Meta-chunking semântico

IDEIA

Muitas vezes projetos de LLM com RAG ignoram o aspecto crucial da fragmentação de texto dentro de seu pipeline, o que impacta a qualidade das tarefas intensivas em conhecimento (Zhao, J., Meta-chunking: Learning efficient text segmentation via logical perception)

Técnicas de chunking inteligentes combinadas com o uso de RAG podem gerar embeddings mais ricos, potencialmente melhorando o desempenho de LLMs, mesmo aqueles com menor capacidade, ao focar o contexto nas informações mais críticas para a tarefa de codificação?

Faz sentido fragmentarmos o texto através de sumarização semântica em vez de divisão sequencial, reduzindo o número de tokens e aumentando a relevância dos chunks, e assim obter melhores resultados na geração de códigos mesmo com LLMs mais simples?

EXPERIMENTO

Nosso trabalho explora o pré-processamento focado em agrupar e resumir informações-chave de documentos-fonte criando tokens por contexto que são utilizados no RAG e no seu impacto na geração de códigos-fonte.

Como exemplo, carregamos a documentação da API dos Correios e, no contexto de engenharia de software, a IA pôde responder perguntas sobre a API, gerando código Python com base na documentação oficial.

Para comparar, criamos um pré-processamento alternativo com divisão sequencial e analisamos a diferença dos códigos-fonte e a quantidade de tokens gerados/utilizados. Além disso, executamos cenários de testes com Llama e o Deep Seek, para verificar a diferença de código entre os dois LLMs.

PROCESSO

API - Correios

Lê PDF e extrai texto usando OCR

Pré-processamento sequencial

texto inteiro é quebrado em chunks de 1000 caracteres com overlap de 10%.

chunk = 10000 caracteres do texto extraido do pdf

Pré-processamento semântico

Cria lista de tópicos do sumario do pdf

Para cada tópico, solicita a LLM (deepseek-r1:32b) resumo de 3 a 4 sentenças

chunk = descrição do texto extraido de cada topico

Embedding
(Vetoriza os chunks e salva em bancos separados por tipo de pré-processamento)
modelo: jinaai/jina-embeddings-v3

1 ----------
2 ----------
3 ----------
4 ----------

Banco de dados vetorial
(Armazena a vetorização)

Pergunta inserida pelo desenvolver:
"Como posso emitir um rótulo de pré-postagem?"

Vetoriza a pergunta

RAG
(Recebe a pergunta do desenvolvedor, busca e retorna os vetores por similaridade)

Executa a consulta na LLM com o prompt ampliado e gera resposta com o código-fonte

Analisa código-fonte gerado

CÓDIGO GERADO

Pré-processamento semântico:

Olá! Para emitir um rótulo de pré-postagem, você precisará seguir os passos abaixo:

1. Faça uma requisição POST para a URL `https://apihom.correios.com.br/prepostagem/v1/prepostagens/rotulo` com o corpo da requisição contendo as IDs das pré-postagens e o tipo de rótulo desejado.
2. No corpo da requisição, você deve incluir um objeto JSON com os seguintes campos:
 * `idsPrePostagem`: uma lista de IDs das pré-postagens geradas anteriormente (até 6 IDs por requisição no caso de cartas simples).
 * `tipoRotulo`: o tipo de rótulo desejado, que pode ser "P" para padrão ou "R" para reduzido.

Aqui está um exemplo de código em Python que demonstra como fazer essa requisição:

```python
import requests

# Dados da pré-postagem
ids_pre_postagem = ["PR1JncySxyRO0Eq0OaQp6dSQ", "PR2qlXxTEeTkGX4VLJE05Daw"]
tipo_rotulo = "R"

# Dados de autenticação (token)
token = "SEU_TOKEN_AQUI"

# Requisição POST
url = "https://apihom.correios.com.br/prepostagem/v1/prepostagens/rotulo"
headers = {"Authorization": f"Bearer {token}"}
dados = {
    "idsPrePostagem": ids_pre_postagem,
    "tipoRotulo": tipo_rotulo
}

response = requests.post(url, headers=headers, json=dados)

if response.status_code == 200:
    print("Rótulo de pré-postagem emitido com sucesso!")
else:
    print(f"Erro ao emitir rótulo de pré-postagem: {response.text}")
```

Lembre-se de substituir `SEU_TOKEN_AQUI` pelo seu token de autenticação válido.

Espero que isso ajude! Se tiver alguma dúvida ou precisar de mais ajuda, não hesite em perguntar.

Pré-processamento tradicional:

Entendo sua pergunta!

Para emitir um rótulo de pré-postagem, você precisa seguir os passos abaixo:

1. Gerar a pré-postagem utilizando a API de pré-postagem (v1/prepostagens/lote ou v1/prepostagens/rotulo/assincrono/pdf).
2. Obter o ID da pré-postagem gerada.
3. Utilizar o ID da pré-postagem para emitir o rótulo de pré-postagem.

Aqui está um exemplo de código em Python que demonstra como emitir um rótulo de pré-postagem:

```python
import requests

# Dados de autenticação
usuario = "seu_usuario"
senha = "sua_senha"

# Dados da pré-postagem
ids_pre_postagem = ["PRgddyJTC0QMyNqlCX7w0H9A"]
tipo_rotulo = "P"
formato_rotulo = "ET"

# Requisição para gerar a pré-postagem
url = "https://apihom.correios.com.br/prepostagem/v1/prepostagens/rotulo/assincrono/pdf"
headers = {"Authorization": f"Bearer {usuario}:{senha}"}
dados = {
    "idsPrePostagem": ids_pre_postagem,
    "tipoRotulo": tipo_rotulo,
    "formatoRotulo": formato_rotulo
}

response = requests.post(url, headers=headers, json=dados)
if response.status_code == 200:
    # Obter o ID da pré-postagem gerada
    id_pre_postagem = response.json()["idPrePostagem"]

    # Emitir o rótulo de pré-postagem
    url_rotulo = f"https://apihom.correios.com.br/prepostagem/v1/prepostagens/rotulo/{id_pre_postagem}"
    headers_rotulo = {"Authorization": f"Bearer {usuario}:{senha}"}
    response_rotulo = requests.get(url_rotulo, headers=headers_rotulo)
    if response_rotulo.status_code == 200:
        # Baixar o arquivo de rótulo
        with open("rotulo.pdf", "wb") as arquivo:
            arquivo.write(response_rotulo.content)
        print("Rótulo de pré-postagem emitido com sucesso!")
    else:
        print(f"Erro ao emitir o rótulo de pré-postagem: {response_rotulo.text}")
else:
    print(f"Erro ao gerar a pré-postagem: {response.text}")
```

Lembre-se de substituir `seu_usuario` e `sua_senha` pelos dados de autenticação reais.

RESULTADOS INICIAIS

Observamos a geração de códigos mais complexos e estruturados nos casos que utilizamos o pré-processamento com semântica, bem como redução na quantidade de tokens nos prompts.

Pré-Processamento Semântico	Pré-Processamento Sequencial
Qualidade Código	**Qualidade Código**
Código específico e detalhado dentro do contexto	Código genérico e prolixo sem contexto

CONCLUSÃO

Observamos ganhos no uso do pré-processamento semântico com códigos mais otimizados e coerentes com a documentação,

Trabalhos futuros:

- Análise de impacto com entrada de documentos distintos.

- Medições da qualidade dos códigos gerados quanto a complexidade ciclomática, segurança, cobertura de testes, duplicação de código e densidade de bugs.

Confira o código:

https://colab.research.google.com/drive/1izkP09QGH9Yq-51m35fJ70oOeLqwnlbs?usp=sharing

Grupo: Anderson Bittencourt, Ana Teresa C. Vicentini, Lys Kang e Renan S. Marques