Curso Prompt Engineering Avançado

Professor(a): Dr Wandré Nunes de Pinho Veloso

ibmec

Objetivos

- **Geral:**
  - Capacitar os servidores no ciclo completo da Engenharia de IA Generativa, desde a criação de prompts avançados para investigação até o deploy eficiente de modelos..

Objetivos

- **Específicos:**
  - Dominar técnicas de Prompt Engineering Avançado (Few-Shot, CoT e Multimodal) para triagem de ocorrências e análise de evidências.
  - Aplicar técnicas de Compressão de Modelos (Quantização e Pruning) para viabilizar IA em hardware limitado.
  - Implementar Fine-Tuning eficiente (LoRA) para especializar modelos em linguagem policial.
  - Realizar o Deploy.

O que faremos

- Projeto IntelliDoc – O Escrivão Digital

Aula 1

Fundamentos, Arquitetura e Engenharia de Prompt

ibmec

Objetivos

1. Entender o Problema: Por que a PCDF precisa de IA?
2. Fundamentos: A diferença entre Programação Clássica e IA Generativa.
3. Arquitetura: O conceito de API e LLM Local (Privacidade).
4. Prática: Criar a API "IntelliDoc" v1.
5. Técnica: Engenharia de Prompt (Zero-Shot vs Few-Shot).

O Cenário Real (O Problema)

- **O Gargalo:** O volume de BOs e inquéritos cresce exponencialmente.
- **A Realidade:** Policiais gastam horas lendo relatos para classificar crimes simples (Furto vs Roubo).
- **A Missão:** Criar um "Escrivão Digital" que roda localmente (sem internet), lê o relato e faz a triagem automática.
- **O Projeto:** IntelliDoc API

Conceito Fundamental: Determinístico vs Probabilístico

Por que a IA às vezes erra?

- Programação Clássica (Determinística):
  - `if x > 10 return "Maior"`
  - Sempre dá o mesmo resultado. É como um trem nos trilhos.
- IA Generativa (Probabilística):
  - Ela não "sabe" a verdade.
  - Ela calcula a probabilidade da próxima palavra.
  - É como o "autocompletar" do celular, mas treinado com toda a internet.
  - “Eu gosto muito de comer...” O que vem a seguir?

Analogia 1: O "Estagiário" (LLM)

- Imagine o modelo de IA (Llama 3.2) como um estagiário:
  1. Muito culto: Leu todos os livros de Direito Penal do mundo (no treinamento).
  2. Sem vivência: Nunca entrou numa delegacia real.
  3. O perigo: Se você não der ordens claras, ele tenta ser criativo ou inventar fatos para te agradar (alucina).
- Nosso trabalho não é escrever código. É gerenciar esse estagiário.

Arquitetura do Sistema: Por que Local?

- Não usaremos ChatGPT ou Cloud. Por quê?
  1. Privacidade e sigilo dados de inquéritos não podem ir para servidores nos EUA.
  2. Latência: A delegacia precisa funcionar mesmo se a internet cair.
  3. Custo: Rodar na CPU local é "grátis" comparado a pagar por token.
- Solução:
  - Ollama: O motor que roda a IA no PC da delegacia.
  - Llama 3.2: O modelo "cérebro" (otimizado e leve).

Analogia 2: O Restaurante (A API)

- Como o Policial (Front-end) conversa com a IA (Back-end)?
  - O Cliente (Policial): Está na mesa, tem um pedido (relato do crime). Não entra na cozinha.
  - A Cozinha (Ollama/IA): Onde o processamento pesado acontece. É complexo e perigoso.
  - O Garçom (FastAPI):
    - Anota o pedido.
    - Leva para a cozinha.
    - Traz o prato pronto (JSON).
- Hoje, nós vamos construir o Garçom.

Mão na massa

Parâmetro Crítico: Temperatura

- A "Temperatura" (0.0 a 1.0) controla a criatividade da IA.
  - Temperatura 1.0 (O "Estagiário Bêbado"):
    - Super criativo, poeta, escreve e-mails emocionantes.
    - Risco: Inventa artigos de lei que não existem (Alucinação).
  - Temperatura 0.0 (O "Estagiário Sóbrio"):
    - Burocrata, repetitivo, preciso.
    - Meta: Para a PCDF, usaremos sempre 0.0. Queremos precisão, não poesia.

Engenharia de Prompt: O que é?

- "*Prompt Engineering é a arte de programar em linguagem natural."*
- Não basta pedir. É preciso saber como pedir.
- A qualidade da resposta (Output) depende inteiramente da qualidade da instrução (Input).

Técnica 1: Zero-Shot (Tiro no escuro)

- É quando pedimos algo para a IA sem dar exemplos.
- Prompt:
  - "Classifique este crime: Dois caras numa moto levaram meu celular.“
  - "Dois sujeitos numa moto pararam ao meu lado, mostraram uma arma na cintura e pediram o celular"
- Resultado Provável (e ruim):
  - "Isso parece ser um assalto."
  - "Artigo 157."
  - "ROBO" (Erro ortográfico comum em modelos pequenos).
  - "Assalto a mão armada"
- Problema: A IA não sabe o padrão que você quer.

Técnica 2: Few-Shot (Dando poucos exemplos)

- A técnica mais poderosa para modelos locais. Mostramos o padrão antes de pedir.
- Prompt:

  Exemplo 1: "Levaram meu celular da mesa." -> FURTO

  Exemplo 2: "Apontaram uma arma." -> ROUBO

  Exemplo 3: "Clone do cartão." -> ESTELIONATO

  Agora classifique: "Dois caras numa moto levaram meu celular." -> ?
- Resultado: ROUBO (A IA entende o padrão e copia).

Técnica 3: *Chain of Thought* (Prévia)

- Para casos complexos, a IA precisa "pensar alto".
  - Problema: Se a IA responde rápido, ela chuta.
  - Solução (CoT): Obrigamos a IA a explicar o raciocínio.
  - E se o crime for complexo? Tipo alguém que pegou o carro emprestado e não devolveu? É furto? Apropriação indébita?
- Prompt: "Analise passo a passo se houve violência. Depois, se houve subtração. Só então dê o veredito."
  - Veremos isso em detalhes depois.

Aula 2

A IA que "Pensa" e "Vê" (Raciocínio Avançado e Multimodalidade)

ibmec

Qual o melhor tipo de raciocínio para este caso?

- “O suspeito afirma que estava em casa em Taguatinga às 22:00. Porém, o radar da EPTG registrou o carro dele passando em alta velocidade sentido Plano Piloto às 21:55. O trajeto leva no mínimo 15 minutos."

O Problema da "Intuição Rápida"

- **O Erro Humano (e da IA):** O "Sistema 1" (Rápido/Instintivo) vs "Sistema 2" (Lento/Analítico).
- **A Falha:** Modelos de linguagem tendem a prever a próxima palavra mais provável, não a mais lógica.
- **O Exemplo Policial:** "Ele levou meu carro emprestado e não devolveu".
  - IA Rápida (Zero-Shot): "ROUBO" (Errado).
  - Realidade Jurídica: "Apropriação Indébita".

Técnica 1: *Chain of Thought* (CoT)

Pense passo a passo

- **Definição:** Técnica de Engenharia de Prompt que força o modelo a decompor problemas complexos em etapas intermediárias.
- **Por que funciona?** Ao gerar texto explicando o raciocínio, o modelo cria seu próprio contexto para a resposta final, reduzindo **alucinações**.
  - “A mente quando não sabe a verdade, cria a sua própria versão dela” Sêneca, filósofo
- **Aplicação PCDF:** Tipificação penal correta baseada nos elementos do crime (Violência? Fraude? Posse prévia?).
- main2-1.py

Técnica 2: Multimodalidade (Visão Computacional)

- **O Desafio:** O Llama 3.2 é "cego". Ele só processa texto.
- **A Solução:** Vision Transformers (ViT).
- **Como funciona:**
  1. A imagem é fatiada em pequenos quadrados (patches).
  2. Cada quadrado vira uma lista de números (vetor).
  3. Um "Projetor" traduz esses números para a linguagem que o LLM entende.
- **Resultado:** A IA "lê" a imagem como se fossem palavras.
- main2-2.py

O "IntelliDoc" Integrado

- **O Fluxo Final:**
  1. **Input:** Policial envia FOTO + RELATO.
  2. **Visão:** Moondream descreve a foto ("Vejo um carro azul amassado").
  3. **Raciocínio:** Llama compara o Relato ("Bateram no meu carro vermelho") com a Visão.
  4. **Veredito:** "Inconsistência detectada".
- main2-3.py

Aula 3

Engenharia de Contexto e Memória Infinita (RAG + RLM)

ibmec

O Problema da Memória Curta

- Context Window **(Janela de Contexto):** A quantidade de texto que a IA consegue "ler" de uma vez.
- **A Limitação:** O Llama 3.2 tem uma janela de 128k tokens (teóricos), mas na prática, quanto mais texto, mais ela se perde ("Lost in the Middle").
- **O Desafio PCDF:** Um inquérito pode ter 2.000 páginas. Não cabe no prompt.
- **A Solução:** Não dar o livro todo para a IA. Dar apenas a página certa.

Técnica 1: RAG (*Retrieval Augmented Generation*)

"Geração Aumentada por Recuperação“

1. **Ingestão:** Quebramos o PDF em pedaços pequenos (Chunks).
2. **Embedding:** Transformamos esses pedaços em listas de números (Vetores).
   - **Ex:** "Tráfico" vira `[0.1, 0.9, 0.3...]`.
3. **Armazenamento:** Guardamos no ChromaDB (Banco Vetorial).
4. **Recuperação:** Quando o usuário pergunta, buscamos os pedaços matematicamente parecidos e entregamos para a IA.

- main3-1.py e main3-2.py

Técnica 2: RLM (*Recursive Language Modeling*)

"O Truque do GTA"

- **O Problema:** E se eu precisar resumir o inquérito inteiro de 2.000 páginas? O RAG não serve (ele só pega pedaços).
- **A Solução (RLM):** Processamento Recursivo.
  1. Dividir o texto em blocos.
  2. Resumir cada bloco.
  3. Juntar os resumos e resumir de novo.
- **Resultado:** Contexto infinito condensado em uma pílula de conhecimento.
- main3-3.py

IBMEC.BR

/IBMEC

IBMEC

@IBMEC_OFICIAL

@IBMEC

Ibmec