Ejercicio Final: Creación de Grafo de Conocimiento de Prompt Engineering

🏷 Objetivo del Ejercicio

Crear un grafo de conocimiento interactivo que capture y visualice todos los conceptos de prompt engineering enseñados en clase, utilizando el audio grabado de la sesión como fuente de datos.

Competencias a Evaluar

- **Prompt Engineering**: Aplicación de técnicas PNI, Few-shot, Zero-shot, Chain of Thought
- **Estructuración de Salida**: Uso de Pydantic y formato JSON
- **Integración de APIs**: Gemini API con LangChain
- **Procesamiento de Audio**: Transcripción y análisis de contenido
- **Visualización de Datos**: Creación de grafos con NetworkX
- **Programación Asistida**: Uso de IDEs con IA (Cursor, Windsurf)

Descripción del Ejercicio

Utilizarás el audio grabado de esta clase para crear un sistema que:

1. **Procese el audio** usando la API de Gemini
2. **Extraiga conceptos y relaciones** mediante prompt engineering avanzado
3. **Estructure la información** en formato JSON usando Pydantic
4. **Genere un grafo visual** que represente el mapa de conocimiento

🛠 Herramientas Requeridas

- **APIs**: Google Gemini API
- **Frameworks**: LangChain
- **Python Libraries**:
  - `pydantic` (estructuración de datos)
  - `networkx` (creación de grafos)
  - `matplotlib` o `plotly` (visualización)
  - `langchain-google-genai` (integración con Gemini)
- **IDEs recomendados**: Cursor o Windsurf con asistente IA

📊 Estructura de Datos Esperada

Modelo Pydantic para Conceptos

python

```python
from pydantic import BaseModel
from typing import List, Optional

class Concepto(BaseModel):
    nombre: str
    definicion: str
    categoria: str  # "tecnica", "herramienta", "metodologia", "concepto"
    importancia: int  # 1-5 (1=básico, 5=avanzado)
    ejemplos: Optional[List[str]] = []

class Relacion(BaseModel):
    concepto_origen: str
    concepto_destino: str
    tipo_relacion: str  # "es_parte_de", "requiere", "mejora", "se_aplica_con"
    fuerza: float  # 0.0-1.0 (qué tan fuerte es la relación)

class GrafoConocimiento(BaseModel):
    conceptos: List[Concepto]
    relaciones: List[Relacion]
    tema_principal: str
    resumen: str
```

Pasos del Ejercicio

Paso 1: Configuración del Entorno

bash

```bash
pip install langchain-google-genai pydantic networkx matplotlib plotly
```

Paso 2: Procesamiento del Audio

Crea un prompt que utilice **Chain of Thought** para analizar el contenido:

```python
PROMPT_ANALISIS = """
Eres un experto en análisis de contenido educativo. Analiza la siguiente transcr

INSTRUCCIONES:
1. IDENTIFICA todos los conceptos mencionados
2. CLASIFICA cada concepto por categoría y nivel de importancia
3. DETECTA las relaciones entre conceptos
4. RAZONA paso a paso tu análisis

TRANSCRIPCIÓN:
{transcripcion}

PIENSA PASO A PASO:
1. Primero, lista todos los conceptos mencionados
2. Luego, categoriza cada concepto
3. Después, identifica las relaciones
4. Finalmente, estructura la información

FORMATO DE SALIDA: JSON siguiendo el esquema GrafoConocimiento
"""
```

Paso 3: Implementación del Sistema

Desarrolla las siguientes funciones clave:

1. **Transcribir audio** usando Gemini API
2. **Extraer conceptos** con prompt engineering
3. **Validar estructura** con Pydantic
4. **Generar grafo** con NetworkX
5. **Visualizar resultados**

Paso 4: Técnicas de Prompt Engineering a Aplicar

Few-Shot Learning

Proporciona 2-3 ejemplos de conceptos y relaciones bien estructurados antes del análisis real.

Zero-Shot con Instrucciones Claras

Define exactamente qué constituye un "concepto" y una "relación" en el contexto educativo.

Chain of Thought

Solicita razonamiento paso a paso para la identificación de relaciones complejas.

Estructuración de Salida

Usa Pydantic para garantizar formato JSON consistente.

📈 Visualización Esperada

El grafo debe mostrar:

- **Nodos**: Conceptos (tamaño proporcional a importancia)
- **Colores**: Categorías diferentes (técnicas=azul, herramientas=verde, etc.)
- **Aristas**: Relaciones (grosor proporcional a fuerza de relación)
- **Layout**: Algoritmo que agrupe conceptos relacionados

Código Base para Visualización

python

```python
import networkx as nx
import matplotlib.pyplot as plt

def crear_grafo_visual(grafo_conocimiento: GrafoConocimiento):
    G = nx.Graph()

    # Agregar nodos
    for concepto in grafo_conocimiento.conceptos:
        G.add_node(concepto.nombre,
                   categoria=concepto.categoria,
                   importancia=concepto.importancia)

    # Agregar aristas
    for relacion in grafo_conocimiento.relaciones:
        G.add_edge(relacion.concepto_origen,
                   relacion.concepto_destino,
                   peso=relacion.fuerza,
                   tipo=relacion.tipo_relacion)

    return G
```