usadas, melhor o score, mas há uma saturação do ganho conforme o número de dimensões aumenta.

os modelos pré-treinados com ganhos relativos que variaram de 23% a mais de 28%, dependendo da complexidade da tarefa.

A eficácia da técnica *Matryoshka Representation Learning* é uma das conclusões práticas mais importantes deste trabalho. Observa-se que, para ambos os cenários, há um ganho de desempenho incremental, porém decrescente, à medida que aumentamos a dimensionalidade dos embeddings de 64 para 768. Por exemplo, a versão de 256 dimensões já captura a maior parte do ganho de qualidade da versão completa de 768 dimensões. Isso oferece um trade-off valioso: sistemas em produção podem optar por embeddings menores (e.g., 256d), que são mais rápidos para calcular, armazenar e comparar, sofrendo uma penalidade de performance mínima em relação à representação de maior dimensão, e ainda assim obtendo uma qualidade muito superior aos modelos genéricos. A Figura 2 traz uma visualização deste trade-off para as diferentes métricas escolhidas na tarefa de similaridade forte.

V. Discussão e Limitações

Os resultados apresentados demonstram um ganho de performance significativo e consistente do modelo `gte-finetune-pgm`. No entanto, uma análise crítica da nossa metodologia revela pontos importantes para discussão e aponta para limitações inerentes que devem ser consideradas.

A principal limitação deste estudo reside na potencial correlação entre o conjunto de dados de treinamento e o benchmark de avaliação. Apesar de não terem interseção, ambos foram construídos utilizando as mesmas heurísticas baseadas em metadados para definir a similaridade entre documentos. O fine-tuning otimizou o modelo para aproximar os embeddings de documentos com metadados idênticos, e a avaliação mediu exatamente essa capacidade.

Isso levanta uma questão crucial: o modelo aprendeu a semântica do texto jurídico ou simplesmente aprendeu a "agradar ao benchmark", tornando-se um classificador eficiente dos metadados a partir do texto? Embora a segunda hipótese seja uma simplificação exagerada, é inegável que o desempenho observado pode não se generalizar perfeitamente para todas as tarefas de similaridade no mundo real, especialmente para aquelas onde a noção de relevância não pode ser capturada puramente pelos quatro metadados que utilizamos.

References

[1] Y. Tang and Y. Yang, "Do we need domain-specific embedding models? an empirical investigation," 2025.

[2] A. Kay, "Tesseract: an open-source optical character recognition engine," *Linux J.*, vol. 2007, p. 2, July 2007.

[3] Y. Du, C. Li, R. Guo, X. Yin, W. Liu, J. Zhou, Y. Bai, Z. Yu, Y. Yang, Q. Dang, and H. Wang, "Pp-ocr: A practical ultra lightweight ocr system," 2020.

[4] Google DeepMind, "Gemini: March 2025 model updates and thinking behind the tech." https://blog.google/technology/google-deepmind/gemini-model-thinking-updates-march-2025/building-on-best-gemini, 2025.

[5] Google DeepMind, "Gemini api - tokenization." https://ai.google.dev/gemini-api/docs/tokens, 2024.

[6] K. Enevoldsen, I. Chung, I. Kerboua, M. Kardos, A. Mathur, D. Stap, J. Gala, W. Siblini, D. Krzemiński, G. I. Winata, S. Sturua, S. Utpala, M. Ciancone, M. Schaeffer, G. Sequeira, D. Misra, S. Dhakal, J. Rystrøm, R. Solomatin, Ömer Çağatan, A. Kundu, M. Bernstorff, S. Xiao, A. Sukhlecha, B. Pahwa, R. Poświata, K. K. GV, S. Ashraf, D. Auras, B. Plüster, J. P. Harries, L. Magne, I. Mohr, M. Hendriksen, D. Zhu, H. Gisserot-Boukhlef, T. Aarsen, J. Kostkan, K. Wojtasik, T. Lee, M. Šuppa, C. Zhang, R. Rocca, M. Hamdy, A. Michail, J. Yang, M. Faysse, A. Vatolin, N. Thakur, M. Dey, D. Vasani, P. Chitale, S. Tedeschi, N. Tai, A. Snegirev, M. Günther, M. Xia, W. Shi, X. H. Lù, J. Clive, G. Krishnakumar, A. Maksimova, S. Wehrli, M. Tikhonova, H. Panchal, A. Abramov, M. Ostendorff, Z. Liu, S. Clematide, L. J. Miranda, A. Fenogenova, G. Song, R. B. Safi, W.-D. Li, A. Borghini, F. Cassano, H. Su, J. Lin, H. Yen, L. Hansen, S. Hooker, C. Xiao, V. Adlakha, O. Weller, S. Reddy, and N. Muennighoff, "Mmteb: Massive multilingual text embedding benchmark," *arXiv preprint arXiv:2502.13595*, 2025.

[7] S. Sturua, I. Mohr, M. K. Akram, M. Günther, B. Wang, M. Krimmel, F. Wang, G. Mastrapas, A. Koukounas, A. Koukounas, N. Wang, and H. Xiao, "jina-embeddings-v3: Multilingual embeddings with task lora," 2024.

[8] X. Zhang, Y. Zhang, D. Long, W. Xie, Z. Dai, J. Tang, H. Lin, B. Yang, P. Xie, F. Huang, *et al.*, "mgte: Generalized long-context text representation and reranking models for multilingual text retrieval," in *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 1393–1412, 2024.

[9] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 2019.

[10] M. Henderson, R. Al-Rfou, B. Strope, Y. hsuan Sung, L. Lukacs, R. Guo, S. Kumar, B. Miklos, and R. Kurzweil, "Efficient natural language response suggestion for smart reply," 2017.
[11] A. Kusupati, G. Bhatt, A. Rege, M. Wallingford, A. Sinha, V. Ramanujan, W. Howard-Snyder, K. Chen, S. Kakade, P. Jain, and A. Farhadi, "Matryoshka representation learning," 2024.