



Documentação Técnica – IA com Embeddings (Projeto Signa)



Arquivos do Projeto

Arquivo	Função
<code>gerador_de_modelo.py</code>	Script completo de pré-processamento e treino de um modelo GPT-2 com os dados institucionais da Signa
<code>signa.txt</code>	Arquivo gerado com o corpus textual (perguntas e respostas sobre a Signa) usado como base de treino
<code>API_gpt2_finetune.py</code>	API em FastAPI que recebe um prompt e retorna texto gerado pelo modelo finetunado
<code>meu_modelo/</code> ou <code>model_treinado/</code>	Pasta com os arquivos do modelo e tokenizer salvos após o treino (<code>pytorch_model.bin</code> , <code>config.json</code> , etc.)
<code>requirements.txt</code>	Lista de bibliotecas necessárias para executar o treino e a API



Lógica de Funcionamento



Treinamento

1. O corpus `signa.txt` é carregado como lista de textos.
2. As entradas são tokenizadas com `distilgpt2` (modelo leve do GPT-2).
3. Cada entrada é usada como entrada e rótulo (self-supervised).
4. O modelo é treinado com `Trainer` por 3 épocas.
5. Ao final, o modelo é salvo com `save_pretrained()`.



API

- Recebe um `prompt` via `POST /gerar`
- Usa o pipeline `text-generation` com o modelo treinado
- Retorna o texto gerado com base no prompt

Como Executar

1. Treinar o modelo

```
python gerador_de_modelo.py
```

2. Iniciar a API

```
uvicorn API_gpt2_finetune:app --reload
```





Exemplo de chamada via **curl**

```
curl -X POST http://localhost:8000/gerar \
-H "Content-Type: application/json" \
-d '{"prompt": "Pergunta: Quais serviços a Signa oferece?\nResposta:"}'
```

Requisitos (requirements.txt)

```
transformers
datasets
torch
pydantic
```

Pontos Positivos

-  Total controle sobre os dados do modelo
-  Geração de texto mais natural, contínua e criativa
-  Funciona offline após o treino
-  API leve, sem dependência de OpenAI ou serviços externos

Pontos a Melhorar

-  Pode gerar respostas irrelevantes ou repetitivas se o prompt for curto

- ⚠️ Modelo pequeno (distilgpt2) tem limitações de entendimento
- 🧠 Não sabe quando não sabe (pode "alucinar")
- 🔄 Sem controle de contexto ou memória de diálogos

Relatório de Testes

Prompt de entrada	Resultado gerado	Avaliação
"Pergunta: O que é a Signa? \nResposta:"	"A Signa é uma empresa portuguesa especializada..."	✅ Ótimo
"Pergunta: Como entrar em contato? \nResposta:"	"Você pode ligar para +351 214 127 780..."	✅ Correto
"Pergunta: Vocês atendem empresas fora de Portugal? \nResposta:"	"O site não informa atuação internacional..."	✅ Adequado
"Pergunta: Qual é o nome do fundador? \nResposta:"	"A Signa é especializada..."	⚠️ Hallucinação
"A Signa oferece plano de saúde?"	"Sim, a Signa..."	❌ Invenção (não real)

Melhorias Futuras

- 🔍 Incluir checagem de factualidade (RAG ou verificação por base vetorial)
- 📈 Usar GPT maior (gpt2-medium ou gpt2-xl)
- 🎯 Implementar filtros para detectar perguntas fora de escopo
- 💾 Habilitar salvamento dos prompts e respostas geradas para avaliação contínua
- 🌐 Oferecer interface web para testes abertos (com streamlit ou Gradio)

Arquivo	Função
Base_de_dados.txt	Base de conhecimento da empresa Signa: perguntas, respostas e fontes extraídas do site
gerador_de_embeddings.py	Script que transforma as perguntas da base em vetores (embeddings)
embeddings.pt.pkl	Arquivo binário com embeddings, perguntas e respostas





Arquivo	Função
<code>API_embeddings_similaridade.py</code>	API em FastAPI que responde perguntas com base em similaridade semântica

Lógica de Funcionamento





1. O usuário envia uma **pergunta**
2. A pergunta é convertida em **vetor (embedding)** com o modelo `paraphrase-multilingual-MiniLM-L12-v2`
3. O sistema calcula a **similaridade com as perguntas da base**
4. A resposta mais próxima (acima do threshold de `0.65`) é retornada via API

Como Executar

Pontos Positivos

-  Controle total sobre o conteúdo respondido
-  Fácil manutenção e expansão da base
-  Respostas sempre baseadas em informações reais e confiáveis
-  API leve, performática e de fácil integração

Pontos a Melhorar

-  Pode confundir perguntas muito parecidas
-  Responde apenas a primeira correspondência (sem re-ranking)
-  Não entende múltiplas intenções na mesma pergunta
-  Limitações semânticas se a entrada do usuário for muito fora da base

Relatório de Testes

Casos que funcionaram:

Entrada	Resultado
"O que é a Signa?"	✅ Resposta correta
"Como falo com a Signa?"	✅ Resposta de contato
"Quais serviços vocês oferecem?"	✅ Listou serviços da empresa

Casos com erro ou baixa precisão:

Entrada	Problema
"Onde fica a Signa?"	❌ Trouxe info sobre clientes
"Vocês oferecem consultoria?"	⚠️ Score abaixo do limite

💡 Melhorias Futuras

- 🔍 Implementar **re-ranking** entre top 3 candidatos
- 🧠 Treinar mini-LLM com foco em domínio fechado
- 📁 Agrupar por **intenção/categoria** (ex: localização, serviços, contato)
- 📊 Adicionar logs, métricas e sistema de feedback do usuário