Eficiência de LLMs para entendimento das Especificações da 3GPP usando abordagens como RAG e Fine-Tunning, e avaliação com Acurácia e RAGAS

José de Arimatéa Passos Lopes Júnior

17 November 2024

Resumo

This work aims to evaluate the performance of small and large language models (LLMs) in answering questions related to telecommunications, leveraging two main datasets: TeleQnA, a question-and-answer set specific to telecommunications, and TSpec-LLM, an extensive repository of technical documents from multiple 3GPP releases. A Retrieval-Augmented Generation (RAG) system was implemented to enhance response accuracy by incorporating relevant context from TSpec-LLM into the TeleQnA queries. This setup enables a comparative analysis of the performance of small and large language models, with a focus on assessing whether fine-tuning a smaller model on the telecommunications-specific dataset improves its performance.

1 Introdução

A crescente utilização de Modelos de Linguagem de Grande Escala (LLMs, do inglês *Large Language Models*) tem transformado diversas áreas do conhecimento, incluindo as telecomunicações, onde a complexidade técnica e a constante atualização dos padrões e especificações exigem modelos capazes de compreender e interpretar documentos técnicos extensos e detalhados. Nesse contexto, a avaliação e aprimoramento de modelos que possam auxiliar na interpretação de normas técnicas e respostas a perguntas específicas de telecomunicações tornam-se relevantes, principalmente para assegurar uma compreensão correta e precisa de informações contidas em documentos normativos do 3GPP (*3rd Generation Partnership Project*).

O objetivo deste trabalho é comparar a performance de diferentes modelos de linguagem de grande escala (LLMs) ao responder a perguntas de um *dataset* específico, o TeleQnA, um conjunto de perguntas e respostas voltado especificamente para telecomunicações. Também foi utilizado o *dataset* TSpec-LLM, um repositório extenso de documentos técnicos do 3GPP que abrange múltiplas *Releases*, que servirá como base para implementar um sistema de Geração Aumentada por Recuperação (RAG, *Retrieval-Augmented Generation*), para enriquecer o contexto das respostas às perguntas do TeleQnA, permitindo uma análise comparativa do desempenho de três modelos de linguagem distintos. Além disso, também será avaliado o impacto de um *fine-tuning* em um modelo menor, visando verificar se essa adaptação ao domínio específico de telecomunicações melhora significativamente sua performance, assim como foi feito em [7].

Os experimentos foram conduzidos com o intuito de avaliar a precisão dos modelos na seleção de respostas corretas e na geração de conteúdo baseado em perguntas do *Release* 17 do 3GPP, testando tanto a capacidade dos modelos de responder a perguntas diretas quanto a sua precisão ao utilizar RAG para recuperar e processar informações adicionais do TSpec-LLM. As métricas de avaliação aplicadas consideram a acurácia, quando são fornecidas opções de respostas, e um conjunto de métricas específicas do RAGAS (*Retrieval-Augmented Generation Assessment Suite*) para as respostas geradas sem opções, com o objetivo de oferecer uma análise comparativa entre os modelos.

2 *Dataset*

Dois *Datasets* diferentes foram usados neste trabalho o TeleQnA e o TSPEC-LLM.

2.1 TeleQnA

O TeleQnA[3] é um *dataset* voltado para telecomunicações, contendo 10.000 perguntas e respostas extraídas de diversas fontes, incluindo normas e artigos de pesquisa. Organizado em cinco categorias principais — *research publications*, *research overview, standards specifications*, *standards overview* e *lexicon* — o *dataset* oferece uma cobertura abrangente do conhecimento em telecomunicações. Com aproximadamente 6 milhões de palavras em 25.000 páginas, o TeleQnA fornece uma representação detalhada de tópicos técnicos e gerais da área, sendo uma ferramenta essencial para avaliar a compreensão e a proficiência em normas, pesquisas e terminologias específicas do setor.

Cada item do *Dataset*[10] é composto por uma pergunta, opções, resposta correta e explicação da resposta. Um exemplo pode ser visto abaixo:

Question 1: What advantages does a data center-enabled HAP (High Altitude Platform) offer from an energy perspective?

- Option 1: Saves cooling energy and harvests solar energy
- Option 2: Uses renewable energy and offsets carbon emissions
- Option 3: Deploys large-scale solar panels and batteries
- Option 4: Requires less energy for communication links

Answer: Option 1: Saves cooling energy and harvests solar energy
Explanation: The data center-enabled HAP saves cooling energy due to its location at the naturally low temperature stratosphere and harvests solar energy using large solar panels.
Category: Research publications

O TeleQnA foi usado para realizar o treinamento do modelo da *Unsloth*[13] e para realizar o teste de comparação entre os três modelos apresentados neste trabalho.

Para os testes serão usadas somente perguntas referentes a *Release* 17 da 3GPP, pois é a *Release* mais recente e completa do *Dataset* TSpec-LLM[8]. Dessa forma, será possível avaliar com mais precisão a efeciência do RAG, pois é mais provável que o contexto da pergunta do TeleQnA

exista no *Dataset* TSpec-LLM, e consequentemente possa ser encontrada na busca utilizando RAG. O número de perguntas sobre a *Release* 17 no *Dataset* TeleQnA é de 733, fazendo apenas parte das categorias *standards specifications* e *standards overview*.

No conjunto de testes foram feitas 100 perguntas somente, sendo 50 delas da categoria *standards specifications* e 50 da categoria *standards overview*.

2.2 TSpec-LLM

O TSpec-LLM[9] é um *dataset* abrangente e *open-source* desenvolvido para auxiliar modelos de linguagem na compreensão de documentos técnicos de telecomunicações, especialmente os produzidos pelo *3rd Generation Partnership Project* (3GPP). Este conjunto de dados cobre todos os documentos das especificações do 3GPP, desde o *Release* 8 até o *Release* 19 (1999–2023), e preserva o conteúdo original, incluindo tabelas e fórmulas importantes para o contexto técnico. Composto por mais de 30.000 documentos e cerca de 535 milhões de palavras, o TSpec-LLM permite que LLMs (Modelos de Linguagem de Grande Escala) analisem e organizem informações complexas de telecomunicações de maneira eficaz, facilitando a interpretação de especificações de padrões. O TSpec-LLM é, portanto, uma ferramenta essencial para pesquisas na área de IA aplicada a telecomunicações, permitindo que os modelos LLM sejam usados como assistentes na leitura e análise de normas técnicas do setor.

O TSpec-LLM extrai os textos dos documentos da 3GPP e coloca em um formato padrão em Markdown[8], facilitando a manipulação das informações e extraindo mais facilmente os dados para serem usados em modelos de *Machine Learning*.

O *Dataset* TSpec-LLM será utilizado para auxiliar na busca de informações no framework de Geração Aumentada por Recuperação (RAG). Nesse contexto, o *dataset* servirá como a base de dados de documentos técnicos do 3GPP, permitindo que o sistema de recuperação localize e forneça trechos relevantes desses documentos para complementar as respostas geradas pelos modelos LLMs.

3 Metodologia

O projeto foi feito em python e pode ser acessado no repositório github em [6].

3.1 Modelos

Foram testados três modelos diferentes como forma de comparação em cima do *Dataset* TeleQnA, o modelo GPT-4o-mini da OpenAi, o Llama 3.2 de 3 bilhões de parâmetros da Meta utilizado a partir da biblioteca da Unsloth[13], e o mesmo Llama 3.2 de 3 bilhões de parâmetros porém com *Fine Tuning*[11] e treinado utilizando os dados do *Dataset* TeleQnA[10].

3.2 *Unsloth*

A *Unsloth*[13] é um repositório *Open Source* que disponibiliza diversos modelos LLM e possui uma estrutura definida que facilita carregar o modelo em uma GPU e então realizar o treinamento do modelo de forma rápida e prática.

3.3 *Fine Tuning*

O *Fine Tuning* do modelo Llama 3.2 de 3 bilhões de parâmetros seguiu a estrutura em [12].

3.3.1 *Dataset* de treinamento

Os dados usados no treinamento foi do *Dataset TeleQnA*[10], em que foram utilizadas 4000 perguntas. Das 4000 perguntas, 500 serão sobre a *Release* 17 (sobrando 233 para teste), e outras 3500 perguntas são sobre outras *Releases*, que não as *Releases* 17 e 18.

Para o treinamento, 2 diferentes pares de contexto e resposta esperada foram fornecidos.
Nas 2000 primeiras perguntas os pares seriam:

- Contexto: pergunta e opções
- Resposta Esperada: a opção correta e o texto da opção correta

Nas 2000 últimas perguntas os pares seriam:

- Contexto: somente a pergunta
- Resposta Esperada: somente o texto da opção correta

Por fim, foi feito um embaralhamento entre as perguntas.

3.3.2 Parâmetros e treinamento

O código de treinamento pode ser visto em [6], no caminho "*/Source/Fine_tuning/*". Como havia limitação de hardware pessoal (GPU local é uma NVIDIA GeForce RTX 3050 Ti. Memória máxima de 3,712 GB), o treinamento do modelo foi feito no Google Colab, com o objetivo de carregar o modelo sem quantização de 4 bits e treinar com mais perguntas em um tempo menor.

O modelo quantizado em 4 bits ocupa um espaço de memória na GPU de 2,768 GB, sem quantização de 4 bits o modelo carregado da Unsloth ocupa um espaço de 6,688 GB, em que o pico de memória ocupada no treinamento foi de 8,658 GB, para as 4000 perguntas.

Os parâmetros de treinamento foram:

- Número de parâmetros treináveis: 24.313.856
- Tamanho do *Batch* por dispositivo: 2
- Passos de acumulação de gradiente: 16
- Tamanho do *Batch*: 32
- Total de passos: 400
- Número de Épocas: 3,2
- Lr = 0,0002
- Otimizador: AdamW
- Perda inicial: 3,207
- Perda Final: 0,333

O treinamento durou cerca de 42 minutos, resultando no modelo que chamaremos de Llama 3.2 3B lora, pela adição de camadas Lora causada pelo *Fine Tuning*.

3.4 RAG

O RAG (*Retrieval-Augmented Generation*) é um sistema de recuperação que busca trechos relevantes em um conjunto de dados ou documentos específicos, oferecendo ao modelo de linguagem um contexto adicional, que aprimora a precisão e a relevância da resposta gerada. No caso deste trabalho, os documentos usados para recuperação de informação foram do *Dataset* TSpec-LLM[8].

Primeiramente os documentos foram divididos usando o "*MarkdownHeaderTextSplitter*", criando 780651 chunks de tamanho 2000 de caracteres cada um e sobreposição de 100 caracteres entre os chunks. O modelo de *Embeddings* usado foi o "*SentenceTransformer('all-mpnet-base-v2')*".

Em seguida, os *Embeddings* foram indexados usando o *Faiss*, gerando o arquivo *"faiss_index.bin"*, que posteriormente é usado para realizar a busca usando similaridade de cossenos. A entrada para busca no RAG foram as perguntas com opções para testes de acurácia, e somente as perguntas para testes das métricas RAGAS.

4 Experimentos

Os modelos GPT-4o-mini, Llama 3.2 3B e Llama 3.2 3B lora foram avaliados de duas formas diferentes usando 100 perguntas do *Dataset* TeleQnA referentes a *Release* 17, tal que no prompt da pergunta foi usado *Chain of Thought* pedindo para o modelo pensar passo a passo antes de responder.

Na primeira foi feita a pergunta e dada as opções, os modelos deveria apenas escolher qual seria a opção correta, e a métrica de avaliação foi a acurácia.

Na segunda foi feita a pergunta sem opções e os modelos deveriam dar uma resposta, em que o método de avaliação foram métricas do RAGAS (*Retrieval-Augmented Generation Assessment Suite*). Algumas métricas do RAGAS dependiam de avaliação de um LLM (GPT-4o-mini foi usado para realizar a avaliação), e outras não dependiam de um LLM.

- Métricas com LLM: *Factual Correctness, Semantic Similarity e Rubrics Score with reference*;
- Métricas sem LLM: *Bleu Score, Rouge Score, Exact Match e String Present.*

4.1 Acurácia

Os prompts usados nas perguntas com opções foram:

- Sem RAG:

```
f"Question: {question}\n"
f"Options:\n" + "\n".join(options) + "\n"
"Think step by step before answering and respond with the correct
option in the format 'correct option: <X>'."
```

- Com RAG:

```
f"Relevant Information:\n{rag_results}\n"
f"Question: {question}\n"
f"Options:\n" + "\n".join(options) + "\n"
"Think step by step and choose the correct option.\n"
"You must respond in the format 'correct option: <X>', where <X>
is the correct letter for the option."
```

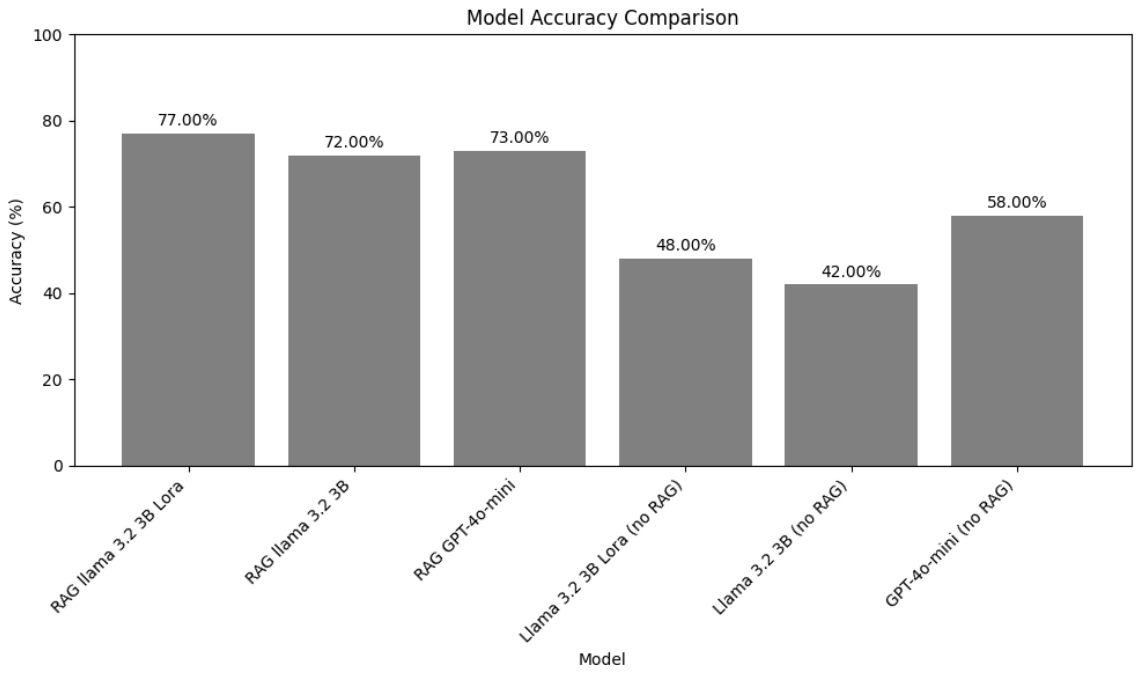

Figura 1: Avaliação das acurácias dos modelos fazendo 100 perguntas com opções

A figura 1 apresenta os resultados obtidos pelos modelos realizando as perguntas com opções e pedindo para o modelo selecionar a opção correta. Os valores dados em porcentagens representam o número de acertos de cada modelo.

4.2 RAGAS

Os prompts usados nas perguntas sem opções foram:

- Sem RAG:

  ```
  f"Question: {question}\n"
  "Think step by step before answering and respond with a final answer in the format 'answer:
  <XXXXX>'."
  ```

- Com RAG:

```
f"Relevant Information:\n{rag_results}\n"
f"Question: {question}\n"
"Think step by step before answering and analyze the relevant information carefully, then
respond with a final answer in the format 'answer: <XXXXX>'."
```

Models	**Factual Correctness Values (0 to 5)**	**Semantic Similarity Values (0 to 1)**	**Rubrics Score with reference Values (0 to 1)**
RAG llama 3.2 3B Lora	0,180	0,410	1,810
RAG llama 3.2 3B	0,141	0,415	2,370
RAG GPT-4o-mini	0,181	0,421	2,530
Llama 3.2 3B Lora (no RAG)	0,008	0,349	1,530
Llama 3.2 3B (no RAG)	0,078	0,500	1,950
GPT-4o-mini (no RAG)	0,061	0,362	2,240

Tabela 1: Resultados dos modelos em termos de Factual Correctness, Semantic Similarity e Rubrics Score com referência

A tabela 1 apresenta os resultados obtidos pelos modelos realizando as perguntas sem opções e avaliando com métricas do RAGAS que exigem um LLM, no caso o GPT-4o-mini.

O *Rubrics Score* com referência é uma métrica customizada em a pontuação é avaliada com valores de 1 a 5. A referência das métricas é dada ao LLM que irá avaliar da seguinte maneira:

```
rubrics = {
"score1_description": "The response is incorrect, irrelevant, or does not align with the
ground truth.",
"score2_description": "The response partially matches the ground truth but includes
significant errors, omissions, or irrelevant information.",
"score3_description": "The response generally aligns with the ground truth but may lack
detail, clarity, or have minor inaccuracies.",
"score4_description": "The response is mostly accurate and aligns well with the ground
truth, with only minor issues or missing details.",
"score5_description": "The response is fully accurate, aligns completely with the ground
truth, and is clear and detailed.",
}
```

Models	Bleu Score Values (0 to 1)	Rouge Score Values (0 to 1)	Exact Match Values (0 to 1)	String Present Values (0 to 1)
RAG llama 3.2 3B Lora	0,066	0,243	0,060	0,070
RAG llama 3.2 3B	0,058	0,200	0,050	0,050
RAG GPT-4o-mini	0,039	0,197	0,040	0,040
Llama 3.2 3B Lora (no RAG)	0,021	0,183	0,020	0,020
Llama 3.2 3B (no RAG)	0,056	0,174	0,000	0,000
GPT-4o-mini (no RAG)	0,003	0,139	0,000	0,020

Tabela 2: Resultados dos modelos em termos de Bleu Score, Rouge Score, Exact Match e String Present

A tabela 2 apresenta os resultados obtidos pelos modelos realizando as perguntas sem opções e avaliando com métricas do RAGAS que exigem não exigem um LLM para realizar a avaliação.

5 Conclusão

5.1 Acurácia

O treinamento do modelo demonstrou melhorias significativas no desempenho, tanto com RAG (Retriever-Augmented Generation) quanto sem. Com o uso de RAG, o modelo foi capaz de compreender melhor o contexto e fornecer respostas mais precisas, superando até mesmo o GPT-4o-mini e alcançando resultados impressionantes. Os testes realizados com RAG mostraram-se promissores, ultrapassando os resultados apresentados no artigo sobre TeleQnA[3], embora o conjunto de dados utilizado seja menor em comparação ao do estudo original.

Mesmo com um treinamento limitado, foi possível observar avanços no desempenho do modelo. Com um conjunto de dados maior, é possível que o modelo melhore ainda mais. Além disso, o uso de RAG mostrou que não é necessário um modelo excessivamente complexo para atingir resultados satisfatórios, pois essa abordagem sozinha já proporciona um desempenho muito bom.

5.2 RAGAS

As pontuações gerais foram baixas, mas o foco não estava em avaliar o desempenho absoluto dos modelos, uma vez que as métricas podem ser subjetivas, especialmente as que dependem de avaliações por LLM. O objetivo principal foi comparar os resultados dos modelos. Nesse sentido, os modelos com RAG se destacaram, apresentando resultados superiores.

No entanto, sem as alternativas, o GPT-4o-mini teve um desempenho melhor no geral em comparação aos outros modelos, especialmente na avaliação por LLM, com respostas mais próximas da correta. Já nas avaliações sem LLM, o Llama 3.2 3B com Fine-Tuning obteve melhores resultados, com respostas que se alinhavam mais à forma correta de escrever, refletindo o treinamento focado em palavras mais precisas e respostas mais objetivas.

É importante notar que, embora o Llama 3.2 lora tenha apresentado respostas mais objetivas, isso não necessariamente indicou um maior número de respostas corretas. Os outros modelos, apesar

de respostas mais variadas ou menos objetivas, também responderam questões de forma igualmente correta.

5.3 Conclusões Gerais

Com este trabalho é possível perceber que com o uso de RAG modelos pequenos como o Llama 3B podem apresentar desempenhos muito bons. Além disso, é importante destacar que no momento da inferência, nos testes das 100 perguntas, o modelo foi carregado com quantização de 4 Bits, de forma que o modelo não ocupasse um espaço de memória grande(seção: 3.3.2). Logo, mesmo com modelo pequeno e quantizado os resultados são parecidos com os do GPT-4o-mini, ambos utilizando RAG. O modelo com Fine Tuning ainda apresentou um resultado superior ao GPT-4o-mini para acurácia(seção: 4.1). Os resultados ainda foram melhores do que os apresentados no artigo [3], levando em conta que as 100 perguntas testadas eram somente das categorias *standards specifications* e *standards overview*, tendo em mente que o conjunto de testes de [3] eram bem maior.

Esses resultados são relevantes para demonstrar que aplicações que necessitem de conhecimento técnico ou específico não precisam de um modelo grande e caro para desempenhar tal tarefa. É possível ter um desempenho parecido ou superior com modelos menores, mais baratos e que podem rodar até localmente já que ocupam um espaço relativamente pequeno de memória, ainda mais porque a quantização do modelo não significou a queda de desempenho.

Vale ressaltar ainda que a inferência dos modelos menores não tinham um tempo de processamento muito maior que o GPT-4o-mini, embora não demonstrado neste trabalho, a inferência em cada pergunta demorava cerca de apenas alguns segundos para dar uma resposta. Nenhuma rodada de 100 perguntas levou mais do que 20 minutos.

6 Agentes

Foram feitos testes utilizando Agentes de modelos de LLM para avaliar o desempenho em comparação com os resultados obtidos com RAG. O Agente tinha uma ferramenta de busca que seria o mesmo RAG utilizando anteriormente.

Os modelos utilizados como Agentes foram o Llama 3.2 3B do Ollama e o Llama 3.1 70B disponibilizado pela empresa Venturus, sediada em Campinas-SP, em parceria com a matéria IA024 da Unicamp, ministrada pelo professor Roberto Lotufo em 2024.

Tentou-se utilizar a biblioteca ReAct da Langchain[2] para produzir o Agente, porém os modelos não se comportaram tão bem com algumas tentivas de prompts, especialmente o modelo menor Llama 3.2 3B.

Dessa forma foi feito uma cadeia de funções utilizando os modelos gerando um agente próprio para este trabalho específico, seguindo como base o código em [1]. O código do Agente pode ser visto em [6], no caminho "*Source/Agent/*". A ferramenta do Agente era o RAG que recuperava as 3 informações mais relevantes.

Os resultados utilizando Agentes estão na tabela 3:

Models	Acccuracy Values (0 a 100)	Number of None Responses Values (0 a 100)
Agent llama 3.2 3B	56%	5
Agent llama 3.1 70B Venturus	56%	7

Tabela 3: Resultados dos Agentes em termos de Acurácia e Quantidade questões sem resposta (None)

Os resultados são superiores aos testes sem RAG, porém piores do que utilizando apenas o RAG com recuperação dos 5 contextos mais relevantes. Como utilizou-se o mesmo RAG como ferramenta, esperava-se que os resultados fossem próximos aos da seção 4.1, e mesmo obrigando o Agente a executar pelo menos uma busca, ainda assim os modelos erraram mais do que o esperado. Acredita-se que talvez possa melhorar o prompt, ou a cadeia de ações do Agente possa melhorar e apresentar resultados melhores. A queda de desempenho não se deve a diminuição do número de informações relevantes dadas pelo RAG, uma vez que testes foram feitos com apenas 3 contextos ao invés de 5 e os resultados foram similares da seção 4.1.

7 Trabalhos Futuros

Os resultados apresentados neste trabalho são muito promissores para modelos pequenos, treinados ou não. Porém, com estudos mais abrangentes e novas soluções que surgem a cada dia é esperado que o desempenho de modelos futuros sejam ainda melhores.

Para trabalhos futuros, seria interessante aprofundar a pesquisa sobre o uso de agentes para responder perguntas, buscando aprimorar a qualidade da pesquisa e do processo de resposta passo a passo. Além disso, melhorias no uso do RAG poderiam ser exploradas, como a criação de *chunks* menores, a implementação de *embeddings* mais eficientes ou a aplicação de um novo pré-processamento para eliminar informações irrelevantes dos documentos.

Com uma infraestrutura de hardware mais robusta, seria viável realizar o treinamento com um *dataset* maior, além de explorar a comparação entre modelos de diferentes tamanhos, observando os impactos do treinamento em cada configuração.

Outra possibilidade é testar o desempenho da pesquisa em documentos locais e na internet, avaliando como cada abordagem impacta a qualidade das respostas. Essa abordagem é semelhante a utilizada no Telco-RAG em [4][5]. Além disso, é possível realizar uma comparação entre os resultados obtidos nos modelos deste trabalho e nos modelos do Telco-RAG.

Referências

[1] How to create your own llm agent from scratch: A step-by-step guide. https://medium.com/@incle/how-to-create-your-own-llm-agent-from-scratch-a-step-by-step-guide-14b763e5b3b8. Accessed: Nov 2024.

[2] React. https://python.langchain.com/v0.1/docs/modules/agents/agent$_{t}ypes/react/.Accessed$: $Nov2024$.

[3] Nicola Piovesan Antonio De Domenico Merouane Debbah Zhi-Quan Luo Ali Maatouk, Fadhel Ayed. Teleqna: A benchmark dataset to assess large language models telecommunications knowledge. *arXiv preprint arXiv:2310.15051*, 2023. Accessed: Nov 2024.

[4] Antonio De Domenico Nicola Piovesan Ali Maatouk Andrei-Laurentiu Bornea, Fadhel Ayed. Telco-rag: Navigating the challenges of retrieval augmented language models for telecommunications. *arXiv preprint arXiv:2404.15939*, 2024. Accessed: Nov 2024.

[5] Mckay Wrigley Andrei-Laurentiu Bornea. Telco-rag repository. https://github.com/netop-team/Telco-RAG. Accessed: Nov 2024.

[6] José Arimatea. 3gpp llm evaluation. https://github.com/josearimatea/3gpp_llm_evaluation. Accessed: Nov 2024.

[7] Alexandros Nikou Farnaz Moradi Christian Olrog Fitsum Gaim-Henrik Holm Doumitrou Daniil Nimara Vincent Huang Athanasios Karapantelakis, Mukesh Thakur. Using large language models to understand telecom standards. *arXiv preprint arXiv:2404.02929*, 2024. Accessed: Nov 2024.

[8] Rasoul Nikbakht. Tspec-llm repository. https://huggingface.co/datasets/rasoul-nikbakht/TSpec-LLM. Accessed: Nov 2024.

[9] Mohamed Benzaghta Rasoul Nikbakht and Giovanni Geraci. Tspec-llm: An open-source dataset for llm understanding of 3gpp specifications. *arXiv preprint arXiv:2406.01768*, 2024. Accessed: Nov 2024.

[10] Netop Team. Dataset teleqna. https://huggingface.co/datasets/netop/TeleQnA. Accessed: Nov 2024.

[11] UnsLoth. How to finetune llama-3 and export to ollama. https://docs.unsloth.ai/tutorials/how-to-finetune-llama-3-and-export-to-ollama. Accessed: Nov 2024.

[12] Unsloth. Llama 3.2 fine tuning. https://colab.research.google.com/drive/1T5-zKWM_5OD21QHwXHiV9ixTRR7k3iB9?usp=sharing. Accessed: Nov 2024.

[13] Unsloth. Unsloth repository. https://github.com/unslothai/unsloth?tab=readme-ov-file. Accessed: Nov 2024.