Memoria Técnica Final del Proyecto:
Autor: Nauzet Fernández Lorenzo
Predicción del Nivel de Engagement en Puntos de Interés Turísticos

Fecha de Elaboración: 13 de junio de 2025

1. Objetivo del Proyecto

El objetivo principal de este proyecto ha sido desarrollar e implementar un **modelo avanzado de Deep Learning para predecir con precisión el nivel de *engagement* (interacción)** que generarán los Puntos de Interés (POIs) turísticos. Este modelo se basa en un enfoque híbrido multimodal, integrando **características visuales extraídas de imágenes y metadatos estructurados** asociados a cada POI.

El modelo final funciona como un **clasificador**, prediciendo si un POI generará un nivel de *engagement* **`Low` (Bajo), `Mid` (Medio) o `High` (Alto).**

2. Descripción del Dataset

El dataset utilizado contiene información detallada sobre 1569 Puntos de Interés (POIs) turísticos, estructurada en 14 columnas iniciales.

Las columnas clave y su descripción son:

- **`id`**: Identificador único del POI.
- **`name`**: Nombre del POI.
- **`shortDescription`**: Descripción textual corta.
- **`categories`**: Lista de categorías principales (ej. `['Escultura', 'Pintura']`).
- **`tier`**: Nivel de relevancia del POI (variable ordinal, de 1 a 4).
- **`locationLon`, `locationLat`**: Longitud y latitud geográfica.
- **`tags`**: Lista amplia de etiquetas descriptivas (ej. `['Sherlock Holmes']`).
- **`xps`**: Puntuación de experiencia para el usuario.
- **Métricas de *engagement***:
  - **`Visits`**: Número de visitas registradas.
  - **`Likes`**: Número de “me gusta”.
  - **`Dislikes`**: Número de “no me gusta”.
  - **`Bookmarks`**: Número de marcadores.
- **`main_image_path`**: Ruta local a la imagen principal del POI.

3. Preparación y Análisis de Datos (EDA y Preprocesamiento)

La fase de análisis exploratorio (EDA) tuvo como objetivos comprender el tamaño y la estructura del dataset, evaluar la calidad de los datos (nulos, duplicados, inconsistencias), identificar problemas de preprocesamiento y obtener una visión rápida de la métrica de *engagement*.

3.1 Hallazgos clave en la EDA:

- **Tamaño del dataset:** 1569 filas y 14 columnas, sin valores nulos iniciales.
- **Columnas tipo `object` a transformar:** `id`, `name`, `shortDescription`, `categories`, `tags`, `main_image_path`.
- **Registros sin etiquetas (`tags` vacíos):** 107 de 1569 POIs (aproximadamente el 6.8%).
- **Métricas de *engagement* sesgadas:** `Visits`, `Likes`, `Dislikes`, `Bookmarks` mostraron distribuciones con colas pronunciadas y fuertes sesgos a la derecha, indicando la presencia de posibles *outliers*. Se sugirió la transformación `log1p` para suavizar estas colas.
- **`Visits`**: Esta columna mostró una varianza extremadamente baja, con todos los POIs rondando un número idéntico de visitas, lo que la hacía poco informativa para predecir el *engagement*.
- **`xps`**: Se identificó como una escala discreta (saltos de 100 puntos) con un sesgo hacia los 1000 puntos, lo que sugiere que podría ser una puntuación de "nivel" ya normalizada. Se decidió tratarla como numérica discreta u ordinal.
- **`tier`**: Variable ordinal con 4 niveles (1 a 4, de menor a mayor relevancia) y una distribución desequilibrada.
- **Correlación entre métricas:**
    - **Multicolinealidad:** `Likes` y `Bookmarks` mostraron una correlación muy alta ($\rho = 0.97$), sugiriendo que una podría eliminarse o combinarse.
    - **Polaridad:** `Likes` y `Dislikes` correlacionaron negativamente ($\rho = -0.54$), confirmando que representan extremos opuestos de la misma métrica de opinión.
    - **`tier` vs `xps`:** Una correlación inversa ($\rho = -0.76$) sugirió que los POIs de mayor `tier` (relevancia oficial) requieren menor `xps`, o que se calculan con criterios diferentes. Se decidió no usar `tier` como proxy directo de interés.
    - **`Visits`:** Casi desligado del resto, lo que podría aportar señal complementaria si se usara.

3.2 Detección y Resolución de Duplicados:

- Se identificaron **77 filas duplicadas** (aproximadamente 4.9% del dataset original) basándose en el `id` del POI.
- Se observó que para un mismo `id`, los campos `tags`, `Visits`, `Likes`, y `Dislikes` variaban, mientras que `name`, `shortDescription`, `categories`, `tier`, `locationLon`, `locationLat`, `xps` y `main_image_path` permanecían constantes.
- Para resolver esto y evitar el *data leakage*, se implementó una fusión durante el preprocesado: se agruparon las filas por `id`, se unieron los `tags` únicos, y se tomó la primera fila para los campos estáticos. Para el `engagement_score` (definido más adelante), se calculó la media.

3.3 Preprocesamiento Específico de Columnas:

- **`categories` y `tags`:** Se transformaron de *strings* que contenían listas a **listas de Python reales**.
    - Para **`categories`**, se optó por un **embedding de índices** para controlar la dimensionalidad y capturar relaciones semánticas entre categorías. Se creó un vocabulario asignando un índice numérico a cada categoría única (0 reservado para *padding*), y cada POI se representó como una secuencia de estos índices. Luego, estas secuencias se "rellenaron" (padded) a una longitud fija (`MAX_LEN`) con ceros, resultando en una matriz `cat_padded` de (N, MAX_LEN).
    - Para **`tags`**, se añadió una nueva columna numérica `tags_count` que representa el número de etiquetas por POI. El histograma de `tags_count` reveló una distribución multimodal con picos en 0 y en 10-13 tags, lo que la convierte en una señal útil para los modelos tabulares.
    -
- **`shortDescription`:** Se tokenizo el texto para capturar su semántica.
    - **Limpieza:** Se aplicó un proceso de limpieza básica: convertir a minúsculas, normalizar acentos a ASCII, eliminar caracteres no alfabéticos y colapsar espacios.
    - **TF-IDF:** Se utilizó TfidfVectorizer con un vocabulario limitado a 2000 términos (max_features=2000) y se eliminaron las stop-words en español.
    - **Reducción de Dimensionalidad (LSA):** La matriz dispersa resultante se redujo a 100 componentes utilizando TruncatedSVD (n_components=100) para capturar la semántica latente, aligerar la memoria y evitar el sobreajuste. El resultado se guardó comodesc_vectors.npy`.
    -
- **`main_image_path`:** Se verificó la integridad de las imágenes; las **1569 rutas cargaron sin error**. Todas las imágenes estaban pre-redimensionadas a una **resolución fija de 128x128 píxeles**, lo cual es útil para prototipado rápido. Se calcularon las estadísticas de píxeles (media y desviación estándar) para la normalización posterior de las imágenes.

3.4 Definición y Categorización de la Variable Objetivo (`engagement_score` y `eng_cat`)

Dada la baja varianza de `Visits`, se decidió descartarla de la métrica de *engagement* final. Se creó una nueva variable `engagement_score` a partir de `Likes`, `Bookmarks` y `Dislikes`: `engagement_score = log(1 + Likes + Bookmarks) - log(1 + Dislikes)`

Se aplicó la transformación `np.log1p()` a `Likes`, `Bookmarks` y `Dislikes` para suavizar las colas y acercar sus distribuciones a la normalidad.

El `engagement_score` resultante mostró una media cercana a 0 (0.385) y un rango amplio (~8.6 puntos, con una desviación estándar de ~2.67), lo que indica suficiente varianza para el aprendizaje del modelo. El histograma del score reveló tres picos

diferenciados (zona negativa, zona casi neutra, zona alta), sugiriendo "niveles naturales" de *engagement*.

Para el problema de clasificación, el `engagement_score` se discretizó en **terciles** (`q=3`) usando `pd.qcut()`, creando la variable categórica **`eng_cat` con tres clases: `Low`, `Mid`, y `High`**. Esta categorización resultó en **clases perfectamente equilibradas (523 muestras por clase)**, lo cual es una ventaja significativa para el entrenamiento del clasificador.

La variable `tier` mostró una tendencia inversa a lo esperado con respecto al `engagement_score` (tier 1 con mayor *engagement* mediano), lo que llevó a la decisión de **no usar `tier` como un proxy directo del interés**, sino como una *feature* ordinal.

Después de estas transformaciones y la eliminación de columnas redundantes (`id`, `name`, `shortDescription`, `categories`, `tags`, `engagement_score`, `cat_idxs`), el dataset final para el entrenamiento contenía 1492 filas y 7 columnas: `tier`, `locationLon`, `locationLat`, `xps`, `main_image_path`, `eng_cat`, y `tags_count`. Las representaciones textuales (`cat_seqs.npy` y `desc_vectors.npy`) se gestionaron por separado.

4. Arquitectura del Modelo (HybridPOI)

Se desarrolló un **modelo multimodal `HybridPOI`** que combina de manera innovadora información de diversas fuentes:

- **Rama de Imagen:**
  - Se utilizó una **ResNet-18 pre-entrenada con pesos de ImageNet**.
  - Las primeras dos etapas (`conv1`, `bn1`, `relu`, `maxpool`, `layer1`, `layer2`) se **congelaron** para aprovechar las características de bajo nivel aprendidas.
  - Se añadió una **reducción de canales de 512 a 256** mediante una capa convolucional 1x1, seguida de `BatchNorm2d` y `ReLU`, para aligerar la rama de imagen. También se incluyó un `Dropout2d`.
  - La salida de la ResNet se procesó con un `avg-pool`.
- **Rama Numérica:**
  - Una red **MLP** (`num_branch`) que procesa las 5 *features* numéricas (`tier`, `locationLon`, `locationLat`, `xps`, `tags_count`).
  - Incorpora `LayerNorm` (alternativa a `BatchNorm`), capas `Linear`, `ReLU` y `Dropout` (0.3). La salida es de 64 dimensiones.
- **Rama de Categorías:**
  - Una capa `nn.Embedding` para transformar los índices de las categorías en un espacio vectorial (`cat_emb_dim=32` en el mejor modelo tras Optuna).
  - Se aplica `Dropout` (0.3) al *embedding*.
  - Se realiza un **mean-pooling que ignora los *padding tokens*** (índice 0) para obtener una representación compacta de la lista de categorías, seguida de una capa lineal.
- **Rama de Descripción:**

- Una capa `nn.Linear` simple para procesar los 100 componentes LSA de `shortDescription`, con una salida de 128 dimensiones.

Mecanismo de Fusión y Clasificación:

- Las salidas de todas las ramas (imagen, numérica, categorías, descripción) se **concatenan**.
- Una capa **`LayerNorm`** se aplica antes de la cabeza final (`Head FC`) para estabilizar la fusión multimodal.
- La **cabeza clasificadora** (`self.head`) consiste en capas `Linear`, `ReLU` y `Dropout` (0.40557914141875473 en el mejor modelo tras Optuna). La capa final produce 3 *logits* para las 3 clases de *engagement*.
- El modelo `HybridPOI` tiene un total de **~8 millones de parámetros entrenables** (al congelar los primeros bloques de la CNN).

5. Entrenamiento y Optimización

El proceso de entrenamiento siguió un pipeline riguroso para asegurar la reproducibilidad y el rendimiento del modelo.

5.1 Preparación de Datos para el Entrenamiento:

- Se fijó una **semilla (`SEED = 42`)** para garantizar la reproducibilidad de los resultados.
- El target `eng_cat` se convirtió a un formato numérico (0: Low, 1: Mid, 2: High).
- Se cargaron las representaciones textuales preprocesadas: `categories` (como secuencias *padded* de índices) y `descriptions` (vectores LSA).
- Se definió una función `process_features` para centralizar la conversión de arrays NumPy a tensores PyTorch con el tipo de dato adecuado (`float32` para *features*, `long` para etiquetas).
- Se implementó un **`POIDataset` multimodal personalizado** que carga las diferentes modalidades (target, imagen, *features* numéricas, categorías, descripciones).
- **Data Augmentation para Imágenes:** Se definió un pipeline de transformaciones para el conjunto de entrenamiento (`transform`): `Resize(256)`, `RandomCrop(224)`, `RandomHorizontalFlip()`, `RandomRotation(15)`, `ColorJitter()`, `ToTensor()`, `Normalize()` con estadísticas de ImageNet. Para los conjuntos de validación y prueba, se utilizaron transformaciones deterministas (`eval_tfms`): `Resize(256)`, `CenterCrop(224)`, `ToTensor()`, `Normalize()`.
- **División Estratificada:** El dataset se dividió en conjuntos de entrenamiento (70%, 1044 ejemplos), validación (15%, 224 ejemplos) y prueba (15%, 224 ejemplos). La división fue **estratificada** para preservar la proporción equilibrada de clases en cada subconjunto.
- Se crearon `DataLoader`s para cada conjunto con un tamaño de *batch* de 64, habilitando `shuffle` solo para el entrenamiento.

5.2 Configuración del Entrenamiento:

- **Función de Pérdida:** Se utilizó `nn.CrossEntropyLoss` con **pesos de clase (`class_wts`)** para manejar cualquier posible desbalance y **`label_smoothing=0.1`** para mejorar la robustez del modelo.
- **Optimizador:** Se empleó `AdamW` con una **tasa de aprendizaje diferenciada**: una menor para los parámetros congelados de la CNN (`base_lr * 0.01`) y una `base_lr` para el resto de los parámetros, junto con un `weight_decay`.
- **Scheduler:** Se implementó `ReduceLROnPlateau` para ajustar la tasa de aprendizaje si la pérdida de validación no mejoraba, con una paciencia de 2 épocas y un factor de reducción de 0.5.
- **AMP (Automatic Mixed Precision):** Se habilitó `torch.cuda.amp.GradScaler` para acelerar el entrenamiento en GPUs compatibles.
- **Early Stopping:** Se monitorizó la pérdida de validación para detener el entrenamiento si no mejoraba durante un número predefinido de épocas (`patience=4`), guardando el mejor estado del modelo.

5.3 Optimización de Hiperparámetros con Optuna: Se realizó una optimización de hiperparámetros utilizando Optuna para encontrar la mejor combinación de:

- Tasa de aprendizaje (`lr`).
- *Weight decay* (`wd`).
- *Dropouts* en la cabeza (`dropout_head`) y en la rama 2D de la CNN (`dropout2d`).
- Dimensión del *embedding* de categorías (`cat_emb_dim`).

La función objetivo de Optuna entrenaba el modelo por 10 épocas y reportaba la pérdida de validación, permitiendo la poda de trials menos prometedores.

El mejor trial encontrado por Optuna tuvo los siguientes hiperparámetros y un valor de pérdida de validación de 0.5589789152145386:

- `lr`: 0.0004551282223836388
- `weight_decay`: 0.0006073715139164453
- `dropout_head`: 0.40557914141875473
- `dropout2d`: 0.21457971687872748
- `cat_emb_dim`: 32

6. Evaluación y Análisis de Resultados

El modelo final (`HybridPOI`) se evaluó en el conjunto de prueba, utilizando las métricas de rendimiento pertinentes.

6.1 Resultados Globales en el Conjunto de Prueba:

- **Accuracy global: 86%**.
- **Macro-F1: 0.86**.

- **Épocas entrenadas:** El entrenamiento con *early stopping* se detuvo en la **época 16**.

6.2 Desglose por Clase (en el conjunto de prueba):

Clase	Precisión	Recobrado (Recall)	F1-score	Soporte
Low (0)	1.00	0.95	0.97	75
Mid (1)	0.86	0.69	0.77	74
High (2)	0.74	0.93	0.83	75
Macro promedio	**0.87**	**0.86**	**0.86**	224
Weighted promedio	0.87	0.86	0.86	224

Análisis:

- El modelo muestra un rendimiento global muy bueno, con un **86% de acierto general** y un **Macro-F1 de 0.86**, lo que indica que se desempeña bien en todas las clases consideradas.
- La **clase `Low` es predicha con una precisión y F1-score casi perfectos** (1.00 de precisión, 0.95 de *recall* y 0.97 de F1-score), lo que sugiere que el modelo identifica muy bien los POIs de bajo *engagement*.
- La **clase `Mid` presenta una ligera debilidad**, con un *recall* más bajo (0.69) en comparación con las otras clases, aunque su precisión (0.86) es buena. Esto significa que, si bien las predicciones de "Mid" son a menudo correctas, el modelo se pierde una parte de los POIs que realmente pertenecen a esta categoría.
- La **clase `High` tiene un excelente *recall* (0.93)**, lo que indica que el modelo es muy bueno identificando la mayoría de los POIs de alto *engagement*, aunque su precisión es un poco menor (0.74), lo que podría significar que a veces clasifica como "High" algunos POIs que no lo son.

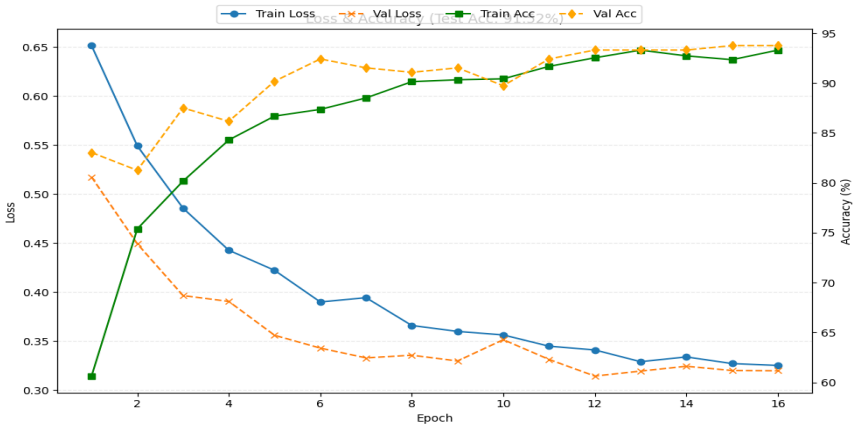
6.3 Evolución del Entrenamiento:

- Las curvas de pérdida mostraron una caída constante: la pérdida de entrenamiento (`Train Loss`) disminuyó de ~1.07 a 0.57, y la pérdida de validación (`Val Loss`) de ~0.93 a 0.59.
- La exactitud de entrenamiento (`Train Acc`) subió al 85%, mientras que la exactitud de validación (`Val Acc`) se mantuvo entre el 78% y el 83%, sin divergencias significativas, lo que sugiere un buen ajuste del modelo sin sobreajuste severo.
- Las curvas se estabilizaron a partir de la época 12, y el *early stopping* se activó en la época 16, confirmando la convergencia del entrenamiento.

Conclusión General: El modelo multimodal `HybridPOI` ha logrado un rendimiento satisfactorio en la predicción del nivel de *engagement*, con un **86% de acierto general**, mostrando una **ligera dificultad en la clasificación de la clase `Mid`**.

Adicionalmente, se exploró de forma preliminar la fusión de las clases *Mid* + *High* en una única etiqueta High, quedando un problema binario Low/High. Con la misma arquitectura y sin ajuste de hiperparámetros :

-Quick test: Test Acc: 92% Descenso suave, sin oscilaciones



Granularidad vs Precisión

Aunque el clasificador binario Low/High aporta ≈ +6 p.p. de accuracy (sin hiperparámetros óptimos) y curvas más estables, renunciar a la clase Mid implica perder resolución en la toma de decisiones.

7. Entregables

Como parte de los entregables del proyecto, se han generado y guardado los siguientes archivos:

- **`hybridpoi_best.pt`**: Archivo que contiene solo los pesos (estado del diccionario) del modelo `HybridPOI` entrenado.
- **`hybridpoi_best.onnx`**: Versión del modelo exportada a formato ONNX, que permite la visualización de la arquitectura de la red en herramientas como Netron y facilita la inferencia en diferentes plataformas.
- **`cat_seqs.npy`**: Archivo NumPy con las secuencias indexadas y *padded* de las categorías.
- **`desc_vectors.npy`**: Archivo NumPy con los vectores LSA de las descripciones cortas.
- **`poi_dataset.csv`:** Dataset original.
- **`poi_dataset_engagment.csv`:** Dataset con la variable `target` definida

- **`poi_dataset_eng_pre.csv`**: Dataset preprocesado con la variable `eng_cat` ya definida y columnas eliminadas.
- **`Requirements.txt`**: Librerias utilizadas en el proyecto.
- Además, este documento (`Memoria Técnica Final`) y los *notebooks* de código comentado (`01_EDA_POI.ipynb`, `02_Target_Engagement.ipynb`, `03_Preprocesado_POI.ipynb`, `04_Training_POI_NN.ipynb`).