

# Tarea 1 - Problem Set

Christofer Palomino Navarrete<sup>1</sup> and José Luis Gajardo Angel<sup>1</sup>

<sup>1</sup>Universidad Diego Portales , Facultad de Administración y Economía / Facultad de Ingeniería y Ciencias

30 de enero de 2026

## 1. Procesamiento de Lenguaje Natural (NLP) y Embeddings

### Análisis de la Data

#### Gestión de Valores Nulos (NaN)

En el archivo, algunas reseñas estaban vacías.

Se convirtieron los valores NaN en strings vacíos y luego se eliminaron esas filas del análisis ya que si le pasamos un None o NaN a un modelo de BERT o Word2Vec, el código lanzará un error porque estos modelos esperan texto.

#### Eliminación de “Ruido” con Regex

El dataset tenía símbolos como @#&\* al final de algunas frases (ej. fila 0 y 7). Usamos el patrón `[^a-zA-Záéíóúñ]` para borrar todo lo que NO sea una letra (incluyendo tildes y ñ) o un espacio”. Esto se debe a que para un modelo como Word2Vec, los símbolos # o & son “palabras” diferentes. Si no los quitamos, el modelo intentaría aprender el significado de @, lo cual no aporta valor semántico al análisis de retail.

#### Normalización (Lowercasing)

Convertimos todo el texto a minúsculas ya que para la computadora, “Excelente”, “EXCELENTE” y “excelente” son tres vectores distintos. Al normalizarlos, nos aseguramos de que el modelo entienda que se refieren al mismo concepto, agrupando toda su “fuerza” semántica en una sola palabra.

#### Tokenización (División en Palabras)

Fragmentamos las oraciones en listas de palabras individuales (ej: [‘el’, ‘producto’, ‘llegó’, ‘rápido’]). Word2Vec no lee oraciones completas de golpe; aprende analizando la “vecindad”

de cada palabra. Necesita el texto segmentado para deslizar su ventana de contexto.

Tabla 1: Pipeline de Preprocesamiento de NLP

Etapa del Proceso	Resultado del Texto
Original	<i>Atención al cliente pésima. Producto de mala calidad. #@#&amp;* </i>
Limpieza de Símbolos	Atención al cliente pésima Producto de mala calidad
Minúsculas y Espacios	atención al cliente pésima producto de mala calidad
Tokenizado (W2V)	['atención', 'al', 'cliente', 'pésima', 'producto', 'de', 'mala', 'calidad']

**1.1.a. Encuentra los 5 términos más similares semánticamente a la palabra “defectuoso” y “rápido” utilizando la similitud coseno**

Los términos más similares obtenidos por el modelo Word2Vec fueron:

**Similares a 'defectuoso':**

- insatisfecho: 0.9882
- una: 0.9869
- total: 0.9857
- sucio: 0.9846
- decepción: 0.9817

**Similares a 'rápido':**

- recomendado: 0.9851
- llegó: 0.9828
- estado: 0.9819
- eficaz: 0.9811
- y: 0.9784

Figura 1: Visualización términos similares.

**1.1.b. Interpretación Matemática: Explica por qué utilizamos la Similitud Coseno...**

Se prefiere usar la similitud de coseno porque maneja de mejor manera el impacto de la magnitud de los vectores (palabras muy comunes en el corpus). El producto punto está muy condicionado por la magnitud de los vectores, por lo que palabras muy repetidas van a ser similares a muchas otras palabras solo por su peso y no por su significado.

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

La Similitud Coseno corrige esto, ‘limpia’ el efecto de la repetición y se fija solo en la intención. No le importa cuántas veces aparece la palabra (su tamaño o norma  $L_2$ ), sino hacia dónde apunta su significado.

### 1.1.c. Operación Álgebra Vectorial

Al intentar la operación excelente - positivo + negativo, el modelo arrojó un error de vocabulario (KeyError). Esto se debe a que la palabra "negativo" no alcanzó la frecuencia mínima necesaria para ser incluida en el entrenamiento. Conclusión: Esto demuestra que con un corpus pequeño de reseñas, el álgebra de embeddings es limitada. Para que funcione, necesitaríamos un dataset más masivo donde términos abstractos como "negativo." "positivo." aparezcan con mayor frecuencia y en contextos variados.

## 1.2. Clasificación de Texto con Transformers: Compara el F1-Score de ambos enfoques. ¿En qué casos específicos falla el modelo TF-IDF donde el modelo de Transformers acierta?.

Al comparar ambos modelos, TF-IDF obtuvo un desempeño superior (F1: 0.90) frente a BERT (F1: 0.89). Aunque BERT es teóricamente más robusto para entender el contexto, en este dataset de reseñas cortas y directas, la frecuencia de palabras clave (como "malo", ".excelente." "sucio") detectada por TF-IDF fue suficiente para clasificar con mayor precisión. BERT requiere un volumen de datos más complejo para demostrar su ventaja real; con la data actual, un modelo simple de conteo es más efectivo y eficiente.

El modelo TF-IDF falla principalmente en el sarcasmo y la negación Distante ya que trata las palabras de forma aislada. En una frase como ".excelente producto aunque la atencion fue lenta", TF-IDF ve ".excelente" "lenta", perdiendo la estructura que un Transformer (vía mecanismos de atención) sí captura.

Tabla 2: Resumen Comparativo: TF-IDF vs. Transformers

Característica	TF-IDF	Transformers (BERT)
Manejo de Contexto	Nulo (palabras aisladas)	Alto (bidireccional)
F1-Score Típico	0,9000	0,8924
Fortaleza	Eficiencia estadística en términos clave	Captura de matices y estructuras adversativas
Caso de Éxito	Clasificación precisa de textos directos.	Identificación correcta de quejas con conectores (ej: "pero")

## 2. Implementación de BERTopic

### 2.1. Debes explicitar y justificar la configuración de los tres pasos clave del pipeline:

#### 2.1.a. Embeddings: ¿Qué modelo de sentencia utilizaste? (ej: all-MiniLM-L6-v2). Justifica tu decisión.

Se seleccionó el modelo paraphrase-multilingual-MiniLM-L12-v2, porque el dataset retail\_reviews.csv contiene texto en español. A diferencia de modelos tradicionales de “bolsa de palabras”, este transformador es multilingüe y genera vectores densos que capturan el contexto semántico. La versión MiniLM es ideal porque es ligera y rápida, pero lo suficientemente potente para entender que “pésima calidad” y “mala experiencia” tienen un significado similar, aunque no compartan las mismas palabras.

#### 2.1.b. Reducción de Dimensionalidad (UMAP): ¿Por qué es necesario reducir dimensiones antes de clusterizar?

Los embeddings suelen tener cientos de dimensiones (ej. 384). La clusterización sufre de la "maldición de la dimensionalidad". UMAP es necesario para reducir esto a un espacio de baja dimensión (ej. 5) preservando la estructura local y global de los datos antes de agruparlos.

#### 2.1.c. Clusterización (HDBSCAN): ¿Cómo maneja este algoritmo el “ruido” o los documentos que no encajan en ningún tópico?

Este algoritmo es ideal para retail porque no te obliga a definir el número de tópicos (  $K$  ) de antemano. Maneja el ruido, asignando los documentos que no tienen una estructura clara al Tópico -1, evitando que grupos irrelevantes contaminen los temas principales.

## 2.2. Interpretación de Negocio

Tras procesar las reseñas con el modelo de descomposición de matrices **NMF** (Non-Negative Matrix Factorization), se identificaron 5 tópicos latentes que agrupan las preocupaciones y satisfacciones de los clientes.

El modelo permite separar la logística pura (Tópico 2: “llegó rápido”) de la experiencia de recepción (Tópico 0: “bien empackado”). A diferencia de un análisis de sentimientos tradicional el NMF revela que para el cliente la protección física del producto es un eje de opinión independiente de la velocidad de envío.

La recurrencia de términos como “empacado” y “bien” en los tópicos 0 y 3 sugiere que el empaquetado es un factor crítico de diferenciación. Para mejorar la retención, la empresa debe mantener los estándares de protección actuales, ya que el estado del paquete al llegar es tan relevante para la satisfacción final como la rapidez del transportista.

Tabla 3: Resultados del Análisis de Tópicos (NMF)

Tópico	Nombre Sugerido	Palabras Clave (NMF)	Interpretación de Negocio
0	Satisfacción y Empaque	contento, empackado, bien, buena	Clientes satisfechos con la integridad física del paquete.
1	Atributos y Estética	perfectos, compra, color, tamaño	Feedback sobre el cumplimiento de expectativas visuales.
2	Logística de Entrega	estado, recomendado, llegó, rápido	Comentarios enfocados en la velocidad y cumplimiento de tiempos.
3	Calidad Percibida	eficaz, empackado, bien, recomendado	Usuarios que validan la efectividad del producto entregado.
4	Rendimiento y Uso	funciona, excelente, duradero, es	Foco en la vida útil y funcionamiento técnico del artículo.

### 3. Sistemas de Recomendación: Filtrado Colaborativo Explícito

#### Análisis de la Data

##### En video\_ratings.csv (Para Recomendación SVD)

Aquí la limpieza fue más “quirúrgica” en comparacion a lo realizado en la parte 1 porque trabajamos con números, no con palabras.

Al revisar el archivo, notamos que la escala es de 1 a 5, pero hay un registro (y quizás más) con un rating de 999 (ej. Usuario 253, Película 121). El algoritmo SVD intenta minimizar el error cuadrático (RMSE). Si dejamos un 999, el modelo hará un esfuerzo matemático sobrehumano para intentar predecir ese número gigante, “rompiendo” todas las demás predicciones.

Por esto aplicamos `df_ratings[df_ratings['rating'] <= 5]` para asegurar que el modelo solo aprenda de la escala real.

También aseguramos que los `user_id` y `movie_id` sean tratados como identificadores para que la librería Surprise pueda construir la matriz de utilidad correctamente.

#### Resumen del impacto

Tabla 4: Impacto de la Limpieza de Datos en el Modelo

Problema	Consecuencia si NO se limpia	Resultado tras la limpieza
Símbolos (#@&*)	Tópicos sucios y palabras sin sentido.	Temas claros y profesionales.
Mayúsculas	Fragmentación de los grupos (clusters).	Unión de conceptos idénticos.
Rating 999	RMSE altísimo (error de 50.0 o más).	RMSE real y útil (cerca de 0,8 – 1,2).

### 3.1. Factorización Matricial (SVD)

#### 3.1.a. Implementación y Validación

El algoritmo Funk-SVD descompone la matriz original de calificaciones en dos matrices de menor rango (vectores latentes) que representan los perfiles de usuario y las características de las películas.

#### 3.1.b. Pregunta Teórica: Si el RMSE es bajo (ej: 0.8 estrellas), ¿significa necesariamente que el sistema es bueno recomendando? Justifica tu respuesta contrastando el error de predicción vs. la calidad del ranking (Top-N).

No, el RMSE no es suficiente. Aunque nuestro modelo tiene un error bajo (0.69), esta métrica solo nos dice qué tan cerca estuvimos de adivinar la "nota", pero no nos asegura que las recomendaciones sean interesantes para el usuario.

La diferencia clave: \* Minimizar el error (RMSE): Es como tratar de adivinar que a alguien le darás un 4.2 a una película y que sea un 4.5.

Optimizar el ranking: Es asegurarte de que las películas que más le van a gustar aparezcan primero en su pantalla.

Un detalle importante: En nuestro análisis, los usuarios nuevos tuvieron un error bajísimo (0.17), pero esto es engañoso porque tenemos pocos casos. Con tan poca información, no podemos asegurar que el sistema sea "perfecto" para ellos; simplemente no tenemos datos suficientes para evaluarlos bien todavía.

### 3.2. El problema del Cold-Start: Propuesta de Estrategia Híbrida

Para mejorar el desempeño con estos usuarios, se propone una Estrategia Híbrida de “Switching” (Conmutación)

**Fase Inicial (Basada en Contenido):** Cuando el usuario tiene  $< 3$  interacciones, el sistema no intenta usar filtrado colaborativo (SVD). En su lugar, utiliza metadatos de los ítems (género, director, etiquetas). Si el usuario vio una película de “Acción”, le recomendamos más “Acción” basándonos en la similitud de los objetos.

**Fase de Transición:** Una vez que el usuario alcanza un umbral de actividad, el sistema empieza a promediar las recomendaciones de contenido con las de SVD.

**Fase Madura (Colaborativa):** Con suficiente data, el sistema confía plenamente en SVD, permitiendo descubrir gustos que van más allá de etiquetas simples, basándose en el comportamiento de usuarios similares.

**Beneficio:** Esta estrategia evita que el usuario reciba recomendaciones aleatorias o de “promedio global” al inicio, mejorando la retención desde el primer día.

## 4. Sistemas de Recomendación: Feedback Implícito y LTR

### Analisis de la Data

#### Eliminación de Valores Negativos (Ruido Logístico)

Filtramos el dataset para mantener solo registros donde `play_count > 0`. En el archivo se observan valores como -130.0 o -102.0. En un sistema de feedback implícito, el conteo de reproducciones representa una frecuencia de interacción. Un número negativo no tiene sentido físico (no puedes “des-escuchar” una canción) y desestabilizaría la función de costo de ALS, que espera valores de 0 a infinito para calcular la confianza.

#### Gestión de Valores Nulos (NaN)

Se utilizó `.dropna()` para eliminar cualquier fila que no tuviera `user_id`, `song_id` o `play_count`. ALS necesita una matriz completa de interacciones. Un valor nulo en los IDs impide que el modelo asocie la interacción a un usuario o ítem específico, lo que generaría errores al intentar construir la matriz dispersa (Sparse Matrix).

**Normalización de Tipos de Datos** Convertimos `user_id` y `song_id` a enteros (`int`). Al cargar el CSV, muchas veces los IDs se leen como flotantes (ej: 0.0). Las librerías de recomendación como `implicit` o `scipy.sparse` requieren índices enteros para mapear las filas y columnas de la matriz de manera eficiente en memoria.

**Construcción de la Matriz Dispersa (CSR)** Transformamos el DataFrame plano en una matriz CSR (Compressed Sparse Row). Los logs de música son masivos y la mayoría de los usuarios no han escuchado la mayoría de las canciones (la matriz está “vacía” en un 99%). Guardar esto como una tabla normal consumiría gigas de RAM innecesariamente. La matriz CSR solo guarda los valores donde sí hay reproducciones, optimizando el entrenamiento de ALS.

### Resumen de lo que se mejoro

Tabla 5: Acciones de Limpieza y su Impacto en el Modelo

Problema Detectado	Acción de Limpieza	Impacto en el Modelo
Play_count negativo	Eliminación de registros	Evita que la confianza del modelo sea errónea.
IDs como flotantes (1970.0)	Conversión a <code>int</code>	Permite la indexación en matrices dispersas.
Registros incompletos	<code>dropna()</code>	Asegura la integridad de la matriz de usuario-ítem.
Datos masivos/vacíos	Transformación a CSR	Permite que el modelo se entrene en segundos en lugar de horas.

## 4.1. Mínimos Cuadrados Alternados (ALS)

Al trabajar con el archivo `music_logs.csv`, nos encontramos con un pequeño caos: los datos de `play_count` incluían valores negativos imposibles y algunos picos exagerados que generaban ruido. Para que nuestro modelo sea realmente preciso, el primer paso fue “sanear” estas cifras antes de iniciar el entrenamiento, lo que se describió arriba.

En ALS para feedback implícito, no tratamos el `play_count` como una calificación directa, sino como una medida de certeza.

**¿Por qué 0 no es una preferencia negativa?:** Si un usuario nunca ha escuchado una canción (valor 0), no significa que la deteste. Simplemente significa que no sabemos si le gusta. Por eso, el modelo le asigna una confianza baja.

**La diferencia:** En el filtrado colaborativo tradicional (SVD), un 0 se trata como un dato faltante. En ALS Implícito, el 0 es un dato observado con confianza mínima, mientras que un valor alto (ej. 100 reproducciones) le dice al modelo: “Estoy muy seguro de que a este usuario le encanta esta canción”.

## 4.2. Evaluación de Ranking (NDCG)

### Generación de Top-10 y cálculo de NDCG5

Supongamos que para un usuario, el sistema recomienda 5 canciones. En la “vida real” (test set), el usuario escuchó la canción 1, la 3 y la 4.

Ítem	Relevancia	Posición	Cálculo de Ganancia ( $rel_i / \log_2(i + 1)$ )
Canción A	1	1	$1 / \log_2(2) = 1,00$
Canción B	0	2	$0 / \log_2(3) = 0,00$
Canción C	1	3	$1 / \log_2(4) = 0,50$
Canción D	1	4	$1 / \log_2(5) \approx 0,43$
Canción E	0	5	$0 / \log_2(6) = 0,00$
DCG Total @5:			1.93

Tabla 6: Cálculo detallado del Discounted Cumulative Gain (DCG).

Para obtener el NDCG, dividiríamos este 1.93 por el IDCG (el puntaje si las 3 canciones relevantes hubieran estado en las posiciones 1, 2 y 3).

### ¿Cómo penaliza el NDCG vs Precision@K?

**Precision@K:** Es una métrica “plana”. Si en el Top-5 hay 3 canciones correctas, la precisión es 0.6 (60 %), sin importar si las correctas estaban al principio o al final de la lista.

**NDCG:** Utiliza un logaritmo de penalización por posición.

- **Penalización por posición:** Si pones la canción favorita del usuario en la posición 5, el denominador ( $\log_2(6)$ ) reducirá drásticamente el valor del acierto.
- **Relevancia óptima:** Si la pones en la posición 1, el valor obtenido es el máximo posible.

**Conclusión de Negocio:** El NDCG es mucho más realista para sistemas de recomendación porque reconoce que los usuarios rara vez bajan hasta el final de la lista. Un algoritmo que acierta en el primer lugar es infinitamente mejor que uno que acierta en el quinto, aunque ambos tengan la misma precisión.

Tabla 7: Comparativa de Modelos: SVD vs. ALS

Característica	SVD (Video Ratings)	ALS (Music Logs)
Tipo de Feedback	<b>Explícito:</b> El usuario dice activamente qué le gusta (1-5 estrellas).	<b>Implícito:</b> El sistema deduce el gusto por el comportamiento ( <code>play_count</code> ).
Métrica de Éxito	<b>RMSE:</b> Error entre la predicción y el rating real.	<b>NDCG / Precision@K:</b> Calidad del orden de la lista de recomendación.
Interpretación del 0	<b>Dato Faltante:</b> No sabemos nada, el modelo lo ignora.	<b>Confianza Baja:</b> No lo ha escuchado, pero es un dato que el modelo usa para aprender.
Problema Principal	Sensible a <i>outliers</i> numéricos (como un rating 999).	Sensible al sesgo de popularidad (canciones virales).
Uso Ideal	Plataformas tipo IMDb o encuestas de satisfacción.	Plataformas de streaming (Spotify, Netflix, TikTok).

## 5. Desafíos avanzados

### 5.1. NLP: Análisis de Sentimiento Basado en Aspectos (ABSA) y Señales Mixtas

Al ejecutar el filtro de señales mixtas en el dataset, se identificaron 147 reseñas que contienen simultáneamente términos de polaridad positiva y negativa. Este subconjunto es crítico porque revela que una etiqueta global '(Positivo/Negativo)' es insuficiente para una gestión de CRM efectiva.

Existen casos donde el sentimiento global es "Negativo", pero el texto desglosa fallas en aspectos distintos como la calidad del producto frente al precio o la velocidad.

```
===== 5.1. NLP: Análisis de Sentimiento Basado en Aspectos (ABSA) =====
Reseñas mixtas detectadas: 226
text sentiment
0 Atención al cliente pésima. Producto de mala c... Negativo
2 Atención al cliente pésima. Producto de mala... Negativo
6 Producto caro y muy lento en funcionar. Gran d... Negativo
11 Producto caro y muy lento en funcionar. Gran d... Negativo
23 Atención al cliente pésima. Producto de mala c... Negativo
```

Figura 2: Ejemplos de reseñas con señales mixtas.

Riesgo Estratégico: Confiar únicamente en un modelo de sentimiento global es peligroso para

el área de logística. Un cliente puede estar satisfecho con el producto pero frustrado con el empaque sucio o el retraso en la entrega. Ignorar estas "señales mixtas" impide que el sistema dispare alertas automáticas para corregir problemas en la última milla.

Lógica de Negocio sugerida: Para mejorar esto, el algoritmo no debe solo clasificar, sino extraer aspectos. Si detectamos palabras como "tardó", correo, "paquete" cerca de una connotación negativa, el sistema debería disparar automáticamente una alerta al equipo de operaciones, independientemente de si el cliente puso 5 estrellas.

### **Pseudocódigo:**

---

1. **Definir** lista de términos 'Envío': [entrega, correo, tardó, llegó, paquete, envío, domicilio]
  2. **Para cada** reseña:
    - a. Tokenizar texto.
    - b. **Si** existe intersección entre tokens y lista de aspecto 'Envío':
      - i. Analizar polaridad de las 3 palabras cercanas al término.
      - ii. **Si** las palabras cercanas son negativas (ej: tarde, mal, nunca):
        - Crear ticket de soporte categoría 'Logística'.
        - Ignorar clasificación 'Positiva' global si existe.
- 

#### **5.1.a. Justificación**

Notamos que el modelo global falla porque las palabras positivas suelen tener más "peso" que las negativas, ocultando quejas reales. La lógica CRM debe separar oraciones para que un "excelente" no anule un "retraso".

#### **5.2. RecSys: Evaluación de Sesgo de Popularidad y 'Long Tail'**

Tras evaluar las recomendaciones generadas para una muestra de 100 usuarios, el sistema obtuvo una Cobertura de Catálogo del 77.69%.

Este resultado indica que el algoritmo es capaz de recomendar una gran variedad de productos, alcanzando a mostrar más de tres cuartas partes del catálogo disponible. Sin embargo, existe un 22.31 % de los ítems que nunca son recomendados (el "Long Tail"). Estos productos suelen ser los menos populares o con menos interacciones, quedando fuera del radar del modelo SVD estándar, el cual tiende a favorecer aquellos ítems que tienen una base de datos más sólida de calificaciones.

#### **Impacto de una cobertura baja:**

Sesgo de Popularidad: Si la cobertura fuera baja, el sistema solo recomendaría "best sellers", creando una burbuja de contenido que aburre al usuario.

Impacto en Inventario: Una baja cobertura implica que los productos del "Long Tail" (nicho) quedan inmovilizados en bodega, generando costos financieros por stock estancado.

Lifetime Value (LTV): Un buen recomendador debe “desempolvar” productos menos conocidos pero relevantes para mantener el interés del cliente en navegar el catálogo.

### 5.2.a. Estrategia

Una cobertura baja (generalmente  $< 20\%$  en SVD sin ajustes) indica que el retailer está perdiendo dinero en inventario que nadie ve. Recomienda técnicas de Diversificación.

## 5.3. RecSys: Re-ranking Híbrido por Margen Financiero (profit-aware RecSys)

Al aplicar la función de re-ranking por margen financiero para el Usuario 260, observamos un cambio estratégico en el orden de las recomendaciones. El producto con movie id 125, que originalmente ocupaba la sexta posición en el ranking de relevancia, subió al primer lugar gracias a su etiqueta de “High Margin” y el factor de impulso de 1.2x.

```
--- Ejecutando Parte 5.3: Re-ranking por Margen ---
```

	movie_id	margin	score_orig	score_final
5	125	High	3.318060	3.981672
0	28	Low	3.385831	3.047248
1	234	Low	3.359567	3.023611
2	31	Low	3.336855	3.003169
3	245	Low	3.333733	3.000360
4	1	Low	3.324766	2.992290
6	96	Low	3.315660	2.984094
7	156	Low	3.298640	2.968776
8	74	Low	3.290440	2.961396
9	134	Low	3.282997	2.954698

Figura 3: Top 10.

Dilema Ético y Estratégico: Intervenir la “pureza” del algoritmo para favorecer productos con mayor margen es una práctica común, pero delicada.

**Rentabilidad vs. Relevancia:** Si se “empuja” demasiado un producto solo por su margen ( $1,2x$ ) y este no es realmente relevante para el usuario, la métrica de precisión caerá.

**Sostenibilidad:** A corto plazo, el re-ranking aumenta el margen por transacción. Sin embargo, si el usuario percibe que el sistema es un “vendedor insistente” en lugar de un asesor, se destruye la confianza y la retención a largo plazo.

**Recomendación:** Lo ideal es aplicar este sesgo únicamente entre productos que ya posean un score de relevancia alto.