ComparativoModelos

November 11, 2025

1 Taller RAG - Comparativo modelos con SOLR y Milvus

Docente: **Luis Gabriel Moreno Sandoval**

Grupo Número 1:

- LUCENA ORJUELA, JULIAN
- MARTINEZ BERMUDEZ, JUAN
- MONTENEGRO MAFLA, MARIA
- REYES PALACIO, FELIPE

1.1 1. Carga de librerías y configuraciones generales

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import tabulate

# Configuración de estilo para los gráficos
sns.set_theme(style="whitegrid")
%matplotlib inline

ARCHIVO_RESULTADOS = "evaluation_results.csv"
```

1.2 2. Cargar los Resultados

Cargamos el archivo CSV que generó nuestro script de evaluación.

[2]:
```
# Cargar en un DataFrame
try:
    df = pd.read_csv(ARCHIVO_RESULTADOS)
    print(f"¡Datos de evaluación {ARCHIVO_RESULTADOS} cargados exitosamente!")
    print(f"Total de registros: {len(df)}")
    df.head()
except FileNotFoundError:
    print(f"Error: No se encontró el archivo en {file_path}")
    print("Asegúrate de que el notebook esté en la carpeta correcta.")
```

```
¡Datos de evaluación evaluation_results.csv cargados exitosamente!
Total de registros: 216
```

1.3 3. Calcular Métricas Promedio

Aquí calculamos la tabla de resumen. Agrupamos por backend y calculamos la media de cada métrica.

[5]:
```python
# Definir las métricas que queremos analizar
metric_columns = ['latency_sec', 'recall_at_k', 'mrr_at_k', 'rouge_l_f1']

# Agrupar por 'backend' y calcular la media
df_summary = df.groupby('backend')[metric_columns].mean()

print("--- Resumen de Métricas (Promedio) ---")
print(df_summary.to_markdown(floatfmt=".4f"))
```

```
--- Resumen de Métricas (Promedio) ---
| backend   |   latency_sec |   recall_at_k |   mrr_at_k |   rouge_l_f1 |
|:----------|--------------:|--------------:|-----------:|-------------:|
| milvus    |        4.2277 |        0.3406 |     0.4852 |       0.0827 |
| solr      |        3.6362 |        0.5807 |     0.6804 |       0.0826 |
```

¿Cómo Interpretar esta Tabla?

Esta tabla resume todo el análisis. Así es como se lee cada métrica:

- `latency_sec` **(Latencia):**
  - **Qué mide:** El tiempo promedio total (en segundos) desde que se envía la pregunta hasta que se recibe la respuesta generada.
  - **Mejor es: Más bajo**. Esto mide la velocidad del sistema.
- `recall_at_k` **(Recall@k):**
  - **Qué mide:** De todas las respuestas "correctas" (tus `relevant_chunk_ids`), ¿qué porcentaje fue *encontrado* por el buscador?
  - **Mejor es: Más alto**. Esta es la métrica de "exhaustividad". Mide qué tan bueno es el buscador para *encontrar* lo que debe.
- `mrr_at_k` **(Mean Reciprocal Rank):**
  - **Qué mide:** Qué tan alto en la lista (del 1 al `k`) apareció el *primer* documento correcto.
  - **Mejor es: Más alto** (un 1.0 perfecto significa que la respuesta correcta *siempre* fue el resultado #1). Mide la "precisión en la cima".
- `rouge_l_f1` **(ROUGE-L):**
  - **Qué mide:** La similitud entre la `generated_answer` (del LLM) y tu `ideal_answer` (del Gold Standard).

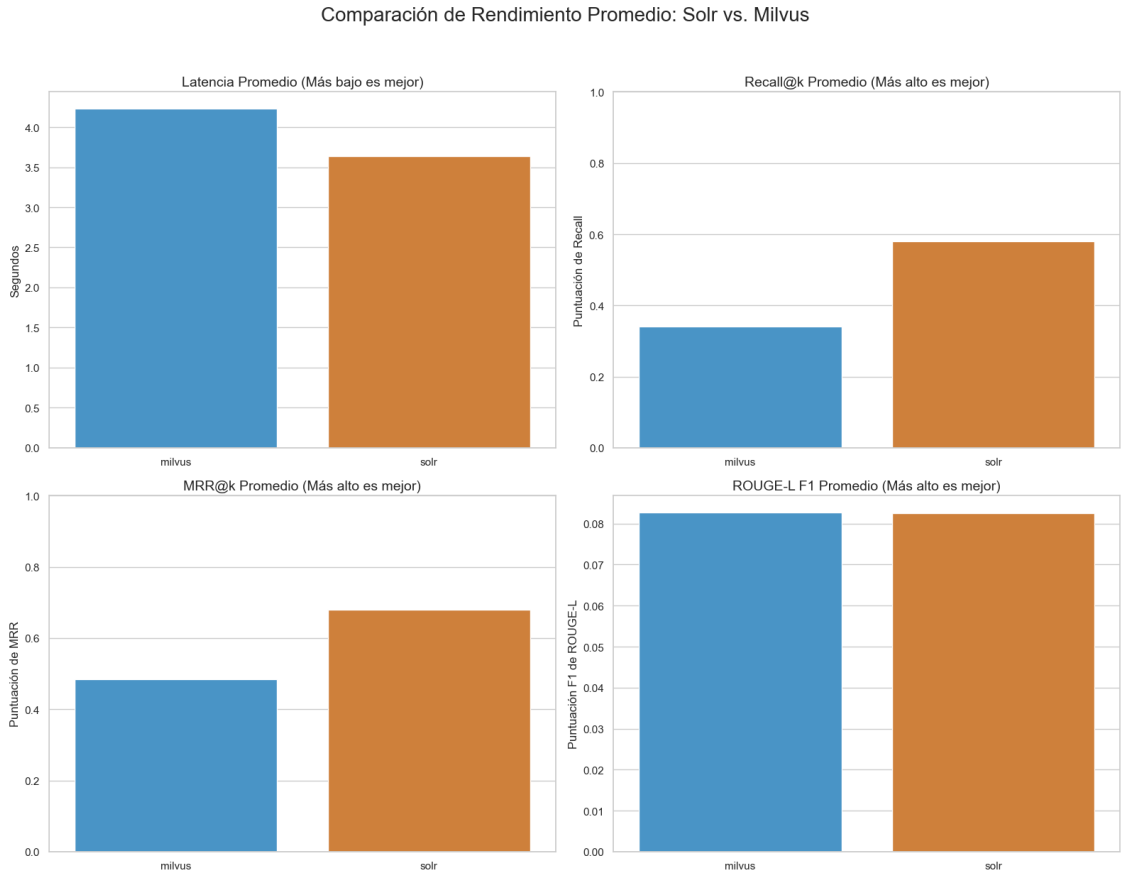
– **Mejor es: Más alto**. Esta es la métrica de "calidad final". Un ROUGE-L alto significa que el *contexto* que le pasó el buscador al LLM fue tan bueno que el LLM pudo generar una respuesta muy similar a la ideal.

1.4 5. Visualización Comparativa (Promedios)

Una tabla es buena, pero los gráficos son mejores. Se crea un gráfico de barras para cada una de las 4 métricas, comparando Solr y Milvus.

[10]:
```
# Creamos una figura con 4 sub-gráficos (2 filas, 2 columnas)
fig, axes = plt.subplots(2, 2, figsize=(16, 12))
fig.suptitle('Comparación de Rendimiento Promedio: Solr vs. Milvus',␣
 ↪fontsize=20, y=1.03)

# Paleta de colores
palette = {"solr": "#E67E22", "milvus": "#3498DB"}
hue_order = df_summary.index # Para asegurar el orden

# 1. Latencia
# --- CAMBIO AQUÍ: Añadido hue= y legend=False ---
sns.barplot(data=df_summary, x=df_summary.index, y='latency_sec', ax=axes[0,␣
 ↪0], palette=palette,
            hue=hue_order, legend=False)
axes[0, 0].set_title('Latencia Promedio (Más bajo es mejor)', fontsize=14)
axes[0, 0].set_ylabel('Segundos')
axes[0, 0].set_xlabel('')

# 2. Recall@k
# --- CAMBIO AQUÍ: Añadido hue= y legend=False ---
sns.barplot(data=df_summary, x=df_summary.index, y='recall_at_k', ax=axes[0,␣
 ↪1], palette=palette,
            hue=hue_order, legend=False)
axes[0, 1].set_title('Recall@k Promedio (Más alto es mejor)', fontsize=14)
axes[0, 1].set_ylabel('Puntuación de Recall')
axes[0, 1].set_xlabel('')
axes[0, 1].set_ylim(0, 1) # Recall es de 0 a 1

# 3. MRR@k
# --- CAMBIO AQUÍ: Añadido hue= y legend=False ---
sns.barplot(data=df_summary, x=df_summary.index, y='mrr_at_k', ax=axes[1, 0],␣
 ↪palette=palette,
            hue=hue_order, legend=False)
axes[1, 0].set_title('MRR@k Promedio (Más alto es mejor)', fontsize=14)
axes[1, 0].set_ylabel('Puntuación de MRR')
axes[1, 0].set_xlabel('')
axes[1, 0].set_ylim(0, 1) # MRR es de 0 a 1
```

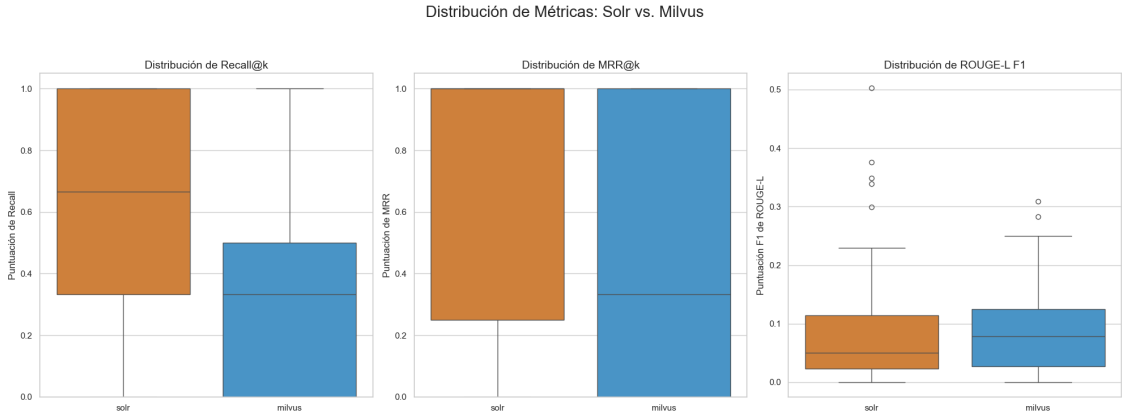
```
# 4. ROUGE-L
# --- CAMBIO AQUÍ: Añadido hue= y legend=False ---
sns.barplot(data=df_summary, x=df_summary.index, y='rouge_l_f1', ax=axes[1, 1],␣
 ↪palette=palette,
            hue=hue_order, legend=False)
axes[1, 1].set_title('ROUGE-L F1 Promedio (Más alto es mejor)', fontsize=14)
axes[1, 1].set_ylabel('Puntuación F1 de ROUGE-L')
axes[1, 1].set_xlabel('')

# Ajustar el layout y guardar la imagen
plt.tight_layout()
plt.show()
```



1.5 6. Análisis de Distribución

Los promedios pueden ser engañosos. ¿Es un backend consistentemente bueno, o solo tuvo suerte en algunas preguntas? Un boxplot (diagrama de caja) nos muestra la distribución de los resultados.

[9]:
```
# Creamos una figura con 3 sub-gráficos
fig, axes = plt.subplots(1, 3, figsize=(20, 7))
fig.suptitle('Distribución de Métricas: Solr vs. Milvus', fontsize=20, y=1.05)

palette = {"solr": "#E67E22", "milvus": "#3498DB"}
hue_order = ['solr', 'milvus'] # Para asegurar el orden

# 1. Distribución de Recall@k
# --- CAMBIO AQUÍ: Añadido hue= y legend=False ---
sns.boxplot(data=df, x='backend', y='recall_at_k', ax=axes[0], palette=palette,
            hue='backend', legend=False, order=hue_order)
axes[0].set_title('Distribución de Recall@k', fontsize=14)
axes[0].set_ylabel('Puntuación de Recall')
axes[0].set_xlabel('')
axes[0].set_ylim(0, 1.05)

# 2. Distribución de MRR@k
# --- CAMBIO AQUÍ: Añadido hue= y legend=False ---
sns.boxplot(data=df, x='backend', y='mrr_at_k', ax=axes[1], palette=palette,
            hue='backend', legend=False, order=hue_order)
axes[1].set_title('Distribución de MRR@k', fontsize=14)
axes[1].set_ylabel('Puntuación de MRR')
axes[1].set_xlabel('')
axes[1].set_ylim(0, 1.05)

# 3. Distribución de ROUGE-L
# --- CAMBIO AQUÍ: Añadido hue= y legend=False ---
sns.boxplot(data=df, x='backend', y='rouge_l_f1', ax=axes[2], palette=palette,
            hue='backend', legend=False, order=hue_order)
axes[2].set_title('Distribución de ROUGE-L F1', fontsize=14)
axes[2].set_ylabel('Puntuación F1 de ROUGE-L')
axes[2].set_xlabel('')

# Guardar y mostrar
plt.tight_layout()
plt.show()
```


Distribución de Métricas: Solr vs. Milvus
Distribución de Recall@k
Puntuación de Recall
1.0
0.8
0.6
0.4
0.2
0.0
solr
milvus
Distribución de MRR@k
Puntuación de MRR
1.0
0.8
0.6
0.4
0.2
0.0
solr
milvus
Distribución de ROUGE-L F1
Puntuación F1 de ROUGE-L
0.5
0.4
0.3
0.2
0.1
0.0
solr
milvus

[ ]: