UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Gabriel Araújo Miranda Santos

Relatório de estágio supervisionado no Núcleo de pesquisa e desenvolvimento e inovação VIRTUS - UFCG

Campina Grande - PB

Outubro de 2024

Gabriel Araújo Miranda Santos

Relatório de estágio supervisionado no Núcleo de Pesquisa e Desenvolvimento VIRTUS - UFCG

Relatório de Estágio Supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Elétrica.

Orientador:

Prof. Dr. Gutemberg Gonçalves dos Santos Júnior

Universidade Federal de Campina Grande - UFCG

Departamento de Engenharia Elétrica - DEE

Campina Grande - PB

Outubro de 2024

Dedico este trabalho a Deus, autor da minha vida, e a família que Ele mesmo me deu. A minha esposa, a quem dedicar minha vida é pouco. A meus pais, que se dedicaram em tornar esse momento possível.

Agradecimentos

Em primeiro lugar, agradeço a Deus Pai pelo amor provado em Jesus que me trouxe até aqui. A Ele por quem todas as coisas vieram a existir e que antes do tempo existir escreveu este momento.

Agradeço a minha esposa, Ana Raquel, por todo carinho e serviço a mim conferidos. Por me acompanhar em toda trajetória desse estágio, orando por mim e celebrando comigo. Por me incentivar a dar o meu melhor, sendo ela mesma a minha referência para isso.

Agradeço a meus pais, Antônio Marcos e Maria do Carmo, pelo esforço imensurável que tornaram esse momento possível. Fizeram por mim o que não puderam fazer por eles. Texto algum será suficiente para expressar o quanto sou grato a vocês.

Ao professor Gutemberg, que desde o início da minha graduação prontamente me atendeu, seja como coordenador ou professor. Tê-lo como orientador é mais uma prova disso. Agradeço-lhe pelo apoio prestado neste trabalho, mas também por todo serviço e acolhimento durante esses anos na universidade.

Ao meu gerente, Paulo de Tarso, que durante todo o período de estágio sempre esteve disponível para ajudar, sendo um exemplo de profissional no desempenho do seu trabalho. Também aos meus companheiros de time Pablo Victor e Jordan Nunes por toda prestatividade e apoio no trabalho.

Aos meus amigos, aqueles que tive o prazer de fazer pelo caminho deste estágio, Ana Flávia, Alexsandra Souto, Jamilson Silva, Larissa Teixeira, Mayra Silva, Lourival Netto, Antônio Lúcio. Ter a vossa companhia abrilhantou esse período.

Além destes, agradeço a todos aqueles que, de alguma forma, colaboraram comigo. A multiforme graça de Deus atua através de inúmeros meios e pessoas. Sou grato a cada uma delas.

"pois nele vivemos, e nos movemos, e existimos" - Atos 17: 28

Resumo

Este relatório apresenta as atividades que foram desenvolvidas pelo aluno Gabriel Araújo Miranda Santos durante o período de estágio supervisionado no Núcleo VIRTUS - UFCG. No decorrer deste documento será discorrido acerca da introdução do discente a uma nova área tecnológica, as metodologias aplicadas durante o processo de estágio, a execução de testes de modelos de *Large Language Models* desenvolvidos no projeto e a documentação dos resultados desses testes.

Palavras-chave: Estágio supervisionado, *Large Language Models*, validação, métricas.

Abstract

This report presents the activities developed by the student Gabriel Araújo Miranda Santos during the supervised internship period at the VIRTUS Center - UFCG. Throughout this document, the student's introduction to a new technological area will be discussed, along with the methodologies applied during the internship process, the execution of tests on Large Language Models developed in the project and the documentation of the results from these tests.

Keywords: Supervised internship, Large Language Models, Validation, Metrics.

Lista de Figuras

1.1 Fotografia do VIRTUS UFCG 2

2.1 Fluxo de projeto seguindo a metodologia Scrum 12

Sumário

Agradecimentos **ii**

Resumo **iv**

Abstract **v**

Lista de Figuras **vi**

1 Introdução **1**

1.1 VIRTUS 1

1.2 Objetivos 2

1.3 Estrutura do relatório 2

2 Fundamentação Teórica **4**

2.1 IA Generativa 4

2.1.1 Definições 4

2.2 *Large Language Models* 6

2.2.1 Engenharia de *Prompts* 6

2.3 *Retrieval Augmented Generation* 7

2.3.1 Conceitos fundamentais 7

2.3.2 Avaliação de LLMs 8

2.4 Metodologia Scrum 11

3 Atividades desenvolvidas **13**

3.1 Metodologia de trabalho 13

3.2 Contextualização ao tópico do projeto 14

3.3 Elaboração de base de dados 18

3.4 Validação e documentação 18

4 Considerações finais **20**

Capítulo 1

Introdução

Este trabalho faz referência ao estágio curricular do aluno do curso de Engenharia Elétrica da Universidade Federal de Campina Grande (UFCG), Gabriel Araújo Miranda Santos no Núcleo de Pesquisa, Desenvolvimento e Inovação em Tecnologia da Informação, Comunicação e Automação - VIRTUS. O período do estágio foi de 25 de junho de 2024 a 03 de outubro de 2024, totalizando uma carga horária de 331 horas. Durante o período de estágio o aluno participou da execução de testes para modelos de linguagem baseados em *deep learning*, atuou no processo de testes, análise de resultados e documentação destes.

1.1 VIRTUS

O VIRTUS é um órgão suplementar da Universidade Federal de Campina Grande e é vinculado ao Centro de Engenharia Elétrica e Informática [3]. Conta com diversas áreas de atuação como, por exemplo, Inteligência Artificial, *Machine learning*, microeletrônica, Internet das Coisas, computação em nuvem, sistemas embarcados, automação industrial, entre outros. Por ser parte de um Instituto de Ciência e Tecnologia, o VIRTUS executa projetos de Lei de Informática, EMBRAPII, ANP, e ainda outros mecanismos de incentivo a pesquisa e inovação.

Atualmente, o VIRTUS é credenciado como Centro de Competência EMBRAPII em *hardware* inteligente para a indústria, cujo objetivo é desenvolver competências para atender a desafios tecnológicos empresariais. Conta com a participação de diversos colaboradores, sendo estes professores da UFCG, estudantes de graduação e pesquisadores.

Figura 1.1: Fotografia do VIRTUS UFCG

Fonte: Disponível em: https://embrapii.org.br

1.2 Objetivos

O objetivo deste trabalho é apresentar as atividades desenvolvidas durante o período de estágio no VIRTUS. As atividades são:

- contextualização a respeito da temática de LLMs;
- elaboração de bases de dados para testes;
- validação de modelos a partir de métricas disponíveis na literatura;
- participação de cerimônias e atividades segundo a metodologia Scrum.

1.3 Estrutura do relatório

Este relatório segue a seguinte organização, seguindo a numeração de cada capítulo:

1. **Introdução** acerca do presente trabalho e do local do estágio;
2. **Fundamentação teórica** a respeito dos temas pertinentes para a compreensão das atividades relatadas;

3. **Atividades desenvolvidas** durante o período de estágio e as metodologias utilizadas;

4. **Considerações finais** acerca do estágio e conclusões do trabalho.

Capítulo 2

Fundamentação Teórica

Neste capítulo serão apresentados conceitos e definições importantes para a compreensão das atividades desenvolvidas no decorrer do estágio. Inicialmente será apresentado acerca do ramo de inteligência artificial generativa e, em seguida, um dos modelos dessa categoria de IA (Inteligência Artificial), o *Large Language Model*, por fim, uma apresentação da metodologia Scrum.

2.1 IA Generativa

Uma inteligência artificial generativa pode ser definida como uma categoria de tecnologia de IA que é capaz de produzir diversos tipos de conteúdo, incluindo texto, imagens, áudios e dados sintéticos [5]. Contudo, por fazer parte de um conjunto dentro do ramo da inteligência artificial, algumas definições anteriores são importantes de serem abordadas.

2.1.1 Definições

Inteligência artificial, ou IA, é uma tecnologia que permite que computadores e máquinas simulem a capacidade de resolução de problemas semelhante a inteligência humana [6]. O aprendizado de máquina é um subcampo dela e este pode ser construído sobre Aprendizado Supervisionado ou Aprendizado Não Supervisionado, caracterizados pela forma como os dados são configurados, rotulados ou não.

Deep Learning (DL) é um campo mais restrito do *Machine Learning* (ML) que tenta simular os neurônios humanos com conexões artificiais com muitas camadas de

neurônios que recebem dados e tentam fazer previsões [6]. O *Deep Learning* pode usar dados rotulados ou não rotulados, sendo chamado de Aprendizado Semi-Supervisionado. Um modelo dessa categoria pode receber algumas entradas rotuladas para treinamento e aprendizado de conceitos básicos, e muitas entradas não rotuladas para ser capaz de generalizar para novos exemplos.

IA Generativa é um subconjunto do *Deep Learning* que utiliza redes neurais artificiais, podendo processar dados rotulados ou não rotulados usando aprendizado supervisionado, não supervisionado ou semi-supervisionado. Os *Large Language Models*, comumente referenciado pela sigla LLM, também são um subconjunto do DL e serão apresentados de maneira mais detalhada na seção2.2.

Modelos de linguagem podem ser divididos em dois tipos: modelos **generativos** ou **discriminativos** [2].

Modelo generativo	**Modelo discriminativo**
São usados para criar novos dados com base na distribuição de probabilidade dos dados existentes.	São usados para classificar ou prever rótulos para dados.
Podem criar textos, imagens, vídeos, áudio, etc., a partir das informações dentro do conjunto de dados.	Comumente treinados com dados rotulados para aprender a relação entre dados e rótulos.

Tabela 2.1: Comparação entre modelos generativos e discriminativos

Modelos de linguagem generativos são sistemas de correspondência de padrões que consistem em encontrar no banco de dados as informações mais relacionadas à entrada e prever qual deveria ser a saída lógica.

Modelos desse tipo são possíveis graças ao uso de *Transformers*, tecnologia responsável pela revolução do *Natural Language Processing* em 2018. Um modelo de *Transformer* consiste de um *encoder* e um *decoder*. O *encoder* codifica a sequência de entrada e a passa para o *decoder*, que aprende a decodificar e usar os dados para tarefas relevantes.

Outro conceito importante são as **alucinações**, que são saídas que não possuem sentido [7]ou são gramaticalmente incorretas. Ocorrem quando um modelo é treinado com dados insuficientes, ruidosos ou sujos, ou não recebe contexto ou restrições suficientes. Alucinações tornam o texto de saída difícil de entender e tornam o modelo mais propenso a gerar informações incorretas ou enganosas.

Existem, ainda, os chamados **modelos de fundação** (ou *foundation models*), que são grandes modelos de IA pré treinados em uma vasta quantidade de dados e são definidos

para serem adaptáveis para uma gama de tarefas.

2.2 *Large Language Models*

LLMs se referem a grandes modelos de linguagem de propósito geral que podem ser pré-treinados e depois ajustados para propósitos específicos [2]. Modelos dessa classe são caracterizados pelos seguintes critérios:

Grandes: este critério pode se referir ao tamanho do conjunto de dados de treinamento ou ao grande número de parâmetros.

Propósito Geral: o modelo é suficiente para resolver problemas comuns, como tarefas relacionadas à linguagem humana.

Pré-treinados e ajustados: Dada a magnitude do processamento necessário para construir modelos LLMs, construir modelos dessa categoria se torna custoso. Portanto, algumas empresas realizam esse trabalho e criam modelos de linguagem para que outros possam utilizá-los para tarefas específicas por meio de ajuste fino, o que requer um conjunto de dados menor.

Prompt Design e *Prompt Engineering* são dois conceitos intimamente relacionados, ambos envolvem o processo de criar uma entrada textual, que serve como estímulo inicial para a geração de um texto ou sequência de caracteres por parte de um modelo de linguagem, chamada de *prompt*. O *prompt* deve ser claro, conciso e informativo.

Prompt Design é o processo de criar um *prompt* adaptado à tarefa específica que o sistema está sendo solicitado a realizar.

Prompt Engineering é o processo de projetar um *prompt* para melhorar o desempenho, o que pode exigir conhecimento específico do domínio. Acerca disso será discorrido na próxima seção.

2.2.1 Engenharia de *Prompts*

A natureza e a complexidade do *prompt* influenciam diretamente a qualidade e a relevância da saída gerada pelo modelo.

Engenharia de *prompts* é a técnica de se comunicar com um LLM de forma que sua resposta seja útil para quem interage com ele [2]. Uma maneira comum de ajudar a IA a fornecer informações úteis é informar o contexto no *prompt*. Isso pode ser feito por:

- fornecendo exemplos da tarefa desejada. Isso é conhecido como aprendizado em contexto;
- especificando como formatar as respostas;
- pedindo ao modelo para agir como uma persona. Por exemplo, para aprender sobre o processo legal, peça ao LLM para assumir a posição de um advogado ao responder;
- incluindo dados adicionais ou específicos para ajudar a aumentar a particularidade da resposta.

2.3 *Retrieval Augmented Generation*

RAG é uma estrutura de IA para recuperar fatos de um banco de dados externo a fim de fundamentar LLMs nas informações mais precisas e atualizadas, e fornecer aos usuários *insights* sobre o processo generativo [8].

É importante saber que os modelos LLM não compreendem o significado das palavras que soletram, mas apenas as relacionam estatisticamente com base nos vetores das palavras. O RAG atua para melhorar a qualidade da resposta gerada com fontes externas de conhecimento, com fatos mais atuais e confiáveis.

Ao permitir que o modelo tenha acesso a essas fontes de informação, a chance de uma resposta ruim diminui em comparação com a resposta gerada apenas a partir do que está nos parâmetros do modelo. Isso reduz as alucinações ou vazamentos de dados. Além disso, o RAG também reduz a necessidade do usuário de treinar o modelo com novos dados atualizados, ajudando assim a reduzir os custos computacionais e financeiros de execução do modelo.

2.3.1 Conceitos fundamentais

Dentro do escopo de um RAG, alguns processos são realizados, os quais envolvem:

Orquestradores: Orquestração é um método para gerenciar e coordenar LLMs a fim de garantir a integração com sistemas. Permite que LLMs executem tarefas complexas, aprendam e melhorem continuamente.

Recuperadores: Recuperadores são uma interface que retorna documentos dado uma consulta não estruturada. Eles não armazenam os documentos, apenas os recuperam.

Os recuperadores aceitam uma consulta de *string* como entrada e retornam uma lista de documentos como saída.

Memória: É a capacidade de manter as mensagens anteriores em um bate-papo de conversação.

2.3.2 Avaliação de LLMs

Como em outros contextos de ML, é necessário verificar a qualidade da resposta do modelo LLM, mas as métricas usadas nesses outros contextos não são aplicadas aqui. Devido a isso, novas métricas são usadas para provar as respostas. As métricas aqui apresentadas se baseiam na estrutura proposta pela biblioteca RAGAs[9], que provê ferramentas úteis para a avaliação de LLMs.

A avaliação de um modelo ajustado finamente, ou RAG, envolve uma comparação entre o desempenho do modelo e um banco de dados de *ground truth*, se disponível, para comparação. No *ground truth* estão os resultados ideais de resposta, que servem como parâmetro para visualizar o quanto as respostas reais do modelo distam do alvo. A partir desses recursos, torna-se dever do programador garantir que o modelo responda como esperado.

Answer Relevance

Esta métrica avalia quão pertinente a resposta é ao *prompt*. Pontuações mais baixas referem-se a respostas incompletas ou redundantes, enquanto pontuações mais altas indicam uma melhor relevância. Esta métrica usa a pergunta, o contexto e a resposta para calcular a pontuação.

Ela é definida matematicamente como a média da similaridade de cosseno da pergunta original a um número de perguntas artificiais que foram geradas com base na resposta. Cada pergunta textual é movida para um plano de vetores definidos numericamente, isso faz com que cada expressão seja expressa em números, tornando possível a visualização da distância de cada vetor no espaço. Matematicamente é expressa como:

$$answer\ relevance = \frac{1}{N}\Sigma_{i=1}^{N} cos(E_{gi}, E_0) \tag{2.1}$$

ou também, como:

$$answer\ relevance = \frac{1}{N}\Sigma_{i=1}^{N}\frac{E_{gi}.E_0}{||E_{gi}||.||E_0||} \tag{2.2}$$

Onde: E_i é a incorporação da i-ésima pergunta gerada. E_0 é a incorporação da pergunta original. N é o número de perguntas geradas, com padrão de 3.

Answer Correctness

Esta métrica também compara a verdade fundamental e a resposta, mas também considerando as informações factuais nelas sobre a similaridade semântica com pesos diferentes em cada categoria de similaridade. O resultado varia entre 0 e 1.

É usada a fórmula F1-score, equação (2.3), para quantificar a correção factual com base no número de afirmações em cada uma das listas, listas que são:

- Verdadeiro Positivo (TP): Afirmações que estão tanto no *ground truth* quanto na resposta gerada.
- Falso Positivo (FP): Afirmações que estão na resposta, mas não no *ground truth*.
- Falso Negativo (FN): Afirmações que estão no *ground truth*, mas não na resposta.

$$F1_{score} = \frac{|TP|}{(|TP| + 0.5.(|FP| + |FN|))} \tag{2.3}$$

Depois disso, a similaridade semântica é calculada e uma média ponderada é obtida a partir dessas duas pontuações para obter a pontuação final. O peso do cálculo pode ser modificado.

Answer Similarity

Esta métrica é baseada no *ground truth* e na resposta, comparando a semântica entre elas com valores variando entre 0 e 1. Os passos para avaliar com base nesta métrica são:

1. Vetorizar a resposta de verdade fundamental usando um modelo de *embbeddings*.
2. Vetorizar a resposta gerada usando o mesmo modelo de *embbeddings*.
3. Calcular a similaridade de cosseno entre os dois vetores.

Context Precision

Avalia se todos os itens relevantes do *ground truth* presentes no contexto estão em alta classificação ou não. O cenário ideal é que todos os trechos relevantes apareçam no topo. Esta métrica usa pergunta, contexto e *ground truth* para ser avaliada, variando entre 0 e 1.

$$Context\ Precision_k = \frac{\Sigma_{k=1}^{K}(Precision_k \times v_k)}{\text{Total de itens relevantes nos K melhores resultados}} \tag{2.4}$$

$$Precision_k = \frac{true\ positives_k}{(true\ positives_k + false\ positives_k)} \tag{2.5}$$

Onde K é o número total de trechos nos contextos e $v_k \in 0,1$ é o indicador da relevância na posição k.

Context Recall

Mede a extensão em que o contexto recuperado se alinha com a resposta, sendo esta tratada como o *ground truth*. Utiliza o *ground truth* e o contexto recuperado para a avaliação e varia entre 0 e 1.

Cada frase na resposta de verdade fundamental é analisada para verificar se pode ser associada ao contexto recuperado ou não. O cenário ideal é que todas as frases na resposta sejam associadas ao contexto recuperado.

$$context\ recall = \frac{|\text{frases do GT que podem ser inferidas pelo contexto}|}{|\text{Número de frases no GT}|} \tag{2.6}$$

Faithfulness

Esta métrica mede a consistência da resposta gerada em relação ao contexto dado e é calculada a partir da resposta e do contexto recuperado em uma escala de 0 a 1. Uma resposta fiel indica que todas as afirmações podem ser inferidas do contexto. A pontuação é calculada por uma divisão do número de afirmações na resposta que podem ser inferidas do contexto sobre o número total de afirmações na resposta.

2.4 Metodologia Scrum

O Scrum é uma estrutura de trabalho que ajuda pessoas, times e organizações a gerar valor através de soluções adaptativas para problemas complexos [1]. Consiste em três fases:

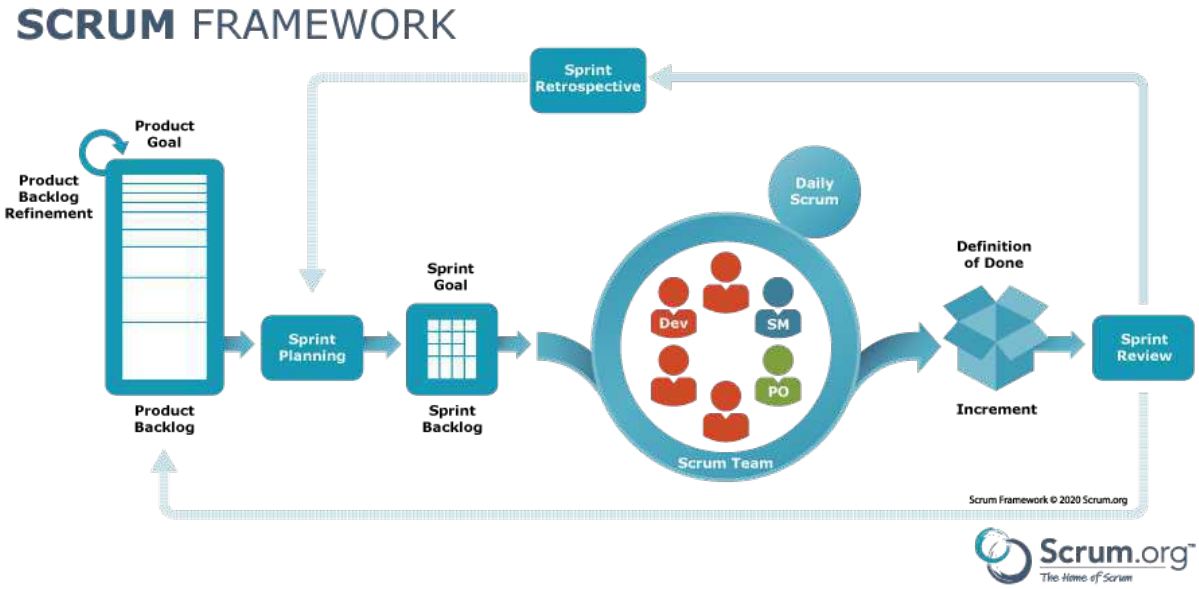
- fase inicial de planejamento, na qual são definidos os objetivos;
- desenvolvimento do trabalho, onde o sistema produzido é incrementado;
- fechamento do projeto, quando este é completado e aceitado pelo cliente.

Dentro dessa metodologia, alguns termos são importantes para se estabelecer uma comunicação clara, provendo também um entendimento de como a estrutura se organiza. O **time de desenvolvimento** é a equipe responsável por produzir um sistema, independentemente do contexto, solicitado por um cliente. As tarefas desse time são organizadas em um ***product backlog***, nele estão contidas todos os itens que devem ser implementados no produto final. O ***product owner*** é o responsável pelo *product backlog* e representa o cliente dentro da equipe. O ***scrum master*** tem como papel gerenciar e manter o processo Scrum funcionando, promovendo reuniões e buscando resolver impedimentos.

Em um projeto que utiliza dessa metodologia, são realizadas diversas reuniões (chamadas cerimônias), dentre elas a ***daily***, a qual é o encontro diário em que cada um do time de desenvolvedores apresenta o que foi feito, o que irá ser feito e se houveram problemas na execução das tarefas.

Todo esse fluxo de projeto acontece em períodos denominados de ***sprints***, que são ciclos de 2 ou 4 semanas, dentro das quais são executados itens escolhidos do *product backlog*. O fluxograma pode ser visualizado na figura 2.1:

Figura 2.1: Fluxo de projeto seguindo a metodologia Scrum



Fonte: Extraído de (SCHWABER, 2020)

Capítulo 3

Atividades desenvolvidas

Neste capítulo serão apresentadas, de maneira que respeite a confidencialidade do projeto, as atividades desenvolvidas. Desde aquelas que se referem ao fluxo de organização da equipe como as de cunho técnico de desenvolvimento.

3.1 Metodologia de trabalho

No desenvolvimento do projeto, foi adotado a metodologia Scrum para gestão de uma equipe composta por três desenvolvedores e um gerente de projetos. As etapas do projeto eram divididas por *sprints* de 15 dias, composta por cerimônias, dentre as quais, a reunião de planejamento da *sprint* era feita quinzenalmente no dia de seu início, na qual eram discutidas todas as atividades a serem desenvolvidas. Além dela, também haviam reuniões diárias para atualização de *status* de cada integrante da equipe.

Foi utilizado o quadro Kanban, por meio de uma ferramenta interna online de gestão de projetos, para acompanhamento das tarefas, permitindo que cada colaborador pudesse acompanhar as atividades dos demais. Essa técnica contribuiu para a comunicação entre a equipe, tornando-a mais clara e assertiva. Foram planejadas três áreas de atuação principais, nas quais as atividades do estágio seriam definidas: contextualização acerca da temática do projeto, criação de uma base de dados para treinamento do modelo e a avaliação do modelo desenvolvido. Estas atividades serão descritas a seguir.

3.2 Contextualização ao tópico do projeto

Como primeira atividade, foi necessário um período de familiaridade do discente com o contexto a ser trabalhado. Portanto, As primeiras *sprints* do projeto foram destinadas a esse propósito. Foram realizadas investigações teóricas acerca da construção de uma base de dados cujas informações são representações matemáticas chamados de vetores [10], RAG e RAG avançada, inferência e otimização de modelos, segurança em modelos de LLM e uma investigação mais prática acerca de engenharia de *prompt*[1].

Para o tópico de engenharia de *prompt*, foi feito um estudo sobre as técnicas utilizadas nessa área, tomando como base para os estudos o modelo *open source* da empresa Meta, o Llama. Esse estudo foi orientado pelo curso introdutório sobre engenharia de *prompt* [4] que aborda os conceitos mais iniciais como a importação e a interação com um modelo, até mecanismos para otimizar a comunicação visando receber melhores respostas do modelo. A ferramenta utilizada para desenvolvimento dos estudos foi o Jupyter Notebook[2].

A estrutura para utilização de um modelo Llama consiste na importação da biblioteca que o compreende, a definição de um *prompt* e a execução do modelo que retorna uma resposta. Um exemplo de código que faz isso é apresentado abaixo:

```
# Importando do modelo
from utils import llama

# Exemplo de pergunta a ser feita para o modelo
prompt = "Quando foi registrada a primeira fotografia de um buraco negro?"

# Chamada do modelo recebendo o prompt como argumento
resposta = llama(prompt)

# Apresenta a resposta que o modelo gerou para o referido prompt
print(resposta)
```

A construção de um *prompt* segue um padrão a respeito dos componentes que o

[1] O curso ofertado pela DeepLearning.AI é encontrado gratuitamente no link: https://www.deeplearning.ai/short-courses/prompt-engineering-with-llama-2/

[2] Link para a ferramenta: https://jupyter.org/

formam. Aquilo que é solicitado ao modelo, antes, vem acompanhado de uma instrução que é indicada no texto fornecido com um `[INST]` no início e um `[/INST]` ao final. Na maioria das vezes, esse sinalizador não é explicitamente indicado, contudo o modelo é capaz de identificar o trecho de instrução mesmo assim.

Além do modo como a instrução é dada para o modelo, outras abordagens que visam otimizar a performance do modelo são as configurações de argumentos, como por exemplo o argumento de `max_tokens` e o argumento `temperature` de um modelo. O primeiro se refere ao número máximo de *tokens* (símbolos) que são suportados em um *prompt*. O segundo corresponde ao nível técnico da resposta de um modelo, sendo índices baixos de temperatura resultando em respostas mais lógicas e diretas, enquanto valores maiores de temperatura permitem respostas mais criativas. O trecho a seguir apresenta um exemplo de definição desses argumentos.

```
from utils import llama

# Exemplo de prompt cuja resposta nao seja objetiva
prompt = "Qual o sentido da vida?"

# Resposta com temperatura baixa
resposta_menor_temperatura = llama(prompt,
                                   max_tokens= 150,
                                   temperature= 0.0)

# Resposta com uma temperatura que torne o modelo mais criativo
resposta_maior_temperatura = llama(prompt,
                                   max_tokens= 150,
                                   temperature= 0.9)
```

Um fator importante de ser considerado na comunicação com um LLM é que os modelos, por padrão, são *stateless*, ou seja, não armazenam o histórico de uma interação. Portanto, para conversações que dependem de dados vistos em momentos prévios na interação, uma técnica utilizada é a de reenviar para o modelo no próximo *prompt* a estrutura da conversa anterior, seguindo a estrutura:

```
prompt_chat = f"""
```

```
  Usuario: {prompt_1}
  Assistente: {resposta_1}
  Usuario: {prompt_2}
  Assistente: {resposta_2}
  Usuario: {prompt_atual}
"""
```

A partir desse conhecimento inicial, foram investigadas técnicas de engenharia de *prompt*, dentre as quais destacam-se:

Zero-shot prompting

Consiste em construir o *prompt* apenas com a instrução para o modelo, sem fornecer-lhe nenhuma informação adicional. Isto faz com que o modelo apresente resultados unicamente baseados naquilo que faz parte da sua base de dados.

Few-shot prompting

Esta técnica incrementa o *zero-shot prompting* na medida em que fornece alguns exemplos do que se espera do modelo. Pode ser um ou mais exemplos de como a resposta deve ser formatada. Em uma tarefa de classificação de sentimento de uma mensagem, um exemplo seria:

```
prompt = """
Sua tarefa consiste em classificar sentimentos de mensagens. Alguns
    exemplos:
Mensagem: Amei a refeicao!
Sentimento: Positivo

Mensagem: O onibus atrasou e cheguei atrasado ao trabalho.
Sentimento: Negativo

Classifique a seguinte mensagem:
Mensagem: A reuniao de ontem foi cansativa.
Sentimento:
"""
```

Especificação de saída

Alguns modelos podem ser prolixos em suas respostas. Expressar no *prompt* a maneira precisa de como a resposta deve ser dada é uma técnica que contribui com o bom desempenho do modelo.

Concluída a etapa de investigações e nivelamento de conhecimento acerca das tecnologias que envolvem o trabalho com LLMs, foi dado início as atividades de testes do modelo. As atividades que compreendem esta etapa são apresentadas nas seções seguintes.

3.3 Elaboração de base de dados

Antes de iniciar a etapa de avaliação, é necessário gerar respostas pelo modelo a partir de um referencial confiável. Esse referencial é o *ground truth*, que consiste em uma base de dados na qual contém respostas validadas que seriam o ideal a ser obtido pelo LLM. Para a realização dessa atividade, foi utilizado o Jupyter Notebook[3], onde foram feitas as configurações do modelo e a definição do RAG responsável pela geração de contextos.

Configurado o ambiente, os documentos recuperados pelo RAG deveriam ser divididos para serem armazenados em uma nova base de dados na qual deveriam haver as seguintes informações:

- **Questão**: uma pergunta de acordo com o cenário específico em que o modelo seria utilizado;
- **Resposta ideal**: a resposta que havia sido previamente validada e tida como ideal para a pergunta feita;
- **Contexto**: conteúdos recuperados pelo RAG que são relevantes para a formulação da resposta do modelo;
- **Resposta do LLM**: resposta do modelo, gerada a partir da questão e do contexto.

Após a criação da base de dados, iniciou-se a etapa de testes, a qual está descrita na próxima seção.

3.4 Validação e documentação

Para aprovação do modelo, ele deve atingir resultados que mostrem-no como confiável para utilização. Para tanto, tendo em vista que algumas métricas necessitam da resposta

[3] *Framework* para desenvolvimento em Python

gerada pelo LLM, o contexto fornecido pelo RAG, e a pergunta da respectiva resposta, foi criado uma base de dados que contém todas essas informações.

As métricas apresentadas na seção 2.3.2 foram implementadas em código Python (essa implementação foi feita por outro colaborador) e utilizadas para avaliar o modelo a partir das respostas obtidas e já armazenadas na base de dados. Vários cenários foram avaliados nos testes.

Observando os resultados obtidos, mudanças foram implementadas no modelo e nos *prompts*, e em cada uma dessas novas alterações eram realizados novos testes, que compreendiam o processo de nova criação da base de dados e nova execução de cada métrica. Ao final de cada série de testes, os resultados eram documentados, contendo as informações de como foi a performance do modelo em cada cenário e em cada métrica, sendo utilizado para consulta e norte para posteriores otimizações no LLM.

Capítulo 4

Considerações finais

Durante o estágio no VIRTUS, a oportunidade de trabalhar com a validação de modelos de *Large Language Models*, uma tecnologia emergente e de grande relevância para o setor de inteligência artificial, foi muito enriquecedora. Apesar deste tópico não integrar a grade do curso de Engenharia Elétrica, as habilidades apreendidas durante a graduação foram fundamentais para me adaptar rapidamente às demandas durante o estágio. A experiência foi enriquecida pela utilização da metodologia Scrum, metodologia estudada no componente curricular de Informática Industrial, que proporcionou uma base sólida para o trabalho em equipe e a entrega de resultados incrementais.

O estágio também destacou a importância do aprendizado contínuo e da capacidade de adaptação a novas tecnologias. Essa experiência prova a necessidade de uma base acadêmica sólida, mas também sobre a necessidade de flexibilidade para aprender e aplicar novos conhecimentos de forma autônoma, sendo esse um aspecto fundamental para o profissional atual.

Referências Bibliográficas

[1] SCHWABER, Ken; SUTHERLAND, Jeff. **The Scrum Guide: The definitive guide to Scrum: The rules of the game**. 2020. Disponível em: https://scrumguides.org/scrum-guide.html. Acesso em: 19 out. 2024.

[2] GOOGLE Cloud. **Beginner: Introduction to Generative AI Learning Path**. Google Cloud Skills Boost, 2024. Disponível em: https://www.cloudskillsboost.google/paths/118. Acesso em: 01 jul. 2024.

[3] Virtus - Núcleo de Pesquisa, Desenvolvimento e Inovação em Tecnologia da Informação. Universidade Federal de Campina Grande, 2024. Disponível em: https://www.virtus.ufcg.edu.br/#sobre. Acesso em: 19 out. 2024.

[4] DEEPLearning.AI. **Prompt Engineering with LLaMA 2**. DeepLearning.AI, 2024. Disponível em: https://www.deeplearning.ai/short-courses/prompt-engineering-with-llama-2/. Acesso em: 12 ago. 2024.

[5] LAWTON, George. **What is Gen AI? Generative AI explained**. TechTarget, 2023. Disponível em: https://www.techtarget.com/searchenterpriseai/definition/generative-AI. Acesso em: 22 out. 2024.

[6] IBM. **O que é inteligência artificial (IA)?**. IBM, 2024. Disponível em: https://www.ibm.com/br-pt/topics/artificial-intelligence. Acesso em: 22 out. 2024.

[7] FARQUHAR, S., KOSSEN, J., KUHN, L., GAL, Y. **Detecting hallucinations in large language models using semantic entropy**. Nature, v. 630, p. 625–630, 2024. Disponível em: https://www.nature.com/articles/s41586-024-07421-0. Acesso em: 22 out. 2024.

[8] IBM. **What is retrieval-augmented generation?**. IBM, 2023. Disponível em: https://research.ibm.com/blog/retrieval-augmented-generation-RAG. Acesso em: 22 out. 2024.

[9] RAGAS. **List of available metrics**. RAGAS Documentation, 2024. Disponível em: https://docs.ragas.io/en/stable/concepts/metrics/available_metrics/. Acesso em: 22 out. 2024.

[10] CLOUDFLARE. **What is a Vector Database?**. Cloudflare, 2024. Disponível em: https://www.cloudflare.com/learning/ai/what-is-vector-database/. Acesso em: 30 out. 2024.