UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE MECÁNICA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

Sistema de Recomendación

Link: https://casoestudio.arnoldfloresdev.com/

Colab-pdf: https://casoestudio.arnoldfloresdev.com/data/colab.pdf

Codigo fuente: https://github.com/Arkds/Sistema-de-recomendacion

Colab: https://colab.research.google.com/drive/1xUtbPZOdwPrT-ivnnJmOja_HbSlWmMqh?usp=sharing

ESTUDIANTE: Flores Andia Arnold Kevin

DOCENTE: CHUCUYA TIJUTANI MARCO ANTONIO

CURSO: **SISTEMAS DE RECOMENDACIÓN**

NIVEL IV – SEMESTRE 2025 - I

PUNO – PERÚ

Título: Sistema de recomendación de películas basado en contenido (TMDB-5000)

Resumen.
Se implementó un sistema de recomendación de películas de enfoque **content-based** que, a partir de la descripción y metadatos de cada película, sugiere títulos similares. El objetivo fue obtener un modelo reproducible y operativo en un **hosting compartido** sin necesidad de servicios de backend. El pipeline integra preprocesamiento de texto, vectorización con *bag-of-words* y una métrica de similitud coseno; los resultados se exportan a archivos JSON consumidos por una página web estática.

Datos y variables.
Se utilizaron los archivos `tmdb_5000_movies.csv` y `tmdb_5000_credits.csv` del dataset TMDB-5000. A nivel de variables, se emplearon: (i) sinopsis (*overview*), (ii) géneros, (iii) palabras clave (*keywords*), (iv) reparto principal (tres actores), y (v) director/a. Estas señales textuales se transformaron en un conjunto de **tags** por película que resumen su contenido.

Metodología.

1. **Unificación y limpieza:** unión de ambas tablas por título; manejo de nulos; parseo seguro de estructuras JSON (listas de diccionarios); normalización (minúsculas y eliminación de espacios internos para tokens compuestos).

2. **Ingeniería de características:** construcción de `tags` concatenando *overview*, géneros, keywords, cast (top-3) y director/a.

3. **Vectorización y similitud:** representación *bag-of-words* mediante `CountVectorizer(max_features=5000, stop_words='english')` y cálculo de la matriz de **similitud del coseno**.

4. **Generación de recomendaciones:** para cada película se ordenan las demás por similitud y se guardan las **Top-K** (K=20) en una estructura compacta de consulta inmediata.

Arquitectura de despliegue.
Para posibilitar su uso en hosting compartido, el sistema exporta dos artefactos: (a) `movies.json` con `{id, title, slug}` y (b) `recs_top20.json` con, para cada `id`, la lista ordenada de similares y su puntuación. Un **front estático** (HTML+CSS+JS) carga estos archivos y resuelve las consultas en el navegador. Opcionalmente, el front solicita los **posters** a la API pública de TMDB usando una API key del cliente, almacenando las URLs en `localStorage` para reducir latencia y consumo.

Resultados.
El sistema entrega recomendaciones coherentes: secuelas, precuelas y películas de temática/estilo afín aparecen entre las primeras posiciones. El tamaño de los artefactos (~300 KB para `movies.json` y ~5 MB para `recs_top20.json`) permite una carga

aceptable en contextos web típicos, manteniendo tiempo de respuesta inmediato (O(1) por consulta, vía tabla precomputada).

Limitaciones.
El enfoque content-based puede sesgarse hacia "más de lo mismo" porque no incorpora preferencias históricas de usuarios. La cobertura de *posters* depende de la disponibilidad de TMDB y de la API key. La vectorización *bag-of-words* no captura relaciones semánticas profundas ni sinónimos.

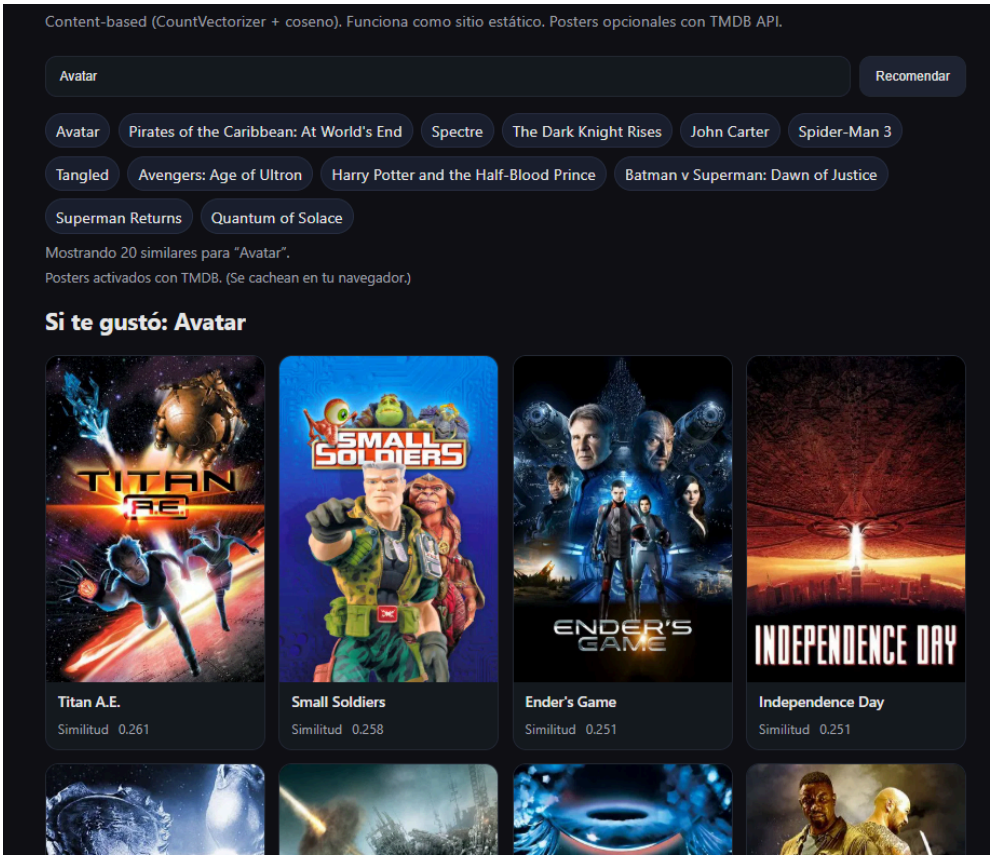
Guía de implementación.

- **Construcción del modelo** en cuaderno: descargar dataset, preparar `tags`, vectorizar y calcular similitudes; exportar `movies.json` y `recs_top20.json`.

```
[5]
✓ 0s
import os, ast, json, difflib, pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from urllib.request import urlretrieve

os.makedirs("/content/data", exist_ok=True)
MOVIES_URL  = "https://raw.githubusercontent.com/rajeevratan84/dataset-repo/master/tmdb_5000_movies.csv"
CREDITS_URL = "https://raw.githubusercontent.com/rajeevratan84/dataset-repo/master/tmdb_5000_credits.csv"
movies_path  = "/content/tmdb_5000_movies.csv"
credits_path = "/content/tmdb_5000_credits.csv"

for url, path in [(MOVIES_URL, movies_path),(CREDITS_URL, credits_path)]:
    if not os.path.exists(path):
        urlretrieve(url, path)

movies  = pd.read_csv(movies_path)
credits = pd.read_csv(credits_path)

movies = movies[['id','title','overview','genres','keywords']].copy()
credits = credits[['title','cast','crew']].copy()
df = movies.merge(credits, on='title', how='inner')
for col in ['overview','genres','keywords','cast','crew']:
    df[col] = df[col].fillna('[]')
df['overview'] = df['overview'].replace('[]','').fillna('')
len(df), df.head(2)
```

- **Despliegue web:** subir `index.html` y los JSON a `…/recommender/` y `…/recommender/data/` respectivamente.

- **Posters (opcional):** configurar API key de TMDB en el `index.html`.



- **Enlaces:**

  Página estática: https://casoestudio.arnoldfloresdev.com/

  Colab informe: https://casoestudio.arnoldfloresdev.com/data/colab.pdf

  Colab: https://colab.research.google.com/drive/1xUtbPZOdwPrT-ivnnJmOja_HbSlWmMqh?usp=sharing
  Código fuente git: https://github.com/Arkds/Sistema-de-recomendacion

Conclusión.

Se logró un recomendador funcional, portable y de bajo costo operativo, adecuado para demostraciones académicas y prototipos productivos en entornos con restricciones de infraestructura (hosting compartido). El diseño modular (artefactos JSON + front estático) facilita su mantenimiento y escalado hacia variantes híbridas con enriquecimiento semántico o señales de usuario.