UNIVERSIDAD EAFIT

Escuela de Ciencias Aplicadas e Ingeniería

ST7003 Procesamiento Natural del Lenguaje

Aplicaciones Avanzadas de NLP

Lecture 07

Contenido

1. Aplicaciones Avanzadas de NLP con LLMs
2. Retrieval-Augmented Generation - RAG

Contenido

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

1. Aplicaciones Avanzadas de NLP con LLMs
2. Retrieval-Augmented Generation - RAG

Aplicaciones Avanzadas de NLP con LLMs

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- Clasificación de Texto
- Análisis de Sentimientos
- Q&A (Preguntas y Respuestas)
- Summarization (Resumen Automático)
- Generación de Texto
- Chatbots y Asistentes Virtuales

Referencias

- Vaswani et al. (2017): Attention Is All You Need
- Brown et al. (2020): Language Models are Few-Shot Learners
- Lewis et al. (2020): Retrieval-Augmented Generation

Sintesis de LLMs

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- Los Modelos de Lenguaje a Gran Escala (LLMs) han revolucionado el procesamiento de lenguaje natural, permitiendo tareas como clasificación de texto, Q&A y generación de contenido.

Evolución de los Modelos NLP

UNIVERSIDAD EAFIT

Escuela de **Ciencias Aplicadas e Ingeniería**

- Modelos Estadísticos: HMM, N-gramas
- Redes Neuronales: LSTM, GRU
- Transformadores: BERT, GPT, T5
- Modelos recientes: GPT-4, LLaMA, Claude

Época	Modelo	Características
Antes de 2010	Modelos basados en reglas y probabilidades	Gramáticas, modelos de Markov ocultos (HMMs), Naive Bayes
2013-2017	Word Embeddings (Word2Vec, GloVe)	Representación distribuida de palabras, captura relaciones semánticas
2017	Introducción de Transformers ("Attention is All You Need")	Nueva arquitectura basada en autoatención
2018	BERT (Bidirectional Encoder Representations from Transformers)	Representación contextualizada del lenguaje
2019-2020	GPT-2, T5	Modelos de generación más potentes
2020-2024	GPT-3, GPT-4, LLaMA, PaLM, Claude	Modelos con miles de millones de parámetros y capacidades multitarea

Clasificación de texto

Definición:

- La clasificación de texto consiste en asignar una categoría predefinida a un fragmento de texto.
- Se usa en filtrado de spam, categorización de noticias y detección de discurso.

Clasificación de texto

UNIVERSIDAD EAFIT

Escuela de Ciencias Aplicadas e Ingeniería

Métodos Tradicionales vs. LLMs

Enfoque	Descripción
Basado en Reglas	Se usan listas de palabras clave para categorizar texto.
Machine Learning Clásico	Se emplean modelos como Naive Bayes, SVM, y Random Forest.
Deep Learning	Se usan RNNs, CNNs y Transformers como BERT y GPT.

Clasificación de texto

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- Usar un **modelo basado en reglas** vs. **GPT-4** para clasificar texto:
  - Un modelo basado en reglas podría usar listas de palabras para clasificar sentimientos, mientras que un LLM puede aprender a interpretar el contexto y detectar ironía o sarcasmo.

Clasificación de texto

UNIVERSIDAD EAFIT

Escuela de Ciencias Aplicadas e Ingeniería

- Un ejemplo.

```
# clasificación de texto
# modelo preentrenado de OpenAI para clasificar reseñas de clientes en positivas o negativas:

from transformers import pipeline

classifier = pipeline("text-classification", model="nlptown/bert-base-multilingual-uncased-sentiment")
result = classifier("Este producto es increíble, me encantó.")
print(result)
```

Análisis de sentimientos

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

Definición

- El análisis de sentimientos identifica emociones y opiniones en texto, útil en marketing y redes sociales.

- **Aplicaciones:** Análisis de reseñas de productos, monitoreo de reputación en redes sociales.

Análisis de sentimientos

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

Un ejemplo:

```
# analisis de sentimientos

from transformers import pipeline
analyzer = pipeline("sentiment-analysis")
analyzer("No me gustó el servicio, fue una experiencia terrible.")
```

Preguntas y Respuestas (Q&A)

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- **Definición**
  - La tarea de Q&A permite responder preguntas específicas con base en un contexto dado.
- **Aplicaciones:**
  - Asistentes virtuales, chatbots de atención al cliente, recuperación de información.

Preguntas y Respuestas (Q&A)

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

¿Qué es Q&A en NLP?

- La tarea de **preguntas y respuestas (Q&A)** en Procesamiento de Lenguaje Natural (NLP) permite que los modelos de lenguaje interpreten una consulta y generen una respuesta precisa.
- Es una de las aplicaciones más avanzadas de los **Large Language Models (LLMs)**
- **S**e usa ampliamente en chatbots, asistentes virtuales y motores de búsqueda inteligentes.

Preguntas y Respuestas (Q&A)

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

Tipos de modelos

Hay distintos enfoques para la tarea de **Preguntas y Respuestas**, dependiendo del tipo de respuesta que se espera y de los datos disponibles:

1. Modelos Extractivos vs. Generativos

Tipo de Q&A	Descripción	Ejemplo de Modelo
Extractivo	Extrae la respuesta directamente de un documento de referencia.	BERT, RoBERTa
Generativo	Genera una respuesta en lenguaje natural basada en el conocimiento del modelo.	GPT-3, GPT-4, T5

Preguntas y Respuestas (Q&A)

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

Ejemplo:

- **Pregunta:** *"¿Quién escribió Hamlet?"*
- **Contexto:** *"Hamlet es una obra de teatro escrita por William Shakespeare en el siglo XVII."*

Modelo Extractivo → "*William Shakespeare*"

Modelo Generativo → "*Hamlet fue escrito por William Shakespeare en el siglo XVII*."

Preguntas y Respuestas (Q&A)

UNIVERSIDAD EAFIT

Escuela de Ciencias Aplicadas e Ingeniería

2. Q&A Basado en Recuperación vs. Basado en Conocimiento Interno

Tipo	Descripción
Basado en Recuperación	Usa una base de datos o documentos para encontrar la mejor respuesta (ejemplo: RAG con FAISS o ChromaDB).
Basado en Conocimiento Interno	El modelo responde solo con lo que aprendió durante su entrenamiento.

GPT-4 y BERT pueden trabajar en ambos enfoques, dependiendo de cómo se configuren.

Preguntas y Respuestas (Q&A)

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

Ejemplo: Uso de un Modelo Extractivo con Hugging Face (BERT-QA)

```
from transformers import pipeline

qa_pipeline = pipeline("question-answering", model="deepset/bert-base-cased-squad2")

context = "El modelo de lenguaje BERT fue desarrollado por Google AI en 2018."
question = "¿Quién desarrolló BERT?"

result = qa_pipeline(question=question, context=context)
print(result['answer'])
```

Ventaja: Este modelo es rápido y preciso cuando se proporciona contexto.
Desventaja: No puede responder sin un documento de referencia.

Preguntas y Respuestas (Q&A)

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

Ejemplo: Uso de GPT-4 para Q&A Generativo

GPT-4 puede responder preguntas sin necesidad de un contexto predefinido.

```
import openai

openai.api_key = "TU_API_KEY"

response = openai.ChatCompletion.create(
model="gpt-4",
messages=[{"role": "user", "content": "¿Quién descubrió la gravedad?"}]
)

print(response['choices'][0]['message']['content'])
```

Ventaja: Puede generar respuestas incluso sin documentos de referencia.
Desventaja: Puede generar respuestas incorrectas si la información aprendida es limitada o incorrecta. (solución: fine-tuning o RAG)

Preguntas y Respuestas (Q&A)

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

Q&A Mejorado con Recuperación de Información (RAG)

Los LLMs pueden mejorar la precisión de sus respuestas combinándolos con **bases de datos vectoriales**.

- **Flujo del Sistema RAG para Q&A**
  - La pregunta del usuario se convierte en un vector (embedding).
  - Se busca en una base de datos (FAISS, ChromaDB) los documentos más relevantes.
  - Los documentos recuperados se pasan al LLM como contexto adicional.
  - El LLM genera la respuesta basada en los documentos y su conocimiento.

Preguntas y Respuestas (Q&A)

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

Ejemplo: Q&A Mejorado con Recuperación de Información (RAG)

```python
from langchain.chat_models import ChatOpenAI
from langchain.chains import ConversationalRetrievalChain
from langchain.vectorstores import Chroma
from langchain.embeddings.openai import OpenAIEmbeddings

# Configurar OpenAI API Key
openai.api_key = "TU_API_KEY"
# Cargar la base de datos de documentos en ChromaDB
vectorstore = Chroma(persist_directory="./chroma_db",
embedding_function=OpenAIEmbeddings())
# Crear un retriever para buscar información en la base de
datos
retriever = vectorstore.as_retriever()
```

```python
# Crear la Conversational Retrieval Chain con GPT-4
qa_chain = ConversationalRetrievalChain.from_llm(
ChatOpenAI(model_name="gpt-4"),
retriever=retriever
)

# Hacer una pregunta con recuperación de documentos
query = "¿Qué es LangChain?"
response = qa_chain({"question": query})
print(response["answer"])
```

Ventaja: El modelo accede a información externa actualizada.
Desventaja: Requiere almacenamiento y procesamiento adicional.

Summarization (Resumen Automático)

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- **Definición**
  - El resumen automático genera versiones condensadas de textos largos sin perder información clave.
- **Métodos**
  - **Extractivo:** Extrae frases clave (Ejemplo: BERTSUM).
  - **Generativo:** Reescribe el contenido en un texto más corto (Ejemplo: T5, Pegasus).
- **Aplicaciones:**
  - Resúmenes de noticias, informes ejecutivos, síntesis de investigaciones.

Summarization (Resumen Automático)

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

- **Ejemplo**

```
from transformers import pipeline
summarizer = pipeline("summarization")

text = "Los modelos de lenguaje han cambiado la forma en que interactuamos con la tecnología..."
summary = summarizer(text, max_length=50, min_length=20, do_sample=False)

print(summary[0]['summary_text'])
```

Generación de Texto

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- **Definición**
  - Los LLMs pueden generar contenido coherente en múltiples idiomas y estilos.
  - En los modelos clásicos, la Generación es muy muy limitada (n-gram LM, LDA, etc)
- **Aplicaciones:**
  - Generación de contenido para blogs, escritura creativa, asistentes de código (Copilot).

Generación de Texto

UNIVERSIDAD EAFIT

Escuela de Ciencias Aplicadas e Ingeniería

- **Ejemplo**

```
import openai

openai.api_key = "TU_API_KEY"
response = openai.ChatCompletion.create(
model="gpt-4",
messages=[{"role": "user", "content": "Escribe un poema sobre la inteligencia artificial."}]
)

print(response['choices'][0]['message']['content'])
```

Chatbots y Asistentes Virtuales

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- **Definición**
  - Un chatbot usa NLP para interactuar con usuarios de manera fluida.
  - (cuales chatbots de hoy en día pasaran el Test de Turing?)
- **Aplicaciones:**
  - Servicio al cliente, asistentes médicos, automatización de procesos.

Conclusiones iniciales

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- **Los LLMs han revolucionado el NLP**, permitiendo mejoras significativas en tareas como clasificación, análisis de sentimientos, Q&A, y una nueva generación de aplicaciones antes no vistas: resumen automático y generación de texto.
- **Cada una de estas aplicaciones tiene un impacto directo en la industria**, desde chatbots hasta sistemas de soporte avanzado.
- **Los modelos como GPT-4, BERT y T5 ofrecen herramientas poderosas** para el procesamiento del lenguaje natural en múltiples contextos. (hacia una IA general)

Contenido

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

1. Aplicaciones Avanzadas de NLP con LLMs
2. Retrieval-Augmented Generation - RAG

RAG - Retrieval-Augmented Generation

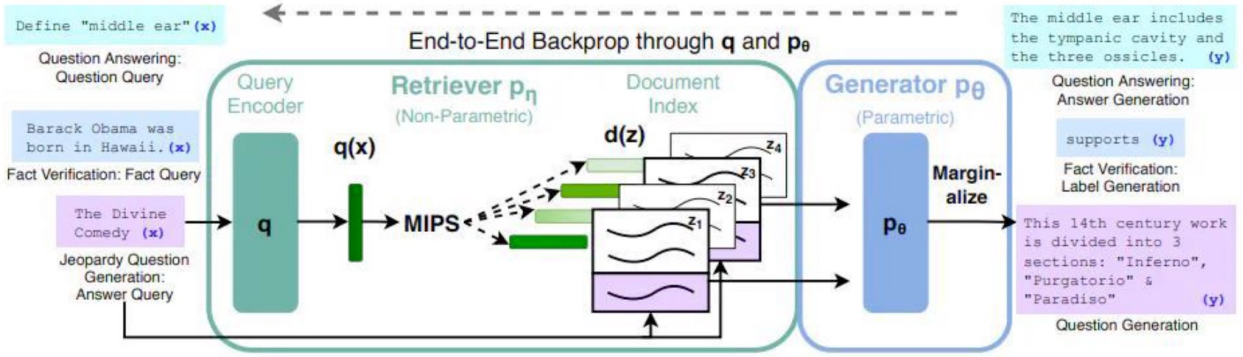


Figure 1: Overview of our approach. We combine a pre-trained retriever (*Query Encoder* + *Document Index*) with a pre-trained seq2seq model (*Generator*) and fine-tune end-to-end. For query x, we use Maximum Inner Product Search (MIPS) to find the top-K documents z_i. For final prediction y, we treat z as a latent variable and marginalize over seq2seq predictions given different documents.

Alternativas para mejorar la recuperación correcta (ej, en QA)

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

- Un modelo de lenguaje solo puede generar texto sobre lo que fue entrenado.
- Prompt Engineering – proporcionando un contexto
  - + / -
- Fine-Tuning
  - + / -
  - Cons:
    - Costoso computacional
    - El Nro de parámetros puede no ser suficiente
    - El fine-tuning no es aditivo, puede “olvidar” lo aprendido
  - Pros:
    - Resultados de mejor calidad en comparación al prompt engineering
    - Tamaño de contexto más pequeño, no necesitamos incluir contexto ni instrucciones largas

Prompt Engineering

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- Técnica de diseñar instrucciones óptimas para obtener respuestas precisas y útiles en modelos de lenguaje como GPT-4, Llama, etc.
- No altera los parámetros del LM, no hay necesidad de fine-tuning
- Reduce la alucinación y mejora la interpretabilidad.
- Ejemplo:
  - Mal prompt: “explica machine learning”
  - Buen prompt: “explica machine learning en términos simples para profesionales en general con ejemplos simples en la vida real”

Prompt Engineering

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- Un buen prompt
  - Contexto: adicione información relevante
  - Claridad: evitar ambigüedades
  - Formato de respuesta deseado: texto, tabla, código
  - Ejemplos: (few-shot promping), ilustra la tarea con ejemplos previos.
- Estrategias de Prompting
  - Zero-shot prompting: sin ejemplos previos
    - Ej: “explica la ley de la gravitación universal”
  - Few-shot prompting

Ejemplo 1: "¿Cómo funciona un motor de combustión interna?" → Respuesta: "..."

Ejemplo 2: "¿Cómo funciona un motor eléctrico?" → Respuesta: "..."

Pregunta: "¿Cómo funciona una turbina eólica?"

Prompt Engineering

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

- **Chain-of-Thought (CoT) Prompting**: Forzar razonamiento paso a paso.

Pregunta: "Juan tiene 3 manzanas, compra 5 más y da 2 a su amigo. ¿Cuántas tiene?"
Pensamiento paso a paso:
1. Juan empieza con 3 manzanas.
2. Compra 5 más, ahora tiene 8.
3. Regala 2, le quedan 6.
Respuesta: 6 manzanas.

Otras estrategias:

- Self-Consistency Prompting: Pedir múltiples respuestas y elegir la más común.
- ReAct Prompting: Alternar entre razonamiento y acción.

Prompt Engineering

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

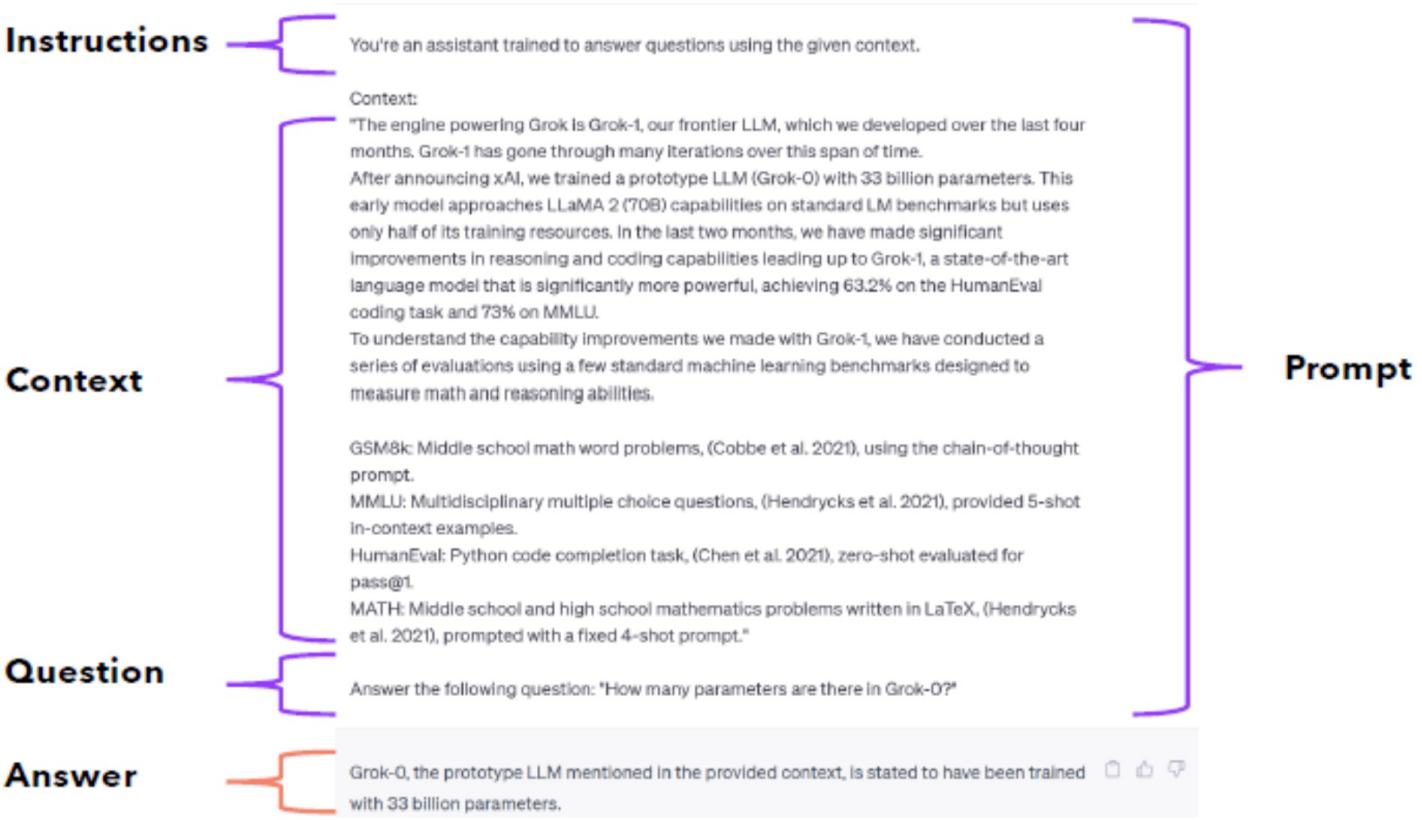
Errores Comunes

- Instrucciones ambiguas: "Describe la historia" (¿de qué?)
- No especificar formato: "Explica el álgebra" (¿breve o extenso?)
- No dar ejemplos cuando es necesario.
- Pedir múltiples tareas a la vez sin claridad.

Prompt Engineering

UNIVERSIDAD EAFIT

Escuela de **Ciencias Aplicadas e Ingeniería**



Prompt Engineering

UNIVERSIDAD EAFIT

Escuela de Ciencias Aplicadas e Ingeniería

Language Models are Few-Shot Learners

Tom B. Brown* Benjamin Mann* Nick Ryder* Melanie Subbiah*

Jared Kaplan† Prafulla Dhariwal Arvind Neelakantan Pranav Shyam Girish Sastry

Amanda Askell Sandhini Agarwal Ariel Herbert-Voss Gretchen Krueger Tom Henighan

Rewon Child Aditya Ramesh Daniel M. Ziegler Jeffrey Wu Clemens Winter

Christopher Hesse Mark Chen Eric Sigler Mateusz Litwin Scott Gray

Benjamin Chess Jack Clark Christopher Berner

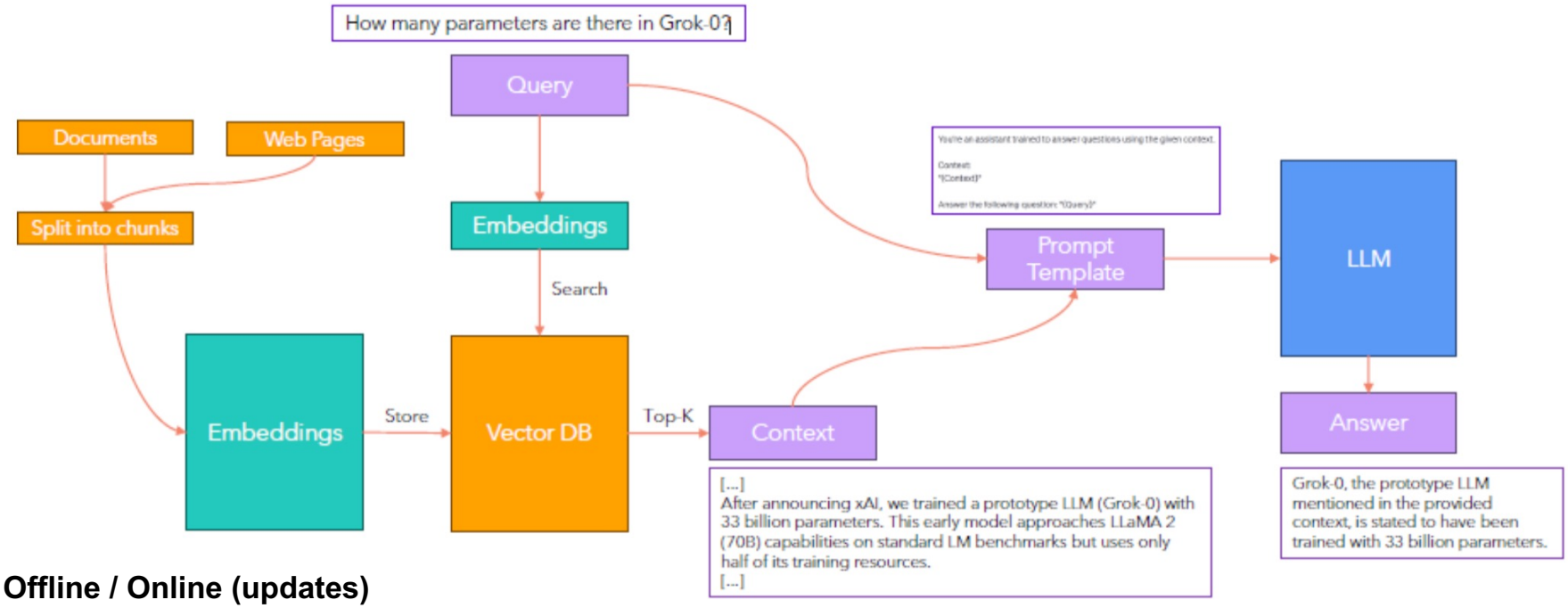
Sam McCandlish Alec Radford Ilya Sutskever Dario Amodei

OpenAI

https://arxiv.org/pdf/2005.14165

Como funciona RAG

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería



Offline / Online (updates)

Definición de RAG

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- Es una técnica de NLP que combina:
  - **generación de texto (LLMs) con**
  - **recuperación de información externa**.
- En lugar de depender solo del conocimiento fijo de un modelo de lenguaje, RAG busca información relevante en bases de datos y la usa para generar respuestas más precisas y actualizadas.

Introducción

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- RAG combina:
  - GENERACIÓN DE TEXTO con
  - RECUPERACIÓN DE INFORMACIÓN

- Ventajas de RAG
  - Respuestas más precisas y actualizadas
  - Reducción de alucinaciones en modelos de lenguaje
  - Integración con bases de conocimiento

Motivación

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- Problemas de LM preentrenados
  - Conocimiento estático y desactualizado
  - Generación de respuestas incorrectas
- Solución con RAG
  - Acceso en tiempo real a bases de datos
  - Respuestas más confiables basadas en documentos externos

RAG

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- Usa una BD externa de documentos como Contexto antes de generar texto
- Que otros métodos hay
  - Fine-Tuning.

- Aplicaciones reales
  - Chatbots avanzados
  - Búsquedas semántica en documentos
  - Resumen automático de artículos extensos

Arquitectura de un Sistema RAG

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

Base de datos de conocimiento:

Offline (cargada en batch) y Online (actualización)

Consultas:

Online

1. Consulta del usuario (query)
2. Búsqueda en la base de datos (IR, Retriever)
    1. Documentos indexados
    2. BD de embeddings (FAISS, Annoy, ChromaDB)
3. Generación basada en contexto (Generator – LLM)
    1. Modelo generativo con la información recuperada
    2. Técnicas de Promping para mejorar la precisión
4. Post-procesamiento y Evaluación de la respuesta

Tecnologías clave

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- **Bases de Datos Vectoriales**: FAISS, Weaviate, Pinecone, ChromaDB
- **Modelos de Embeddings**: OpenAI Embeddings, Cohere, SBERT (sentence BERT)
- **LLMs compatibles con RAG**: GPT-4, Llama2, Claude, Mistral

Fundamentos matemáticos y arquitectónicos de RAG

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

Representación de la Información en RAG

- Un sistema RAG se puede representar de la siguiente manera:
- Dado un **input query** *q*, el modelo debe encontrar un conjunto de documentos relevantes *D={d1,d2,...,dk}* de un corpus *C*, que maximicen la probabilidad de generar la mejor respuesta *r*.
- Esto se modela matemáticamente como:

$$P(r|q) = \sum_{d \in D} P(r|q,d)P(d|q)$$

donde:

- *P(d/q)* es la probabilidad de recuperación del documento d dada la consulta q.
- *P(r/q,d)* es la probabilidad de generar r dado el documento recuperado.

Proceso RAG

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

1. **Retriever:** Se usa un modelo de recuperación $P(d|q)$ para seleccionar los documentos relevantes.
2. **Generator:** Se usa un modelo de generación $P(r|q,d)$ para producir la respuesta basada en la información recuperada.

Recuperación (Retriever)

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- En RAG, la recuperación de información suele realizarse con **métodos de búsqueda densa (Dense Retrieval)** basados en **Embeddings Vectoriales**.
- Dado un espacio de documentos preprocesados *D*, cada documento d_i se representa como un vector en un espacio de alta dimensión:

E(di)=Encoder(di)

- Luego, una consulta q también se codifica en el mismo espacio:

E(q)=Encoder(q)

Finalmente, la similitud entre la consu s se calcula mediante una métrica como:

$$\text{Sim}(E(q), E(d_i)) = \frac{E(q) \cdot E(d_i)}{\|E(q)\| \|E(d_i)\|}$$

Donde Sim es la similitud coseno. Los documentos con mayor puntuación se seleccionan como los más relevantes.

Métodos de Recuperación Usados en RAG:

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- BM25: Basado en búsqueda estadística tradicional (modelo probabilístico).
- Embeddings de Transformers: Modelos como SBERT, OpenAI Embeddings, Cohere, que generan representaciones vectoriales semánticamente ricas.
- Bases de Datos Vectoriales: FAISS, Weaviate, Pinecone, ChromaDB.

BM25 (Okapi BM25)

UNIVERSIDAD EAFIT

Escuela de
Ciencias Aplicadas e Ingeniería

BM25 (Okapi BM25)

Modelo probabilístico basado en frecuencia de términos:

$$\text{BM25}(q,d) = \sum_{t \in q} \log\left(\frac{N - n_t + 0.5}{n_t + 0.5} + 1\right) \frac{(k_1 + 1) f_{t,d}}{k_1(1 - b + b|d|/|\bar{d}|) + f_{t,d}}$$

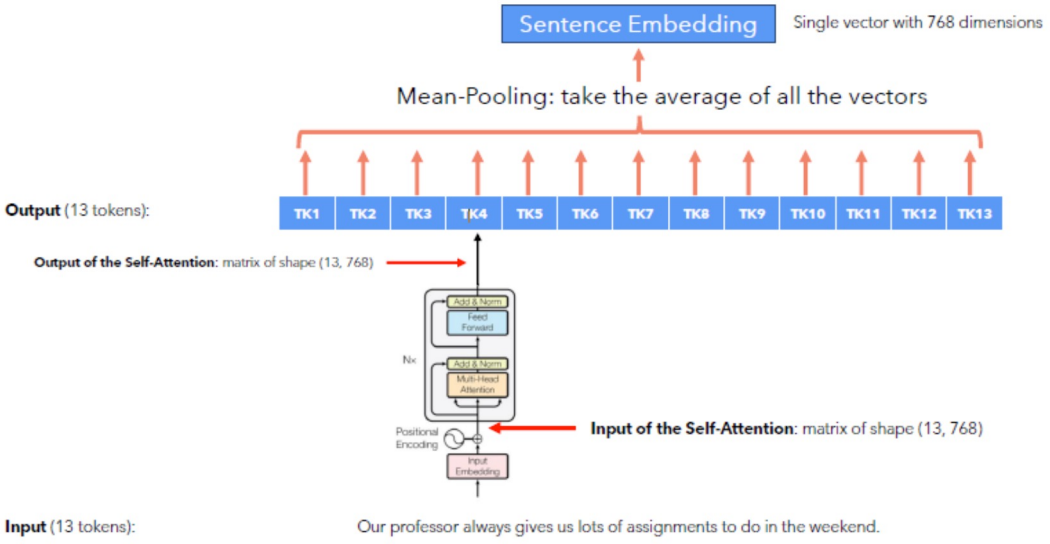
donde:

- $f_{t,d}$ es la frecuencia del término t en el documento d.
- n_t es el número de documentos que contienen t.
- N es el número total de documentos.
- $|d|$ es la longitud del documento y $|\bar{d}|$ es la longitud promedio en el corpus.
- k_1 y b son hiperparámetros.

Sentence Embeddings

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

- Podemos utilizar el mecanismo de Self-Attention para capturer el "significado" de una oración completa.
- Podemos usar un modelo BERT preentrenado para producer embeddings de oraciones completas.



Sentence Embeddings: comparación

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

- Podriamos usar similitud coseno:

$$\text{cosine similarity} = S_C(A, B) := \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

- Propuesta de Sentence BERT:

Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks

Nils Reimers and Iryna Gurevych
Ubiquitous Knowledge Processing Lab (UKP-TUDA)
Department of Computer Science, Technische Universität Darmstadt
www.ukp.tu-darmstadt.de

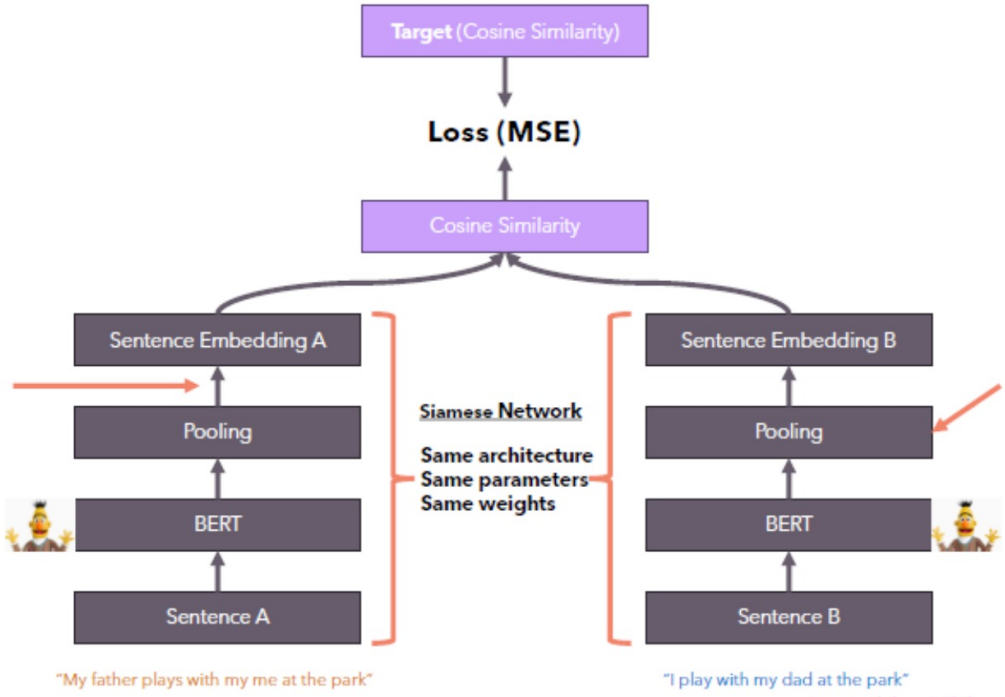
https://arxiv.org/abs/1908.10084

Sentence Embeddings: Arquitectura

UNIVERSIDAD EAFIT

Escuela de Ciencias Aplicadas e Ingeniería

Podemos aplicar una capa lineal para reducir el tamaño del vector de embeddings, por ejemplo, de 768 a 512.



Podemos usar mean-pooling, max-pooling o simplemente usar el token [cls] como embedding de oración..

Embeddings Semánticos (Dense Retrieval)

UNIVERSIDAD EAFIT

Escuela de **Ciencias Aplicadas e Ingeniería**

Embeddings Semánticos (Dense Retrieval) Representa documentos y consultas en un espacio de alta dimensión usando modelos de embeddings como **SBERT** o **OpenAI Embeddings**:

$$E(q) = f_\theta(q), \quad E(d) = f_\theta(d)$$

donde f_θ es un modelo de transformación basado en Transformers.

La similitud entre consulta y documento es evaluada mediante **coseno de similitud**:

$$\text{Sim}(E(q), E(d)) = \frac{E(q) \cdot E(d)}{\|E(q)\| \|E(d)\|}$$

Se almacenan en **bases de datos vectoriales** como **FAISS, Pinecone, Weaviate o ChromaDB**.

Generación (Generator)

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

- Después de recuperar los documentos, se utiliza un **modelo de lenguaje generativo** para producir la respuesta.
- Dado un conjunto de documentos relevantes D, el modelo de generación sigue la probabilidad condicional:

$$P(r|q,\mathrm{D}) = \prod_t P(r_t|r_{<t},q,\mathrm{D})$$

- Donde
  - $P(r_t|r_{<t},q,D)$ es la probabilidad de la palabra r_t en la respuesta en el tiempo t, dada la secuencia previa y la información recuperada.
- **Modelos Usados**:
  - GPT-4, LLaMA-2, Claude, Mistral, BLOOM.
  - Modelos autoregresivos basados en *Transformers*.

Técnicas para mejorar la generación

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

1. **Chain-of-Thought (CoT)**: Mejor contextualización con razonamiento explícito.
2. **Retrieval Re-Ranking**: Ajustar la ponderación de documentos para mejorar la coherencia.
3. **Fusion-in-Decoder (FiD)**: Modelos como T5-FID que procesan múltiples documentos simultáneamente.

Evaluación de Sistemas RAG

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

- Un sistema RAG debe evaluarse tanto en recuperación como en generación.
- Métricas de Recuperación
  - *¿Qué tan buenos son los documentos recuperados?*
  - **Recall@K**: Proporción de documentos relevantes entre los top-K resultados.

$$\text{Recall@K} = \frac{\sum_{i=1}^{K} 1(d_i \text{ es relevante})}{\text{total de documentos relevantes}}$$

  - **Mean Reciprocal Rank (MRR)**: Evalúa la posición del primer documento relevante.

$$MRR = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{\text{rank}_q}$$

Evaluación de Sistemas RAG

UNIVERSIDAD EAFIT | Escuela de Ciencias Aplicadas e Ingeniería

- Métricas de Generación
  - *¿Qué tan buena es la respuesta generada?*
  - **BLEU (Bilingual Evaluation Understudy)**: Comparación con una referencia humana.

$$BLEU = \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

  - **ROUGE (Recall-Oriented Understudy for Gisting Evaluation)**: Evalúa superposición de palabras con referencia.

$$\text{ROUGE-N} = \frac{\sum_{\text{superposición de n-gramas}}}{\sum_{\text{total de n-gramas en referencia}}}$$

  - **Perplexity (PPL)**: Evalúa la fluidez del modelo generativo.

$$PPL = e^{-\frac{1}{N}\sum_{i=1}^{N} \log P(w_i|w_{<i})}$$

Ejemplo RAG

UNIVERSIDAD EAFIT
Escuela de Ciencias Aplicadas e Ingeniería

- **Construcción de un sistema RAG con LangChain**
- **Requerimientos previos**: Python, OpenAI API, FAISS
- Construir un chatbot con RAG utilizando **LangChain + FAISS + OpenAI**

- Notebook lecture07