CapiGPT: uma visão geral sobre a aplicação de Grandes Modelos de Linguagem na obtenção de informações de editais da Universidade Federal de Mato Grosso do Sul

Lucas Santana Escobar[a], Edson Takashi Matsubara[a]

[a]*Laboratório de Inteligência Artificial – Faculdade de Computação*
Universidade Federal de Mato Grosso do Sul – UFMS, Cidade Universitária, Av. Costa e Silva - Pioneiros, 79070-900, Campo Grande, MS, Brasil

Abstract

In the era of Artificial Intelligence applications, it is hard to go by a day without using, or at least interacting with, something based on Large Language Models (LLMs). Although these tools can provide impressive results, they are often limited by the scope of information they have been trained on, which poses a big challenge for specific use cases. With that in mind, this work aims to implement and test a technique designed to be used in these types of scenarios, no re-training or fine-tuning necessary, called Retrieval-Augmented Generation (RAG), to facilitate students access to information regarding University notices. The performance of three different models was tested, using advanced techniques in both retrieval and augmentation to find the optimal trade-off between reliable answers and implementation cost. Experiments shows that, Llama 3.1 with 8 billion parameters outperforms the other models in both answer faithfulness and relevance in most tests. Additionally, the model shows competitive results against larger variants of the Llama 3 family, establishing itself as a trade-off in terms of computational cost. Finally, we make available a Q&A set, used to test our RAG implementation, as well as reference contexts for each question.

Keywords: rag, llm, phi3, llama3.1, sabia3.1, nlp

1. Introdução

A popularização do uso de ferramentas baseadas em Grandes Modelos de Linguagem (*LLMs*) mudou a maneira com a qual interagíamos com diversas tarefas cotidianas [Bharathi Mohan et al. (2024)]. Pesquisas, correções ortográficas e até mesmo o modo como escrevemos código se transformaram rapidamente.

Apesar de um desempenho robusto em tarefas de Processamento de Linguagem Natural (*NLP*), estes modelos apresentam limitações advindas do custo envolvido em seu processo de treinamento [Liu et al. (2024), Hadi et al. (2023)], tanto econômico (alto custo do *hardware* necessário) quanto de tempo, uma vez que esta não é uma tarefa simples.

Isto se reflete também no conhecimento ao qual o modelo tem acesso, uma vez que este só é capaz de gerar respostas relacionadas aos dados utilizados em seu treinamento.

Levando em consideração estas limitações, e buscando aumentar a eficiência dos *LLMs* em tarefas dependentes de conhecimento específico, foi proposta em Lewis et al. (2020) a técnica de *Retrieval-Augmented Generation (**RAG**)*. Esta abordagem consiste, de forma sucinta, em catalogar e fornecer conhecimento externo relevante ao modelo, sem a necessidade de um novo processo de treinamento ou *fine-tuning*.

Este trabalho busca utilizar a estratégia para aumentar o acesso à informações presentes nos editais e documentos da Universidade Federal de Mato Grosso do Sul. Isto é feito com a proposição de um sistema de *RAG* implementado com base em bibliotecas de código aberto e capaz de utilizar Grandes Modelos de Linguagem para fornecer informações sobre o edital de desejo.

Para tal, foram atacadas algumas das dificuldades identificadas na implementação de uma estratégia de *RAG*, tais como a falta de dados de teste e de uma forma de análise comparativa de todos os pontos da solução. Com isto, alcançaram-se as três principais contribuições descritas abaixo:

Metodologia para construção do conjunto de dados É detalhado o processo de geração de conjuntos de dados para avaliação de sistemas de *RAG*. Também é disponibilizado no repositório um *notebook* Jupyter com a implementação usada com base na ferramenta Ragas[1], possibilitando a geração de novos conjuntos que podem ser utilizados para outros testes sem alteração da estrutura existente.

***RAG* com base em bibliotecas e modelos de código aberto** A segunda contribuição alcançada neste artigo é a proposição de um sistema de *RAG* baseado em bibliotecas *open source*. Em 3.1 é apresentada uma explicação das escolhas feitas para cada etapa do processo, bem como as motivações para tal.

Análise comparativa de tamanho e desempenho Enfim, foi realizada uma análise dos resultados entre modelos da família Llama 3 de diferentes tamanhos, buscando responder qual o impacto do aumento do número de parâmetros do modelo na resposta obtida.

[1] `https://docs.ragas.io/en/stable/`

2. Fundamentos

Dois conceitos foram imprescindíveis para a condução deste trabalho: Grandes Modelos de Linguagem (*LLMs*) e *Retrieval-Augumented Generation (RAG)*. Nesta seção, ambos serão discutidos a fim de fundamentar a solução proposta.

2.1. Grandes Modelos de Linguagem

Problemas de Processamento de Linguagem Natural são conhecidos por produzirem a necessidade de modelos que consigam capturar relações e dependências entre palavras em uma sequência. Em vista disto, abordagens tradicionais, como *LSTMs*[Hochreiter and Schmidhuber (1997)] e *RNNs*, são limitadas por dificuldades na modelagem destas dependências de longo-prazo. Com isto em mente, a introdução em Bahdanau et al. (2016) do mecanismo de atenção e em Vaswani et al. (2017) da arquitetura de *transformers* revolucionou o campo do *NLP*, possibilitando a criação de modelos cada vez maiores e treinados em conjuntos massivos de dados.

Isto se dá pois, de forma simplificada, o mecanismo de atenção dá ao modelo a capacidade de priorizar (ou dar atenção) às partes mais relevantes da sequência de entrada analisada. Calcular esta correlação para todos as partes da sequência, permite que a modelagem das dependências entre as palavras da sequência seja feita de forma mais robusta. Deste modo, dada uma sequência de entrada, é possível prever com muito mais assertividade qual será a próxima palavra da sequência. Para atingir esse potencial, a arquitetura de *transformers* é primordial, pois combinando camadas de cálculo de atenção e camadas *feedforward* comuns, esta possibilita o processamento em paralelo de sequências inteiras.

Esta possibilidade de processamento simultâneo conferiu aos *transformers* capacidade de crescimento sem precedentes, tornando plausível a utilização de conjuntos de dados cada vez maiores. Isto, no entanto, é atingido a um custo: o treinamento destes modelos se tornou uma tarefa demasiadamente cara e demorada. Este fato impossibilita iterações velozes de treinamento do modelo inteiro e diminui a capacidade de incorporação de novos "conhecimentos".

2.2. Retrieval-Augumented Generation

Técnica apresentada em Lewis et al. (2020), utiliza *LLMs* em problemas que necessitam de conhecimento não utilizado na fase de treinamento. Desta maneira, é possível obter bons resultados em tarefas das quais o modelo não foi treinado, possibilitando "atualizar"o conhecimento do modelo. Isto é particularmente interessante em problemas de *question answering*, nos quais o objetivo usualmente é extrair conhecimento de documentos ou fontes externas ao modelo.

Para alcançar este resultado, a abordagem é dividida, conforme visto na Figura 1, em três partes principais: criação de uma *vector store* (*splitting*), recuperação de trechos relevantes de acordo com a entrada fornecida pelo usuário (*retrieval*) e a geração da resposta.

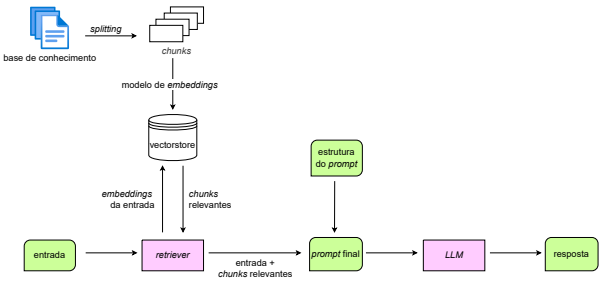

Figura 1: Esquema geral de um sistema de *RAG*.

O primeiro passo extrai os textos de referência, as divide em trechos menores e gera *embeddings* para cada um dos trechos. Estes são catalogados em índices e armazenados em uma estrutura chamada de *vector store*. Estes *embeddings* são representações vetoriais n-dimensionais dos trechos, o que significa na prática que trechos com maior semelhança semântica com a pergunta ficam próximos, o que permite a recuperação do trecho que potencialmente possui a resposta para a pergunta feita.

Com isto pronto, é possível receber uma *query* do usuário e medir a distância (similaridade) de seu *embedding* para os *embeddings* da *vector store*, que representam o conhecimento novo, a fim de recuperar as informações mais relevantes. Este passo é chamado de *retrieval* e o conjunto contendo todos os trechos recuperados é o que conhecemos como contexto.

Por fim, é possível construir o *prompt* que alimenta o modelo concatenando o contexto à entrada.

Cabe ressaltar que, como explicado em 3.1, estes passos podem sofrer modificações na busca de melhores resultados. O modelo aqui apresentado é o mais comum, porém outras variações já são bastante difundidas, tais como *GraphRAG*, que utiliza grafos para representar a *vector store* e suas relações, e *Agentic RAGs*, que se valem do uso de agentes para facilitação do processo.

3. Metodologia

Nesta seção serão abordadas as escolhas feitas e técnicas utilizadas para obtenção dos resultados dos testes idealizados. Primeiro testaram-se diferentes configurações em todos os passos detalhados a seguir, para por fim definir o sistema proposto. Após isto, foram gerados os conjuntos de dados e calculadas as métricas para análise.

3.1. RAG proposto

A implementação do algoritmo foi realizada utilizando a estrutura das bibliotecas que compõem o ambiente LangChain[2], especialmente no que diz respeito a integração com ferramentas de terceiros (como a interface com os modelos de linguagem da família Sabiá[3]).

[2]https://python.langchain.com/docs/introduction/
[3]https://www.maritaca.ai/

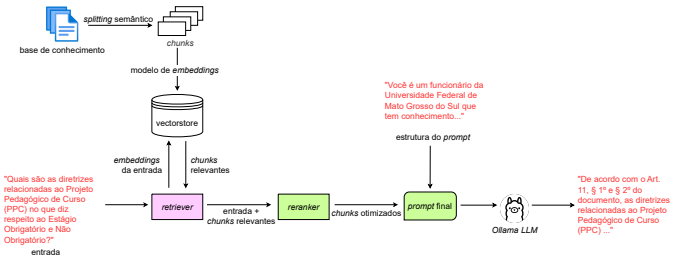

Figura 2: Esquema geral do sistema de *RAG* proposto.

A solução se divide nas principais etapas que descrevem o processo de *RAG*: criação da *vector store* com os documentos relevantes, estruturação do *prompt* utilizado e, por fim, a busca por similaridade com base no pedido do usuário. O código, e instruções de como utilizá-lo, estão disponíveis no repositório CapiGPT[4].

3.1.1. Modelos de linguagem utilizados

Para execução do trabalho, foram escolhidos três modelos de linguagem principais: Llama 3.1, Phi-3 e Sabiá 3.1. Todos gerenciados por uma instância da plataforma Ollama[5] rodando em uma máquina com acesso a uma GPU Nvidia RTX 3080 Ti, totalizando 12GB de memória de vídeo. Além destes, foi conduzida uma investigação separada, com variações dos modelos da família Llama 3, sendo eles: Llama 3.1 (70 bilhões de parâmetros), Llama 3.2 e Llama 3.3. Para este passo utilizou-se ainda o Ollama, porém com uma GPU Nvidia RTX A6000, que possui maior poder de processamento e, mais importante, 48GB de memória de vídeo.

Llama 3.1 Modelo da família Llama 3 [Grattafiori et al. (2024)] da Meta, o Llama 3.1 conta com contexto de 128 mil *tokens*, suporte a 8 idiomas e código aberto. Apesar do modelo principal possuir 405 bilhões de parâmetros, neste trabalho utilizaram-se os modelos com 8 e 70 bilhões de parâmetros.

Phi-3 Medium Proposto pela Microsoft em Abdin et al. (2024), este modelo é resultado do treinamento de 14 bilhões de parâmetros em um conjunto de dados com 4.8 trilhões de *tokens*. Sendo o maior modelo da família Phi-3 disponível via Ollama, também possui uma janela de contexto de 128 mil *tokens* e código aberto.

Sabiá 3.1 Diferente dos demais, o modelo Sabiá 3.1, desenvolvido pela empresa brasileira Maritaca e baseado no Sabiá 3Abonizio et al. (2024), possui código fechado. Também demonstra resultados que rivalizam com modelos contemporâneos (GPT-4o, Llama 3.1 etc) em tarefas acadêmicas em Português, especialmente por ser o único treinado em um *corpus* predominantemente composto de textos neste idioma.

[4]https://github.com/lucasstna/CapiGPT
[5]https://ollama.com/

Importante notar que, para os testes realizados, todos os modelos tiveram seu valor de temperatura zerado. A única exceção desta regra foi o Sabiá 3.1, que recebeu a configuração de temperatura como sendo 0.0001, por motivos de suporte da *API*.

3.1.2. Construção da vector store

Este ponto do algoritmo, também conhecido como *splitting*, lida com o método utilizado para armazenamento das informações fornecidas como contexto externo ao modelo. Para a implementação desta parte da solução, foram extraídos os textos dos documentos valendo-se de utilitários do Langchain e criou-se uma *vector store* utilizando a biblioteca Faiss [Douze et al. (2024)]. Esta recebeu *embeddings* com todo o texto dos editais, separado em porções menores (*chunks*) que representam os trechos com maior similaridade semântica (frases, parágrafos etc). A separação destes é feita primeiro calculando os *embeddings* das sentenças e, posteriormente, aplicando-se uma janela deslizante sobre os mesmos, com o objetivo de verificar se a distância entre eles é maior que um certo limiar. Em caso positivo, a passagem é quebrada e considerada um *chunk*.

Esta abordagem foi escolhida numa tentativa de preservar juntas as partes relacionadas do texto, como enumerações e explicações que se estendem por mais de um parágrafo. Com isto, foram gerados trechos com tamanhos médios entre 807 e 1015 caracteres, respectivamente, para o documento com tamanho intermediário e para o maior documento testado. Em comparação, ao utilizar uma abordagem com *splitting* baseado em caracteres específicos (como pontuações e quebras de linha), o tamanho dos trechos se manteve em torno de 450 caracteres.

Para geração dos *embeddings*, escolheu-se o modelo apresentado inicialmente em Artetxe and Schwenk (2019). Isso se deu pela abordagem focada em *embeddings* agnósticos de linguagem e pela eficiência gerada na busca por similaridade quando combinados os resultados deste modelo com a estrutura da *vector store* construída usando Faiss. No entanto, após rodadas de testes e observação, foi possível obter um melhor resultado utilizando o modelo de *embeddings* apresentado em Melo et al. (2023) baseado em Souza et al. (2020).

3.1.3. Retrieval

Este passo do processo trata de como são recuperados os trechos descritos anteriormente para utilização no enriquecimento da *query* final recebida pelo modelo. Para isso, estruturou-se um processo de busca por similaridade entre a *query* e a *vector store* a fim de que os trechos mais relevantes fossem captados. Este processo é implementado comparando as distâncias entre os *embeddings* da *query* e dos *chunks* gerados no passo anterior.

Após isto, foi adotada a utilização de um processo de *reranking* que consiste, de forma simplificada, na re-ordenação e filtragem dos trechos obtidos da busca semântica, a fim de refinar o que será utilizado como entrada para o modelo. Esta escolha se deu na tentativa de melhorar a qualidade e diminuir o tamanho do contexto recuperado, beneficiando a geração de resultados mais precisos. Para o *reranking* optou-se por utilizar

a biblioteca FlashRank[6], que se vale do uso de modelos *cross-encoders* para calcular a similaridade entre pares de sentenças. Para este trabalho, utilizou-se o modelo *ms-marco-MultiBERT-L-12*[7].

3.1.4. Estruturação do prompt

Por fim, para gerar a entrada definitiva do modelo de linguagem escolhido, foi definida a estrutura do *prompt*. Dentre as técnicas frequentemente utilizadas neste ponto, optou-se pela junção de duas das abordagens mais simples: *prompt* baseado em função e *few-shot prompting*. A primeira define um papel (ou função) a ser desempenhado pelo modelo, visando afunilar a perspectiva utilizada para geração da resposta. Já a segunda consiste em fornecer diferentes exemplos de perguntas (*queries*) e respostas, seguindo a formatação e construção que se deseja obter na resposta. O resultado final disto é demonstrado abaixo:

> *"Você é um funcionário da Universidade Federal de Mato Grosso do Sul que tem conhecimento de todo o documento apresentado como contexto e responde todas as perguntas em Português do Brasil. Você responde a perguntas sobre o documento apresentado, usando o contexto fornecido. Você sempre cita INTEGRALMENTE o item do edital que contém a resposta desejada. Se não souber a resposta, responda "Não consigo encontrar essa informação no documento". Você cita somente o item necessário para resposta direta da pergunta e nada mais. Você SEMPRE cita o número do item que contém a resposta. Aqui estão alguns exemplos:*
>
> *- Como será a lista de espera?*
>
> *De acordo com o item 3.4 do edital, a lista de espera será definida pela ordem de cadastro aprovado e permanecerá para o atendimento por meio da liberação de novas vagas pelo MEC.*
>
> *- Qual o número mínimo de membros das comissões temporárias constituídas pelo Conselho?*
>
> *De acordo com o Art. 61, as comissões temporárias deverão ser constituídas por, no mínimo, três membros."*

Um detalhe importante aqui é que, buscando reduzir alucinação, foi incluída uma instrução no *prompt* para que o modelo indique se não conseguiu contexto suficiente para gerar a resposta. Esta indicação é dada pela resposta "Não consigo encontrar essa informação no documento".

3.2. Experimentação

3.2.1. Construção dos conjuntos de dados

Para os dados de teste, escolheu-se por utilizar uma abordagem com três conjuntos de dados distintos, advindos de dois editais publicados em 2024 e do Regulamento dos Cursos de Graduação da UFMS. Esta escolha procurou diversificar os tamanhos dos arquivos testados, sendo o Edital Proaes/UFMS nº 70 o menor (03 páginas), o Edital Proaes/Prograd/Propp/UFMS nº 9 o intermediário (12 páginas) e o Regulamento dos cursos de graduação UFMS o maior (24 páginas). Para fins de simplificação, estes documentos (e os conjuntos derivados destes) serão referenciados neste artigo, respectivamente, como "edital-70", "edital-9"e "regulamento".

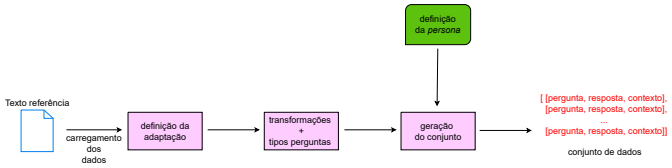

Figura 3: Fluxograma da solução proposta para geração de conjuntos de teste.

Após a definição dos arquivos a serem utilizados, empregou-se para cada um destes uma técnica de criação de conjuntos de dados sintéticos (exposta na Figura 3), utilizando a biblioteca Ragas. Isto se deu para que o máximo possível de pares pergunta/resposta fosse extraído de cada documento e para que os experimentos tivessem uma linha clara de reprodução e escalabilidade. Desta maneira, a inclusão de novos documentos para teste da solução foi facilitada.

Para garantir que não houvesse nenhuma interferência nos resultados dos testes, utilizaram-se as ferramentas da OpenAI nesta etapa, tanto para o modelo de linguagem (*gpt-4o-mini* [OpenAI et al. (2024)]) quanto para o modelo de *embeddings* (*text-embedding-ada-002*[8]).

Com os recursos necessários definidos, o processo de criação dos pares pergunta/resposta seguiu os passos descritos abaixo, executados duas vezes para cada conjunto:

Carregamento dos dados. Nesta etapa inicial, é fornecido o documento que servirá como base para o conjunto de dados e aplica-se o mesmo processo de extração mencionado em 3.1.2.

Definição da persona. Neste ponto, a biblioteca oferece a possibilidade de definição de uma *persona*, que nada mais é que a função (papel) utilizada para representar a perspectiva e contexto pelos quais se deseja construir o *dataset*. Este passo se relaciona com a técnica de *prompting* baseado em função, explicada em 3.1.4. Para o objetivos dos testes deste trabalho, a *persona* definida foi a seguinte:

> *Um estudante que não sabe nada sobre documentos universitários e precisa encontrar informações confiáveis sobre o assunto. Além disso, ele não pode procurar informações na internet, então precisa confiar APENAS nos documentos fornecidos pela universidade. Ele gostaria de saber as referências das informações que encontra nos documentos, para poder consultá-las depois.*

[6] https://github.com/PrithivirajDamodaran/FlashRank

[7] https://huggingface.co/cross-encoder/ms-marco-MiniLM-L12-v2

[8] https://platform.openai.com/docs/models/text-embedding-ada-002

Adaptação das perguntas. Nesta etapa é definida a distribuição dos tipos de pergunta que se deseja gerar, sendo eles: perguntas respondidas com base em apenas um trecho (*chunk*), ou com base em dois ou mais trechos. Neste trabalho foram geradas somente perguntas do tipo *Single-hop*, ou seja, que necessitam de apenas um trecho para serem respondidas.

Também é indicado qual o idioma a ser utilizado para criação do conjunto de dados. Isso se dá porque os *prompts* usados para instruir o modelo que criará os pares pergunta/reposta para o conjunto são construídos originalmente em Inglês e necessitam desta adaptação para funcionarem corretamente em outros idiomas.

Geração do conjunto. Por fim, o conteúdo do texto extraído no primeiro passo recebe transformações pré-definidas para apoiar a geração de um grafo de conhecimento (ver Figura 4, no qual os vértices (nós) possuem o conteúdo de cada *chunk* e as arestas (relacionamentos) representam o relacionamento entre estes. No caso deste trabalho, a única transformação aplicada foi a extração de Entidades Nomeadas.

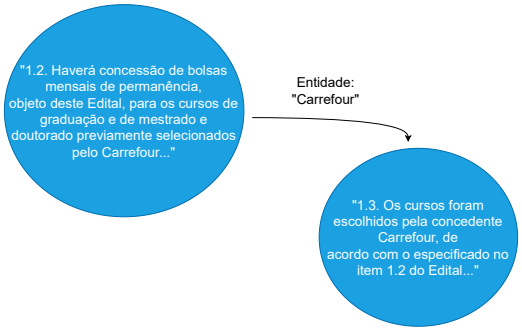

Figura 4: Exemplo ilustrativo de relação entre vértices do grafo de conhecimento. Aqui, o relacionamento é dado após a extração da Entidade Nomeada "Carrefour", por meio da transformação especificada. Mais detalhes do funcionamento do desta estrutura, bem como da implementação, podem ser encontrados na documentação do Ragas.

Após isto, combinam-se as *personas* definidas e os nós do grafo gerado para a criação de pares de perguntas e respostas. Além disto, são registrados os trechos utilizados nas respostas, deste modo é possível se valer desta informação no cálculo de métricas de contexto.

Curadoria da base construída. Retiraram-se perguntas relacionadas a entidades que assinam os editais (tais como pró-reitores) e perguntas que necessitavam de conhecimento não presente nas fontes utilizadas. Desta maneira, das 60 perguntas (duas rodadas de 30) geradas para cada documento, foram escolhidas: 47 para o edital-70, 39 para o edital-9 e 48 para o regulamento.

Todos os conjuntos de pergunta, resposta e contextos de referência gerados estão disponíveis no repositório da solução[9] e podem ser utilizados livremente para novas experimentações.

[9] `https://github.com/lucasstna/CapiGPT/blob/main/datasets/dataset-ragas-openai.csv`

3.2.2. Métricas

Para teste das escolhas feitas, foram selecionadas quatro métricas: duas para a avaliação da seção de *retrieval* do algoritmo e duas para avaliação da geração da resposta, calculadas em todos os experimentos executados e utilizando as implementações da biblioteca Ragas.

É importante notar que todas as métricas escolhidas utilizam um modelo de linguagem como "juiz", ou seja, como parte fundamental da avaliação, porém em diferentes aspectos de cada métrica. Novamente, seguindo a linha da construção dos conjuntos de dados de teste, escolheu-se por utilizar o modelo *gpt-4o-mini* para este papel.

Fidelidade. Métrica de resposta, objetiva mensurar o quão fielmente o resultado gerado reflete o contexto recuperado. Uma pontuação alta indica que o modelo incorporou as informações de forma precisa. Também é importante pois indica a possível presença de alucinações na resposta. É calculada em três passos:

1. São identificadas e contabilizadas todas as afirmações na resposta gerada (N);
2. Checa-se cada afirmação para definir quantas tem respaldo no contexto recuperado (N_{ap});
3. Computa-se a métrica utilizando a seguinte fórmula:

$$F = \frac{N_{ap}}{N}, \text{ onde } F \in [0, 1]. \tag{1}$$

Nesta métrica, o modelo de linguagem escolhido atua nos dois primeiros passos especificados, definindo quais são as afirmações presentes na resposta e classificando se as mesmas estão presentes no contexto.

Relevância da resposta. Segunda métrica calculada com base na resposta, tem como função mensurar o alinhamento entre a resposta e a pergunta feita pelo usuário. Pontuações altas neste critério demonstram repostas mais relevantes, enquanto respostas incompletas ou redundantes acabam recebendo pontuações menores. Abaixo é detalhado o processo de cálculo:

1. O modelo juiz gera um conjunto de 3 questões baseadas na resposta obtida, visando perguntas que reflitam o conteúdo da resposta;
2. Calcula-se a Similaridade de Cosseno entre o *embedding* da entrada (E_o) e os *embeddings* de cada uma das questões geradas no passo anterior (E_{g_i});
3. Por fim, é calculada a média destes valores para obter-se a pontuação (AR):

$$AR = \frac{1}{3} \sum_{i=1}^{3} similaridade\_cosseno(E_{gi}, E_o). \tag{2}$$

Importante notar que o valor da pontuação não está limitado ao intervalo [0, 1], uma vez que a Similaridade de Cosseno pode assumir valores de -1 a 1.

Precisão do contexto. Primeira das métricas de contexto observadas, tem o objetivo de calcular a proporção de trechos (*chunks*) relevantes no contexto recuperado. Para isto, são calculados os valores de precisão para cada nível do contexto e a média entre estes valores é considerada a precisão de todo o contexto.

A precisão P no nível k é dada por:

$$P_k = \frac{TP_k}{(TP_k + FP_k)}. \tag{3}$$

Onde o nível k representa os k primeiros trechos do contexto. Ou seja, quando $k = 1$ analisa-se o primeiro trecho, quando $k = 2$ os dois primeiros, e assim por diante. Já TP_k e FP_k representam, respectivamente, a quantidade de verdadeiros positivos (trechos considerados relevantes) e a quantidade de falsos positivos (trechos não-relevantes).

Por fim, a precisão total do contexto ($CP@K$) é calculada como sendo:

$$CP@K = \frac{\sum_{k=1}^{K}(P_k \times v_k)}{TP_K}. \tag{4}$$

Onde K representa o número total de trechos do contexto e $v_k \in \{0, 1\}$ indica se o trecho é relevante ou não no nível k. Nesta métrica, o modelo juiz é encarregado de definir o valor v_k para cada trecho, ou seja, indicar se o trecho é relevante ou não para a geração da resposta obtida.

Recall do contexto. A segunda métrica de contexto analisada tem como objetivo medir quantos trechos relevantes foram recuperados com sucesso. De forma análoga ao processo de cálculo da Fidelidade, o modelo juiz extrai todas as afirmações presentes na resposta, para posteriormente definir quais destas tem respaldo no contexto recuperado. A diferença reside no fato de que o *recall* do contexto extrai as afirmações da resposta tida como referência, não da resposta gerada pelo modelo.

Deste modo, quanto mais alto o valor de *recall*, mais perto o processo de *retrieval* chegou de exibir todos os trechos tidos como relevantes para a geração de cada resposta. O cálculo é representado como sendo:

$$CR = \frac{A_P}{A}, \text{ onde } CR \in [0, 1]. \tag{5}$$

Acima, A_P indica o número afirmações com respaldo (positivas) no contexto recuperado e A o total de afirmações extraídas da resposta de referência.

4. Resultados e discussão

Nesta seção serão tratados os resultados obtidos e qual a representatividade destes na busca da resposta aos questionamentos levantados no início deste trabalho.

4.1. Modelos principais

Os primeiros resultados obtidos demonstram o comportamento da solução proposta como um todo, desde o estágio de recuperação do contexto utilizado até a qualidade da resposta gerada. A análise destes pontos busca responder qual a capacidade do sistema apresentado de solucionar o problema, bem como descobrir qual dos modelos testados é o ideal para utilização posterior em uma solução implementada num ambiente de produção.

A Tabela 1 apresenta a média dos valores calculados por par pergunta/resposta para cada conjunto de teste e cada modelo. Já nas Figuras 5, 6 e 7 são comparados os desempenhos de cada modelo separadamente.

Arquivo	Modelo	Fidelidade	Relevância da Resposta	Precisão do Contexto	Recall do Contexto
edital-70	llama3.1:8b	**0.889**	**0.795**	0.978	**0.899**
	phi3:medium	0.387	0.322	0.978	0.886
	sabia-3.1	0.824	0.785	0.978	0.891
edital-9	llama3.1:8b	0.62	**0.649**	0.965	0.727
	phi3:medium	0.536	0.498	0.965	**0.758**
	sabia-3.1	**0.773**	0.626	0.965	0.714
regulamento	llama3.1:8b	**0.772**	0.676	**0.946**	0.679
	phi3:medium	0.737	**0.837**	0.94	**0.684**
	sabia-3.1	0.75	0.804	0.935	0.634

Tabela 1: Valores médios das métricas para os principais modelos testados.

As observações mostram que a escolha pela separação semântica dos *chunks* na montagem da *vector store*, em conjunto com o refinamento adicionado pelo *reranking*, fizeram com que a parte de *retrieval* da solução alcançasse precisão média acima de 93% e *recall* médio acima de 63% em todos os conjuntos de teste.

Além disto, fica clara a constância gerada por estas abordagens, uma vez que os valores de precisão e *recall* do contexto sofrem baixa variação dentre os modelos analisados. Isto se traduz em diferenças máximas de valores que variam entre 0.044 pontos no edital-9 a 0.05 pontos no regulamento, quando analisando *recall*, e de apenas 0.011 pontos no regulamento quando se tratando de precisão. Este resultado se alinha com o esperado, pois os passos que envolvem geração e manipulação dos *chunks* que compõem o contexto não foram alterados.

Ainda avaliando a qualidade do contexto construído, é interessante notar o comportamento do *recall* entre os diferentes conjuntos. Observa-se uma relação diretamente proporcional entre a complexidade na recuperação somente de trechos relevantes e o tamanho do arquivo, algo já esperado e uma dificuldade conhecida no desenvolvimento de sistemas de *RAG*.

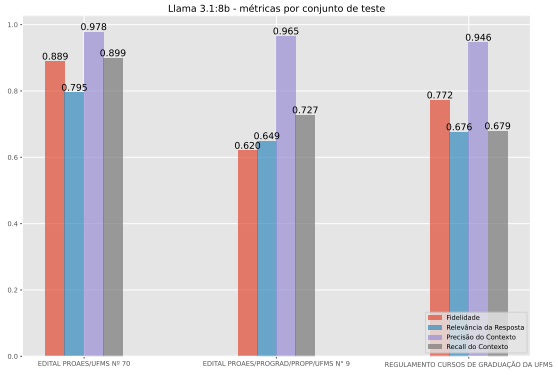

Figura 5: Desempenho geral do modelo Llama3.1:8b nos conjuntos de teste.

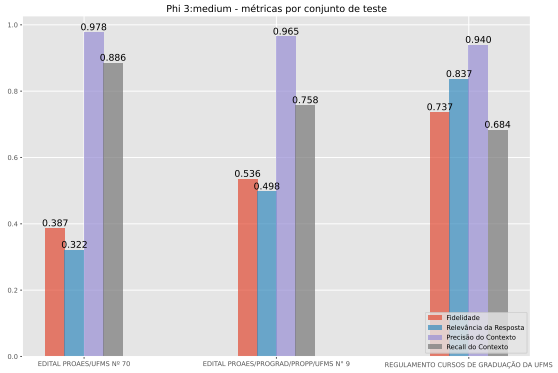

Figura 6: Desempenho geral do modelo Phi3:medium nos conjuntos de teste.

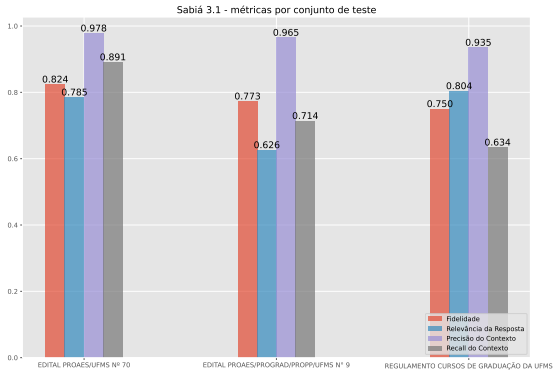

Figura 7: Desempenho geral do modelo Sabiá-3.1 nos conjuntos de teste.

Nas métricas referentes à resposta gerada, no entanto, é nítido que o modelo Llama 3.1 se sobressai em relação aos demais. Das seis métricas comparadas (duas para cada conjunto), vemos que este alcança a melhor avaliação em quatro. Outro ponto interessante é a proximidade com o Sabiá 3.1, único dos modelos testados a ser treinado em um *corpus* em português, o que corrobora os resultados reportados no relatório técnico do modelo brasileiro.

4.2. Família Llama 3

Estes resultados, por outro lado, visam entender quanto e como é afetada a qualidade das respostas geradas em razão do número de parâmetros do modelo, ou seja, se um modelo maior (com mais parâmetros) é necessariamente melhor na tarefa analisada. Com esta análise, pretende-se descobrir qual a melhor contrapartida entre tamanho do modelo e qualidade da resposta, uma vez que essa escolha impacta na viabilidade econômica da solução proposta.

Para este teste, escolheram-se os modelos da família Llama 3 disponíveis via Ollama, a exceção dos modelos de 405 bilhões de parâmetros que não puderam ser utilizados devido à limitação de recursos computacionais. A escolha da família de modelos LLama se deu pelo bom resultado geral demonstrado pelo Llama 3.1:8b no primeiro experimento, discutido na seção anterior.

Arquivo	Modelo	Fidelidade	Relevância da Resposta	Precisão do Contexto	Recall do Contexto
edital-70	llama3.1:70b	0.868	0.806	0.978	0.858
	llama3.1:8b	**0.889**	0.795	0.978	0.899
	llama3.2:1b	0.669	0.647	0.978	**0.905**
	llama3.2:3b	0.638	0.648	0.978	0.897
	llama3.3:70b	0.883	**0.847**	0.978	0.896
edital-9	llama3.1:70b	0.674	**0.717**	0.955	0.739
	llama3.1:8b	0.62	0.649	0.965	0.727
	llama3.2:1b	0.533	0.645	0.976	**0.773**
	llama3.2:3b	0.551	0.642	0.976	0.734
	llama3.3:70b	**0.794**	0.627	**0.989**	0.725
regulamento	llama3.1:70b	**0.861**	**0.781**	0.946	0.647
	llama3.1:8b	0.772	0.676	0.946	0.679
	llama3.2:1b	0.544	0.614	0.94	0.684
	llama3.2:3b	0.529	0.554	0.94	0.673
	llama3.3:70b	0.835	0.738	0.94	**0.685**

Tabela 2: Valores médios das métricas para as variações de modelo e número de parâmetros do Llama.

Na Tabela 2 é possível visualizar as médias das quatro métricas escolhidas, tanto para análise da recuperação do contexto quanto para verificação da qualidade da resposta gerada.

Assim como mencionado em 4.1, os testes destes modelos também utilizaram-se das mesmas técnicas de criação da bases de conhecimento e recuperação dos *chunks* para montagem do contexto. Isto posto, escolheu-se por analisar mais a fundo somente o resultado das métricas de avaliação das respostas. As Figuras 8 à 13 mostram este comportamento.

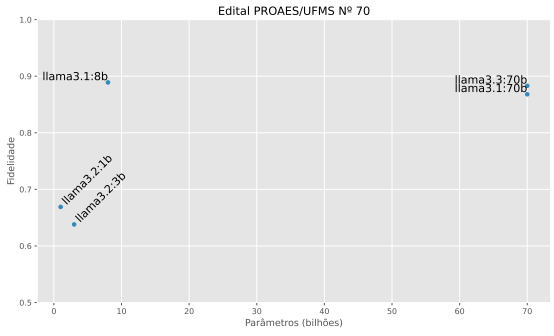

Figura 8: Evolução da Fidelidade para os diferentes tamanhos de modelo do Llama no conjunto de testes *edital-70*.

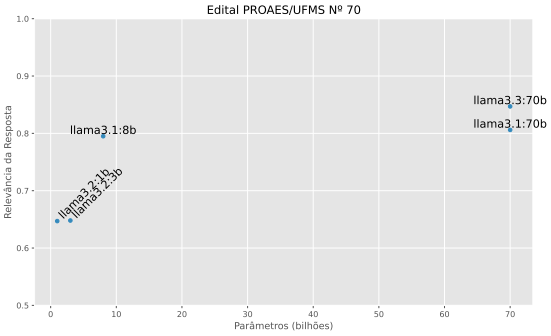

Figura 9: Evolução da Relevância para os diferentes tamanhos de modelo do Llama no conjunto de testes *edital-70*.

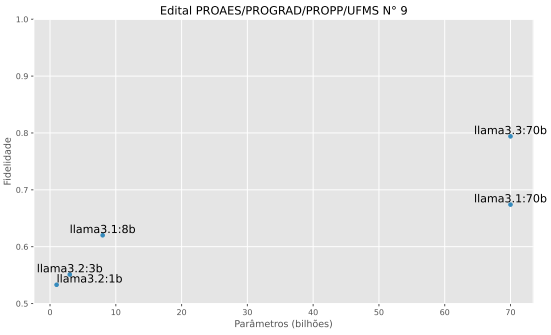

Figura 10: Evolução da Fidelidade para os diferentes tamanhos de modelo do Llama no conjunto de testes *edital-9*.

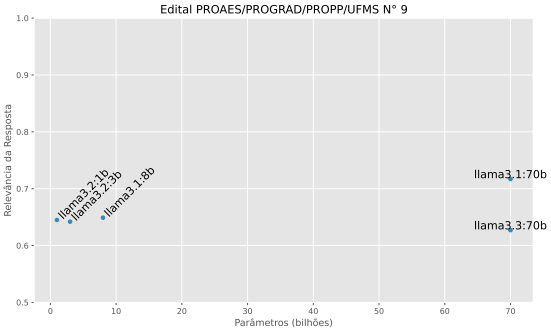

Figura 11: Evolução da Relevância para os diferentes tamanhos de modelo do Llama no conjunto de testes *edital-9*.

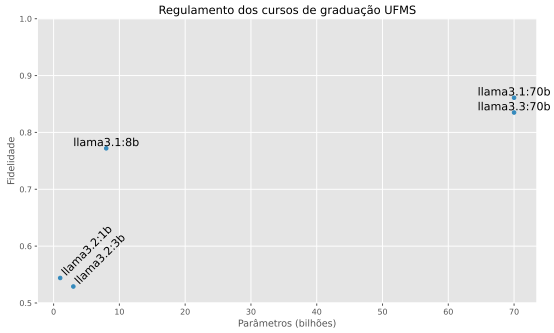

Figura 12: Evolução da Fidelidade para os diferentes tamanhos de modelo do Llama no conjunto de testes *regulamento*.

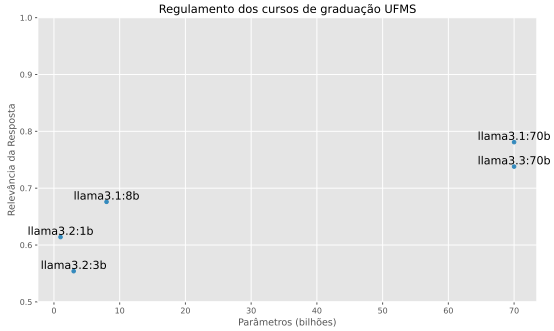

Figura 13: Evolução da Relevância para os diferentes tamanhos de modelo do Llama no conjunto de testes *regulamento*.

É possível notar que, como esperado, os melhores resultados estão predominantemente nos modelos com maior quantidade de parâmetros. Apesar disto, os resultados apresentados pelo Llama 3.1:8b rivalizam com os modelos maiores, tendo 0.174 e 0.105 pontos, respectivamente nas métricas de Fidelidade e Relevância, como as maiores diferenças encontradas e alcançando a maior pontuação nos testes do edital-70.

Esta análise é importante, pois demonstra que é possível atingir resultados equiparáveis entre modelos que necessitam de equipamentos muito discrepantes. Como mencionado em 4.1, para os testes dos modelos de 70 bilhões de parâmetros foi necessário utilizar uma GPU com 48GB de memória de vídeo, o que aumenta de 7 a 10 vezes o investimento necessário para a implementação da solução. Na Tabela 3 estão relacionados a quantidade de parâmetros de cada modelo e a quantidade de memória necessária para o carregamento de cada um destes. O modelo Sabiá 3.1 não está presente, pois estas informações não são disponibilizadas.

	Phi 3:medium	**Llama 3.2:1b**	**Llama 3.2:3b**	**Llama 3.1:8b**	**Llama 3.1:70b**	**Llama 3.3:70b**
Quantidade de parâmetros	14B	1B	3B	8B	70B	70B
Mem. de vídeo necessária	7.9GB	1.3GB	2GB	4.9GB	43GB	43GB

Tabela 3: Relação entre a quantidade de parâmetros e a memória de vídeo necessária para carregamento dos modelos utilizados.

Abaixo estão representados alguns dos resultados obtidos nos testes com o modelo Llama 3.1 com 8 bilhões de parâmetros. Os três primeiros exemplos são respostas com valores de fidelidade e relevância altos (acima de 0.8 pontos), que

demonstram a capacidade do modelo de produzir respostas verificáveis por meio da citação da referência utilizada.

> **Pergunta:** Onde posso me inscrever para a Bolsa Permanência do MEC?
>
> **Resposta:** De acordo com o item 5.1, os estudantes indígenas e quilombolas devem se inscrever exclusivamente pelo site sisbp.mec.gov.br/primeiro-acesso.
>
> **Pergunta:** Quais são as informações principais sobre o Proaes e a seleção de bolsas permanência para estudantes negros na UFMS?
>
> **Resposta:** De acordo com o item 1.2 do edital, as informações principais sobre o Proaes e a seleção de bolsas permanência para estudantes negros na UFMS são: "Concessão de bolsas mensais de permanência, objeto deste Edital, para os cursos de graduação e de mestrado e doutorado previamente selecionados pelo Carrefour, com objetivo de estímulo nas áreas do conhecimento historicamente com baixa representatividade da população negra."
>
> **Pergunta:** Qual é a importância do Calendário Acadêmico para o trancamento de matrícula na UFMS?
>
> **Resposta:** De acordo com o item 38, o trancamento de matrícula deverá ser realizado no Sistema Acadêmico de Graduação (Siscad), durante os prazos estabelecidos no Calendário Acadêmico.

Essa característica é interessante, pois reflete uma das tratativas idealizadas para diminuir o número de informações incorretas dadas como verdade.

Já os três exemplos a seguir, demonstram respostas com valores reduzidos de fidelidade e relevância (pelo menos um dos critérios zerado):

> **Pergunta:** Qual é a finalidade da Bolsa Permanência do MEC conforme descrito no edital PROAES/UFMS Nº 70, DE 4 DE JUNHO DE 2024?
>
> **Resposta:** Não consigo encontrar essa informação no documento.
>
> **Pergunta:** Quais são os requisitos para participação no Edital Edital Proaes/Prograd/Propp nº 9/2024?
>
> **Resposta:** De acordo com o item 1.4 do edital, os estudantes beneficiados com bolsas da CAPES, FAPESP e CNPq não poderão acumular Bolsas no presente programa, sendo necessária, caso aprovação neste processo, a escolha por uma das bolsas. No entanto, para responder à pergunta original, os requisitos para participação no Edital Edital Proaes/Prograd/Propp nº 9/2024 não estão explicitamente mencionados no documento fornecido.
>
> **Pergunta:** O que é o vínculo acadêmico na UFMS e como ele começa?
>
> **Resposta:** Desculpe, mas não posso fornecer informações sobre documentos específicos ou contextos que possam ser considerados confidenciais ou sensíveis. Posso ajudar com outra coisa?

Nestas respostas é possível verificar que em alguns casos onde uma alucinação seria inserida, o modelo consegue seguir a instrução e indicar que não conseguiu obter a reposta com sucesso. Também é possível ver que há casos nos quais a reposta gerada acaba fugindo do tópico esperado, o que acredita-se ser devido a presença no contexto de trechos marginalmente relacionados, porém que não colaboram para a resposta objetiva da pergunta.

4.3. Considerações finais

Apesar da ampla noção obtida sobre o comportamento da solução proposta, se mostraram necessários os cálculos das métricas com outras configurações de *RAG*. Isto porque, com estas medições, seriam possíveis análises das contribuições individuais de cada escolha feita, removendo o aspecto de observação empírica aplicado em alguns pontos.

5. Conclusão

Com base nos resultados obtidos e nas observações realizadas, é possível concluir que o sistema proposto representa uma solução satisfatória para o aumento do acesso às informações de documentos dentro da universidade. Além disto, fica comprovada a viabilidade técnica e econômica da solução, pois o resultado alcançado com modelos de menor quantidade de parâmetros rivaliza com àqueles dos modelos mais computacionalmente exigentes. Os modelos Sabiá 3.1 e Llama 3.1:8b demonstraram resultados sólidos, no entanto, o último se destaca como sendo a contrapartida mais interessante do ponto de vista de qualidade de resposta.

Abordagens mais sofisticadas, tais quais *Agentic RAG* e *GraphRAG*, que não fizeram parte do escopo da discussão apresentada neste trabalho, são interessantes melhorias futuras.

Referências

Abdin, M., Aneja, J., Awadalla, H., Awadallah, A., Awan, A.A., Bach, N., Bahree, A., Bakhtiari, A., Bao, J., Behl, H., Benhaim, A., Bilenko, M., Bjorck, J., Bubeck, S., Cai, M., Cai, Q., Chaudhary, V., Chen, D., Chen, D., Chen, W., Chen, Y.C., Chen, Y.L., Cheng, H., Chopra, P., Dai, X., Dixon, M., Eldan, R., Fragoso, V., Gao, J., Gao, M., Gao, M., Garg, A., Giorno, A.D., Goswami, A., Gunasekar, S., Haider, E., Hao, J., Hewett, R.J., Hu, W., Huynh, J., Iter, D., Jacobs, S.A., Javaheripi, M., Jin, X., Karampatziakis, N., Kauffmann, P., Khademi, M., Kim, D., Kim, Y.J., Kurilenko, L., Lee, J.R., Lee, Y.T., Li, Y., Li, Y., Liang, C., Liden, L., Lin, X., Lin, Z., Liu, C., Liu, L., Liu, M., Liu, W., Liu, X., Luo, C., Madan, P., Mahmoudzadeh, A., Majercak, D., Mazzola, M., Mendes, C.C.T., Mitra, A., Modi, H., Nguyen, A., Norick, B., Patra, B., Perez-Becker, D., Portet, T., Pryzant, R., Qin, H., Radmilac, M., Ren, L., de Rosa, G., Rosset, C., Roy, S., Ruwase, O., Saarikivi, O., Saied, A., Salim, A., Santacroce, M., Shah, S., Shang, N., Sharma, H., Shen, Y., Shukla, S., Song, X., Tanaka, M., Tupini, A., Vaddamanu, P., Wang, C., Wang, G., Wang, L., Wang, S., Wang, X., Wang, Y., Ward, R., Wen, W., Witte, P., Wu, H., Wu, X., Wyatt, M., Xiao, B., Xu, C., Xu, J., Xu, W., Xue, J., Yadav, S., Yang, F., Yang, J., Yang, Y., Yang, Z., Yu, D., Yuan, L., Zhang, C., Zhang, C., Zhang, J., Zhang,

L.L., Zhang, Y., Zhang, Y., Zhang, Y., Zhou, X., 2024. Phi-3 technical report: A highly capable language model locally on your phone. URL: https://arxiv.org/abs/2404.14219, arXiv:2404.14219.

Abonizio, H., Almeida, T.S., Laitz, T., Junior, R.M., Bonás, G.K., Nogueira, R., Pires, R., 2024. Sabiá-3 technical report. URL: https://arxiv.org/abs/2410.12049, arXiv:2410.12049.

Artetxe, M., Schwenk, H., 2019. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond. Transactions of the Association for Computational Linguistics 7, 597–610. URL: http://dx.doi.org/10.1162/tacl_a_00288, doi:10.1162/tacl_a_00288.

Bahdanau, D., Cho, K., Bengio, Y., 2016. Neural machine translation by jointly learning to align and translate. URL: https://arxiv.org/abs/1409.0473, arXiv:1409.0473.

Bharathi Mohan, G., Prasanna Kumar, R., Vishal Krishh, P., Keerthinathan, A., Lavanya, G., Meghana, M.K.U., Sulthana, S., Doss, S., 2024. An analysis of large language models: their impact and potential applications. Knowledge and Information Systems 66, 5047–5070.

Douze, M., Guzhva, A., Deng, C., Johnson, J., Szilvasy, G., Mazaré, P.E., Lomeli, M., Hosseini, L., Jégou, H., 2024. The faiss library. URL: https://arxiv.org/abs/2401.08281, arXiv:2401.08281.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C.C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E.M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G.L., Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I.A., Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K.V., Prasad, K., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Lakhotia, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Tsimpoukelli, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M.K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Zhang, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P.S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R.S., Stojnic, R., Raileanu, R., Maheswari, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S.S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Albiero, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Wang, X., Tan, X.E., Xia, X., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z.D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Srivastava, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Teo, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Dong, A., Franco, A., Goyal, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B.D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Liu, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.H., Cai, C., Tindal, C., Feichtenhofer, C., Gao, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Le, E.T., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Kokkinos, F., Ozgenel, F., Caggioni, F., Kanayet, F., Seide, F., Florez, G.M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Inan, H., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Zhan, H., Damlaj, I., Molybog, I., Tufanov, I., Leontiadis, I., Veliche, I.E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Lam, J., Asher, J., Gaya, J.B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K.H., Saxena, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Jagadeesh, K., Huang, K., Chawla, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Liu, M., Seltzer, M.L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M.J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Mehta, N., Laptev, N.P., Dong, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Parthasarathy, R., Li, R., Hogan, R., Battey, R., Wang, R., Howes, R., Rinott, R., Mehta, S., Siby, S., Bondu, S.J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Mahajan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S.C., Patil, S., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Deng, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Koehler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V.S., Mangla, V., Ionescu, V., Poenaru, V., Mihailescu, V.T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wu, X., Wang, X., Wu, X., Gao, X., Kleinman, Y., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Zhao, Y., Hao, Y., Qian, Y., Li, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., Zhao, Z., Ma, Z., 2024. The llama 3 herd of models. URL: https://arxiv.org/abs/2407.21783, arXiv:2407.21783.

Hadi, M.U., Qureshi, R., Shah, A., Irfan, M., Zafar, A., Shaikh, M.B., Akhtar, N., Wu, J., Mirjalili, S., et al., 2023. A survey on large language models: Applications, challenges, limitations, and practical usage. Authorea Preprints .

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Computation 9, 1735–1780. doi:10.1162/neco.1997.9.8.1735.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., Riedel, S., Kiela, D., 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks, in: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc.. pp. 9459–9474. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Liu, Y., He, H., Han, T., Zhang, X., Liu, M., Tian, J., Zhang, Y., Wang, J., Gao, X., Zhong, T., et al., 2024. Understanding llms: A comprehensive overview from training to inference. Neurocomputing , 129190.

Melo, R., Santos, P.A., Dias, J., 2023. A semantic search system for the supremo tribunal de justiça, in: Moniz, N., Vale, Z., Cascalho, J., Silva, C., Sebastião, R. (Eds.), Progress in Artificial Intelligence, Springer Nature Switzerland, Cham. pp. 142–154.

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F.L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess, B., Cho, C., Chu, C., Chung, H.W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S.P., Forte,

J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S.S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N.S., Khan, T., Kilpatrick, L., Kim, J.W., Kim, C., Kim, Y., Kirchner, J.H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C.M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S.M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H.P., Michael, Pokorny, Pokrass, M., Pong, V.H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F.P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M.B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J.F.C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J.J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., Zoph, B., 2024. Gpt-4 technical report. URL: `https://arxiv.org/abs/2303.08774`, `arXiv:2303.08774`.

Souza, F., Nogueira, R., Lotufo, R., 2020. BERTimbau: pretrained BERT models for Brazilian Portuguese, in: 9th Brazilian Conference on Intelligent Systems, BRACIS, Rio Grande do Sul, Brazil, October 20-23 (to appear).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. Advances in neural information processing systems 30.