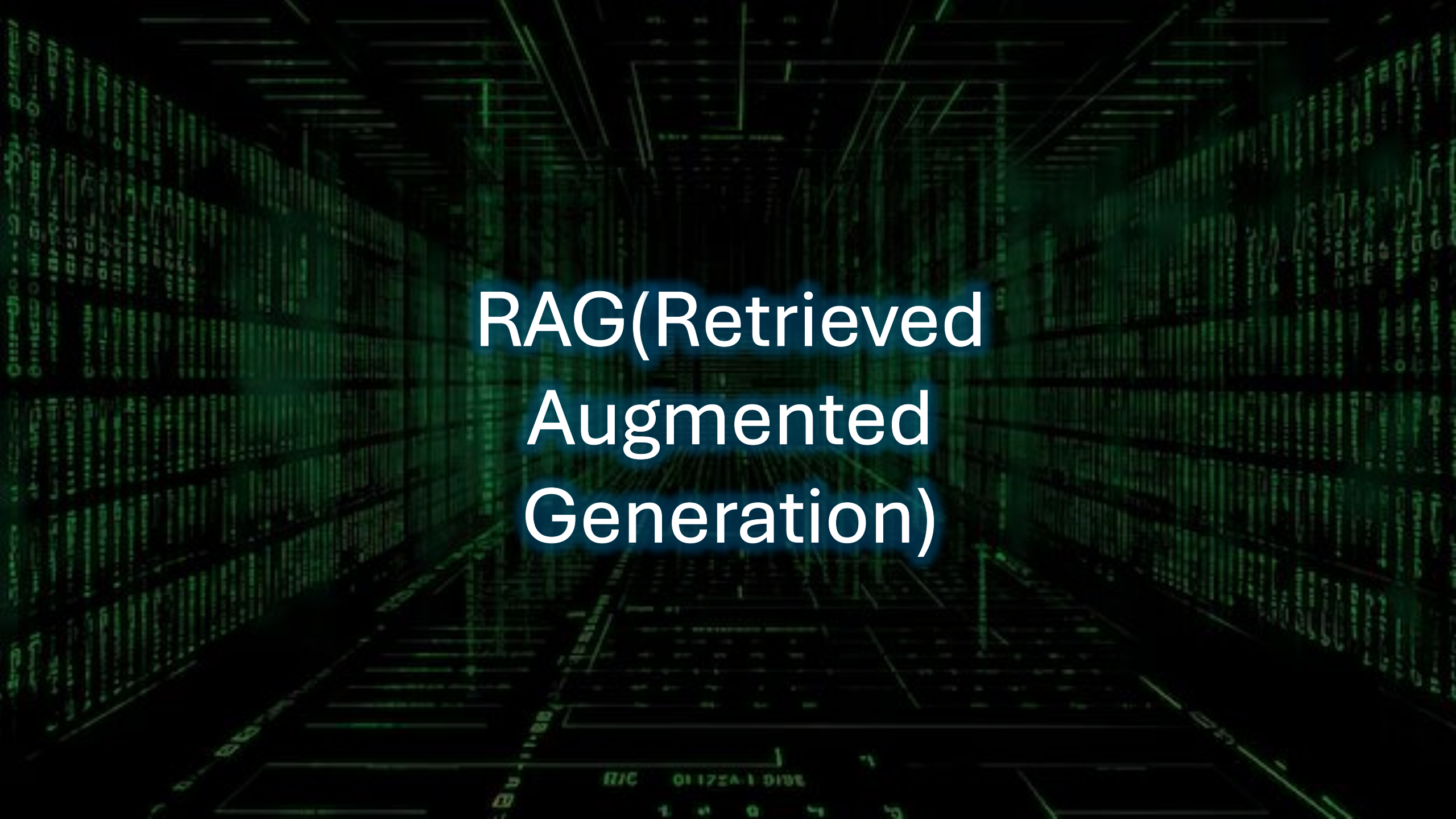
RAG(Retrieved
Augmented
Generation)

Problema

- Modelos de linguagem (LLMs), como BERT, T5 ou GPT, têm uma limitação eles não sabem informações que não estejam em sua base de treinamento.

Isso causa dois problemas:

Informações desatualizadas ou ausentes: O modelo não conhece fatos novos ou específicos.

Alucinação: Ele inventa respostas quando não encontra algo parecido no seu treinamento.

Solução

1. **Base de conhecimento:** Os documentos são fornecidos como textos em um dicionário que representam a fonte de verdade que o sistema usará para responder perguntas.

2. Geração de Embeddings: O modelo all-MiniLM-L6-v2 converte cada texto em um vetor numérico, esses vetores capturam o significado dos documentos.

3. Recuperação (Retrieve): Para uma pergunta do usuário o sistema, gera o embedding da pergunta, calcula a similaridade com todos os documentos e seleciona o documento mais relevante (top_k=1)

4. Modelo de Perguntas e Respostas (Answer): Usamos um modelo da HuggingFace destinado a QA, esse modelo consegue extrair, dentro do contexto, a resposta mais apropriada.

5. Pipeline RAG: A função rag() combina tudo, recupera o documento mais relevante, envia contexto + pergunta para o modelo QA e devolve a resposta produzida pelo LLM

Código

```
pip install transformers sentence-transformers numpy
```

```
import numpy as np
from sentence_transformers import SentenceTransformer, util
from transformers import pipeline
documents = {
    "doc1.txt": "O Python é uma linguagem de programação popular para ciência de dados.",
    "doc2.txt": "Modelos de linguagem como BERT e T5 são amplamente usados em NLP.",
    "doc3.txt": "RAG significa Retrieval Augmented Generation, uma técnica que combina recuperação com geração.",
}
doc_texts = list(documents.values())
embedder = SentenceTransformer("all-MiniLM-L6-v2")
doc_embeddings = embedder.encode(doc_texts)
def retrieve(query, top_k=1):
    query_emb = embedder.encode(query)
    scores = util.cos_sim(query_emb, doc_embeddings)[0]
    top_results = scores.topk(k=top_k)
    best_docs = [doc_texts[idx] for idx in top_results.indices]
    return best_docs
qa = pipeline("question-answering", model="distilbert-base-uncased-distilled-squad")
def rag(query):
    context = retrieve(query, top_k=1)[0]

    answer = qa({
        "question": query,
        "context": context
    })
    return {
        "query": query,
        "contexto_usado": context,
        "resposta": answer["answer"]
    }
result = rag("O que é RAG?")
print(result)
```