Informe Técnico del Proyecto: Bot de Recomendación de Optativos

Adrián Hernández Castellanos - C312
Laura Martir Beltrán - C311
Yesenia Valdés Rodríguez - C311
Gabriel Alonso Coro - C312
Josué Rolando Naranjo Sieiro - C311
Abel Ponce González - C311

June 20, 2025

Abstract

Este trabajo presenta el diseño e implementación de un sistema inteligente de recomendación de asignaturas optativas para estudiantes universitarios. El sistema combina técnicas de procesamiento de lenguaje natural, minería de opiniones y modelos de recomendación basados en contenido. Se incluye un análisis detallado de los métodos utilizados y se discute su potencial impacto en la toma de decisiones académicas.

1 Introducción

En muchas universidades, los estudiantes deben seleccionar asignaturas optativas con información limitada, lo cual puede derivar en elecciones poco acertadas que afectan su rendimiento académico y su motivación. Este proyecto busca mejorar ese proceso mediante un sistema de recomendación de optativas personalizado.

La propuesta se basa en representar tanto a los estudiantes como a las asignaturas mediante vectores temáticos, extraer conocimiento útil desde reseñas estudiantiles, y ofrecer recomendaciones fundamentadas. Aunque el sistema fue implementado como un bot de Telegram, el enfoque metodológico puede trasladarse fácilmente a otras plataformas.

2 Estado del Arte

Los sistemas de recomendación son ampliamente utilizados en dominios como comercio electrónico y entretenimiento, pero su aplicación en contextos educativos sigue en desarrollo. Existen propuestas para recomendar cursos en plataformas como EVEA, pero pocas iniciativas para recomendaciones personalizadas dentro de una universidad presencial y con restricciones curriculares específicas.

Nuestro enfoque combina representaciones semánticas de contenido (TF-IDF) con minería de opiniones de estudiantes anteriores. A diferencia de métodos manuales o rankings globales, aquí se considera el perfil individual de cada estudiante.

3 Especificaciones Técnicas

3.1 Módulos del sistema

A través de una interfaz conversacional, los usuarios pueden explorar optativas, calificarlas, dejar reseñas y recibir recomendaciones personalizadas. Por su parte, los profesores cuentan con herramientas para gestionar el sistema de optativas, validar archivos y supervisar la actividad del bot. El sistema hace uso de estructuras de datos en formato JSON, control de autenticación y técnicas de recuperación de información para garantizar una experiencia eficiente y segura.
La arquitectura del proyecto se compone de varios módulos, cada uno responsable de una funcionalidad específica:

main.py: Implementa la lógica principal del bot, gestionando los flujos conversacionales, los comandos disponibles, la autenticación de profesores, y la interacción con los datos.

search_engine.py: Motor de búsqueda avanzado que combina TF-IDF y técnicas de ponderación para ofrecer resultados relevantes y adaptados a las preferencias del usuario.

Archivos de datos data/: Contienen la información estructurada sobre estudiantes, optativas, profesores y reseñas. Estos archivos actúan como la fuente central del conocimiento del sistema.

3.2 Autenticación de Profesores

El sistema incorpora un mecanismo de autenticación para profesores mediante el comando /login, que requiere credenciales válidas. Se distingue entre profesores regulares y un perfil especial de superadmin, el cual cuenta con permisos ampliados para modificar datos sensibles y ejecutar acciones administrativas. Una vez autenticado, el profesor mantiene su sesión activa gracias a la identificación por ID de Telegram, lo cual evita accesos no autorizados y facilita un control más preciso de las operaciones realizadas.

3.3 Manejo de Optativas y Estudiantes

El bot permite a los profesores gestionar el ciclo de vida de las optativas y los estudiantes asignados a ellas. Las operaciones disponibles incluyen:
”

- Agregar o eliminar registros de estudiantes y profesores.
- Crear nuevas optativas definiendo campos como nombre, profesor responsable, descripción, número de plazas y asignaturas relacionadas.
- Asignar estudiantes a optativas específicas, lo cual permite mantener una correspondencia clara entre cada usuario y su elección académica.

Estas acciones están restringidas a usuarios autenticados, lo que garantiza la integridad del sistema.

3.4 Sistema de Reseñas

Con el objetivo de fomentar la retroalimentación y mejorar las decisiones futuras, el sistema permite que cada estudiante deje una reseña sobre la optativa que tiene asignada. La reseña incluye una puntuación numérica del 1 al 5 y un comentario descriptivo. Si el estudiante ya ha emitido una reseña sobre la optativa en cuestión, la nueva entrada sustituye automáticamente a la anterior. Todas las reseñas se almacenan en el archivo reseñas.json. Además, el proceso puede cancelarse en cualquier momento para evitar entradas incompletas o involuntarias.

3.5 Motor de Búsqueda Inteligente

Uno de los elementos más importantes del proyecto es su motor de búsqueda, diseñado para ofrecer recomendaciones y resultados relevantes. Entre sus características destacan:

- Modelo de similitud TF-IDF para evaluar la relevancia de términos en función de su frecuencia y contexto.
- Ponderación personalizada mediante una sintaxis como ‘palabra***‘, que incrementa el peso de ciertos términos en la búsqueda.
- Exclusión de términos mediante el uso de ‘!palabra‘, permitiendo refinar consultas y eliminar resultados no deseados.
- Priorización semántica, favoreciendo optativas con mejores calificaciones y comentarios más positivos.
- Cualquier término insertado en la búsqueda que aparezca en un campo relacionado a una optativa resultará en otorgar importancia a esa optativa a la hora de mostrar una respuesta a la consulta. Los campos incluyen nombre de la optativa, nombre del profesor que la imparte, descripción y asignaturas relacionadas.

Estas funcionalidades permiten realizar consultas complejas de manera natural, acercando al usuario a una experiencia de búsqueda adaptable.

3.6 Carga de Archivos por Profesores

El sistema permite a los profesores autenticados subir archivos .json para actualizar la base de datos del sistema. Solo se admiten archivos con nombres específicos (estudiantes.json, optativas.json, profesores.json) y con estructuras válidas. La codificación de los archivos debe ser UTF-8 sin escape ASCII. El bot valida automáticamente el contenido antes de reemplazar los datos existentes, garantizando la consistencia y evitando corrupciones accidentales. Esta funcionalidad proporciona una vía rápida y segura para mantener actualizado el sistema sin intervención manual en el servidor. A continuación se muestra el formato esperado de cada archivo .json reemplazable.

3.7 Registro de Operaciones

Cada acción realizada por un profesor queda registrada en el archivo **logs/registro_operaciones.txt**. Se almacena información detallada sobre qué acción se realizó,

por quién, en qué momento y sobre qué entidad (estudiante, optativa o archivo). Este registro se limita a las últimas 1000 entradas, evitando así el crecimiento descontrolado del archivo. Además, mediante el comando **/log**, los profesores pueden descargar el historial para su revisión.

3.8 Comandos del Bot

El bot responde a una serie de comandos estructurados que permiten una interacción fluida para estudiantes y profesores:

/start: Muestra las optativas disponibles al usuario.

/help: Despliega una guía de uso y comandos disponibles. Esta guía cambia si se está logueado como profesor.

/rev: Permite al estudiante escribir una reseña sobre su optativa asignada.

/vrev: Muestra las reseñas realizadas sobre una optativa específica.

/login: Inicia sesión como profesor o superadmin.

/log: Descarga el historial de operaciones recientes (solo para profesores).

/delrev: Elimina todas las reseñas del sistema (solo superadmin).

3.9 Superadmin

El superadmin es un usuario de categoría profesor con permisos administrativos especiales, que puede agregar y eliminar profesores y modificar el archivo profesores.json (es el único usuario que puede ejecutar estas operaciones por motivos de seguridad). Sus acciones quedarán registradas en el registro de la misma manera que las de cualquier otro usuario, y podrá realizar las mismas acciones que estos últimos. Para autenticarse como superadmin, se debe ejecutar el comando /login y se debe insertar como **usuario: superadmin** y **contraseña: admin1234**.

4 Descripción del Sistema de Búsqueda

4.1 Método de recomendación

Uno de los desafíos centrales en sistemas de recomendación basados en contenido es cómo representar de manera eficiente y significativa los elementos a comparar. En nuestro caso, esto se traduce en representar tanto a las optativas como a las preferencias o perfiles de los estudiantes. El modelo TF-IDF (Term Frequency–Inverse Document Frequency) se adopta como una solución práctica y bien fundamentada para este propósito.

TF-IDF parte de una premisa sencilla: no todos los términos tienen la misma importancia. Las palabras que aparecen frecuentemente en un documento son potencialmente relevantes para describirlo (frecuencia de término), pero si una palabra aparece en casi todos los documentos, su capacidad de discriminación disminuye (frecuencia inversa de documento). Al ponderar ambos factores, TF-IDF permite resaltar los términos que mejor caracterizan un contenido frente al resto de la colección.

Matemáticamente, el peso de un término t en un documento d se define como:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \cdot \text{IDF}(t)$$

donde:

- $\text{TF}(t, d)$ es la frecuencia del término t en el documento d,
- $\text{IDF}(t) = \log\left(\frac{N}{1+n_t}\right)$, con N el total de documentos y n_t el número de documentos que contienen t.

En este proyecto, cada optativa se representa como un documento, compuesto por una combinación de campos textuales: nombre, descripción, profesor que la imparte y asignaturas asociadas. Esta representación permite construir una matriz dispersa donde cada fila es una optativa y cada columna un término, con valores TF-IDF como pesos. Lo mismo se aplica para construir un vector representativo de la consulta de un estudiante.

El uso de TF-IDF permite entonces medir la similitud entre vectores mediante el coseno del ángulo entre ellos, lo cual da lugar a una métrica de recomendación directa:

$$\text{sim}(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

Esta formulación tiene varias ventajas:

- Es independiente de la longitud de los textos.
- Requiere poco cómputo una vez construida la matriz.
- Se adapta fácilmente a nuevas entradas mediante transformaciones directas.
- No depende de anotaciones manuales ni estructuras semánticas complejas.

Si bien existen modelos más sofisticados, como *embeddings* neuronales o representaciones semánticas profundas, TF-IDF sigue siendo una base sólida para tareas de recomendación en dominios con datos estructurados y acotados como este.

4.2 Sistema de reseñas dentro del TF-IDF

Los estudiantes pueden escribir reseñas sobre optativas previamente cursadas, incluyendo puntuaciones del 1 al 5 y comentarios libres. Estas reseñas se almacenan en un archivo JSON estructurado con nombre, grupo, optativa, puntuación y comentario.

Las reseñas y sus puntuaciones se integran al sistema de recomendación como un campo extra a analizar cuando se está procesando la consulta. Las reseñas positivas agregan más peso al valor final de una optativa respecto a una consulta, lo que garantiza un orden de relevancia que prioriza a las optativas mejor valoradas por encima de las demás.

4.3 Palabras ponderadas y excluidas

Esta característica representa una mejora al clásico sistema de búsqueda TF-IDF al permitir que los estudiantes expresen de una manera más precisa sus intereses respecto al contexto que están estableciendo en su consulta. Las palabras precedidas por un número entre 1 y 5 del símbolo ***** tendrán un impacto más significativo en la búsqueda realizada y se otorgará un peso final mayor a las optativas que las contengan. Por otro lado, las palabras antecedidas por un símbolo **!** se considerarán no queridas en los resultados otorgados por el motor de búsqueda, y otorgará peso cero inmediato a cualquier optativa que contenga dicha palabra en su descripción, lo cual conduce a no ser mostrada en el resultado de la búsqueda.

4.4 Asignaturas relacionadas a una optativa

Es posible que algunos profesores quieran agregar como dato extra las asignaturas con las que está relacionada una asignatura optativa. Esto serviría de guía para los estudiantes, ya que tienen una comparación fuerte por su propia experiencia personal para decantarse por una u otra optativa. Es por este propósito que se decidió implementar el sistema de asignaturas relacionadas y agregarlo como una característica extra para el motor de búsqueda. Este campo es analizado por el motor TF-IDF y cada asignatura es tratada como un término al que además se le asocia un peso sumado fuera de los pesos de la matriz, lo cual ayuda a que el orden final de las respuestas estén en sintonía con las intenciones de los estudiantes.

5 Capas de seguridad

5.1 Cuentas autenticadas

Los profesores autenticados son los únicos con acceso a visualizar y modificar el registro de estudiantes con sus respectivas optativas vinculadas. Para llevar a cabo esta medida, es necesario que cada profesor cuente con una cuenta y contraseña en el sistema. Estos datos serán guardados en el JSON correspondiente y servirán para que el profesor realice cualquier gestión o modificación que crea pertinente en el sistema. Además, podrán agregar si así lo desean ofertas de optativas que deseen a través de la interfaz conversacional o la subida de archivo JSON correspondiente a las optativas siempre que cumpla todas las especificaciones anteriormente mencionadas.

5.2 Superadmin

Se prevee que al desplegar el sistema, no se tenga conocimiento de los profesores que usarán el sistema. Para agregarlos a la base de datos es necesario contar con un usuario que además tendrá el privilegio de eliminarlos si resulta necesario. Esta es la funcionalidad del superadmin, pues además de contar con todos los privilegios de profesor, también puede gestionar a los mismos y limpiar la base de datos de reseñas realizadas en caso de limpieza del sistema. Este cargo debe delegarse a un operario específico de la entidad que use este sistema, y será el encargado del mantenimiento del mismo.

5.3 Registro

Todas las acciones realizadas por profesores y por el superadmin deben quedar registradas para prevenir confusiones, malentendidos o usos no intencionados del sistema bajo la condición de profesor usuario del sistema. Este registro posee actualización y limpieza automática y se puede obtener a través del comando especificado en el bot.

6 Evaluación y experimentación

A continuación se presenta de manera detallada el análisis estadístico efectuado sobre las simulaciones realizadas con estudiantes para evaluar y mejorar el proyecto. A través de simulaciones de elecciones reales y recomendaciones generadas por un sistema basado en contenido, así como del estudio de la influencia de reseñas en las decisiones de los usuarios, se obtienen métricas y visualizaciones que permiten extraer conclusiones beneficiosas al proyecto.

Se diseñaron experimentos para evaluar:

- Simulación de elecciones reales de estudiantes.
- Generación de recomendaciones basadas en contenido.
- Cálculo de métricas de evaluación para el sistema de recomendación.
- Visualización de resultados de la calidad de las recomendaciones.
- Simulación de la influencia de reseñas en las decisiones de selección.
- Análisis estadístico de la influencia: pruebas de hipótesis y regresiones.
- Interpretación de los resultados y conclusiones.

6.1 Simulación de Elecciones Reales de Estudiantes

Para obtener un *ground truth* (verdad de suelo) con el cual comparar las recomendaciones, se genera una simulación de las elecciones que realizarían S estudiantes de manera aleatoria pero ponderada por la *popularidad* de cada optativa. El algoritmo empleado es el siguiente:

1. Definir $S = 70$ (número total de estudiantes por defecto).
2. Para cada óptativa i:
   - Asignar un valor de *popularidad* p_i muestreado de manera uniforme en el rango $[0.5, 2.0]$.
3. Para cada estudiante j de 1 a S:
   - Construir el vector de pesos $w_{j,i} = p_i$ para todas las optativas i.
   - Normalizar los pesos:
   $$\pi_{j,i} = \frac{w_{j,i}}{\sum_{k=1}^{N} w_{j,k}} \quad \forall i = 1, \ldots, N.$$
   - Seleccionar una optativa $\hat{i}_j$ mediante muestreo categórico con probabilidad $\pi_{j,i}$.

- Registrar la elección real como un registro (estudiante$_j$, $\hat{i}_j$, nombre de la óptativa $\hat{i}_j$) en un `DataFrame`.

El resultado de este proceso es un conjunto de S filas que representan las elecciones "reales" de los estudiantes. Estas elecciones se almacenan en un `CSV` con columnas:

- `estudiante_id`: Identificador del estudiante $(1, \ldots, S)$.
- `optativa_id`: Identificador de la optativa seleccionada.
- `optativa_nombre`: Nombre de la optativa seleccionada.

Se planea usar métricas como precisión, cobertura y ranking medio, así como porcentajes de coincidencia y tablas de comparación entre distintos métodos.

6.2 Pruebas realizadas

Se diseñó y ejecutó una batería de pruebas funcionales, estructuradas según los distintos perfiles de usuario (estudiante, profesor y superadmin) y las funcionalidades clave del sistema. Las pruebas se dividieron en los siguientes bloques:

- Pruebas de acceso y autenticación: se verificaron los flujos de inicio de sesión para profesores y superadmin, la validación de credenciales, la persistencia de sesión por ID de Telegram y el cierre de sesión con restauración del menú inicial.
- Gestión de archivos: se probaron casos de carga de archivos .json, incluyendo validaciones de nombre, estructura, codificación UTF-8 y contenido. Se verificó que los archivos válidos fueran correctamente reemplazados, y que cualquier acción quedara registrada en el archivo de log del sistema.
- Manejo de estudiantes: se probaron los flujos para agregar, visualizar y eliminar estudiantes, tanto de forma individual como masiva (usando la palabra clave 'TODO'). Se incluyó detección de duplicados, manejo de entradas mal formateadas y agrupamiento por grupo.
- Manejo de optativas: se evaluó la creación guiada de optativas mediante flujo conversacional, con validación de nombres duplicados, opción de cancelación y campos obligatorios. También se probaron la eliminación (por nombre y global), la desasignación automática de estudiantes y la visualización formateada.
- Gestión de profesores: el superadmin pudo agregar y eliminar profesores, mientras que los usuarios no autorizados recibieron los mensajes de advertencia correspondientes. También se probó la visualización general de profesores registrados.
- Sistema de reseñas: se evaluó el flujo completo para dejar una reseña, incluyendo validación del estudiante, optativa asignada, puntuación válida (1-5), reemplazo de reseñas existentes y almacenamiento en disco. Además, se probó la consulta de reseñas por optativa y la eliminación total de reseñas por el superadmin.

- Motor de búsqueda: se probaron búsquedas por texto libre con soporte para ponderación **(***)** y exclusión (**!palabra**). El sistema generó resultados basados en similitud semántica mediante un modelo TF-IDF, devolviendo información detallada por optativa, incluyendo descripción, plazas disponibles, asignaturas relacionadas y las mejores y peores reseñas, todo en mensajes separados para evitar límites de Telegram.
- Comandos y flujo conversacional: se probaron todos los comandos disponibles (**/start, /help, /login, /rev, /vrev, /log, /delrev**), así como sus comportamientos según el rol del usuario. El menú dinámico del profesor también fue verificado.
- Validación y tolerancia a errores: se incluyeron pruebas deliberadas con datos malformados, credenciales erróneas, comandos fuera de lugar y cancelaciones para comprobar la resiliencia del bot y la claridad de sus mensajes ante fallos.
- Registro y trazabilidad: se confirmó que toda acción administrativa (agregados, eliminaciones, modificaciones y cargas de archivo) quedara documentada correctamente en el archivo **registro_operaciones.txt**, manteniéndose dentro del límite de 1000 entradas más recientes.

Estas pruebas se ejecutaron manualmente desde múltiples cuentas de Telegram. El sistema cumplió con todos los objetivos definidos en la etapa de diseño.

6.3 Sistema de Recomendación Basado en TF-IDF

Para mejorar la capacidad del sistema de recomendación en capturar la relevancia semántica de las optativas respecto a los intereses de los estudiantes, se implementa un enfoque basado en TF-IDF (Term Frequency–Inverse Document Frequency) y similitud de coseno. Los pasos detallados son:

1. Construcción del corpus: Se crea un *corpus* textual compuesto por todos los documentos concatenados de cada optativa. Para la optativa i, el documento se define como:

$$\texttt{doc}_i = \texttt{nombre}_i + \text{“ ”} + \texttt{descripcion}_i,$$

   todo convertido a minúsculas y preprocesado (eliminación de puntuación y palabras vacías).

2. Cálculo de la matriz TF-IDF: Se construye la matriz TF-IDF $\mathbf{T} \in \mathbb{R}^{N \times V}$, donde N es el número de optativas y V el tamaño del vocabulario resultante. Cada fila $\mathbf{t}_i$ contiene los pesos TF-IDF de los términos en el documento `doc_i`.

3. Perfil de estudiante: Para cada estudiante j, se crean entre 1 y 3 palabras clave (intereses) muestreadas aleatoriamente de la lista:

$$\{\texttt{programación}, \texttt{matemáticas}, \texttt{estadística}, \texttt{comunicación}, \texttt{software}, \texttt{teoría}\}.$$

   Estas palabras clave se concatenan en un *documento de consulta* $\texttt{query}_j = \text{“}w_{j,1}\, w_{j,2}\, w''_{j,3}$ (solo las seleccionadas), también convertido a minúsculas y preprocesado.

4. Vectorización de la consulta: Se utiliza el mismo transformador TF-IDF entrenado en el corpus de optativas para transformar $\texttt{query}_j$ en un vector TF-IDF $\mathbf{q}_j \in \mathbb{R}^V$.

5. Cálculo de similitud: Se calcula la similitud de coseno entre el vector de consulta $\mathbf{q}_j$ y cada vector de optativa $\mathbf{t}_i$:
$$\text{sim}_{j,i} = \frac{\mathbf{q}_j \cdot \mathbf{t}_i}{\|\mathbf{q}_j\|_2 \, \|\mathbf{t}_i\|_2}, \quad i = 1, \ldots, N.$$

6. Generación de recomendaciones: Para cada estudiante j, se ordenan las optativas de mayor a menor según $\text{sim}_{j,i}$ y se seleccionan las primeras $K = 3$ como recomendaciones. Para cada posición $r = 1, 2, 3$, se almacena:
   - `estudiante_id`: Identificador del estudiante.
   - `optativa_id`: Identificador de la optativa recomendada.
   - `optativa_nombre`: Nombre de la optativa.
   - `rank`: Posición r (1=mejor recomendación, 3=tercer puesto).
   - `score`: Valor de similitud $\text{sim}_{j,i}$.

7. Almacenamiento: Las recomendaciones resultantes se guardan en un `DataFrame` con $S \times K = 210$ filas, cada fila representando (`estudiante_id`, `optativa_id`, `optativa_nombre`, `rank`, `score`). Finalmente, se exporta a un `CSV` para su posterior análisis.

6.4 Métricas de Evaluación del Sistema de Recomendación

Con el *ground truth* (elecciones reales) y las recomendaciones generadas, se calculan las siguientes métricas para evaluar la calidad del sistema de recomendación:

1. **Precisión en Top-K (Precision@K)**: Proporción de estudiantes cuya elección real aparece dentro de las $K = 3$ recomendaciones. Dados S estudiantes y la variable indicadora
$$\text{hit}_j = \begin{cases} 1, & \text{si optativa\_real}_j \in \{\text{optativas recomendadas}_j\}, \\ 0, & \text{en otro caso,} \end{cases}$$
la precisión se define como:
$$\text{Precision@K} = \frac{1}{S} \sum_{j=1}^{S} \text{hit}_j.$$

2. **Mean Reciprocal Rank (MRR)**: Para cada estudiante cuyo elemento real se encuentre en la lista de recomendaciones, se toma la posición r_j en que ocurre dicha elección real. El *reciprocal rank* es $\frac{1}{r_j}$. La MRR se calcula como:
$$\text{MRR} = \frac{1}{S} \sum_{j=1}^{S} \begin{cases} \frac{1}{r_j}, & \text{si hit}_j = 1, \\ 0, & \text{si hit}_j = 0. \end{cases}$$

3. **Cobertura (Coverage)**: Proporción de optativas distintas que aparecen en *cualquier* recomendación, respecto al total de optativas N. Si U es el número de optativas únicas recomendadas en las $S \times K$ celdas de recomendación, entonces:
$$\text{Coverage} = \frac{U}{N}.$$

4. **F1-Score**: Dado que en este escenario cada estudiante sólo tiene una elección real, el *recall* coincide con la *precisión* (ya que si la real está en Top-K, es tanto relevante como recuperada). Por lo tanto,

$$\text{Recall} = \text{Precision@K}, \quad \text{F1-Score} = 2 \times \frac{\text{Precision@K} \times \text{Recall}}{\text{Precision@K} + \text{Recall}} = \text{Precision@K}.$$

Estos valores se calculan automáticamente y se guardan en un `CSV` con una sola fila y cuatro columnas: `precision_at_k`, `mrr`, `coverage`, `f1_score`. A continuación, se muestran las métricas obtenidas (valores de ejemplo):

Métrica	**Valor**
Precision@3	0.314
MRR	0.735
Coverage	0.857
F1-Score	0.314

6.5 Visualizaciones de la Calidad de las Recomendaciones

Para interpretar la calidad y comportamiento del sistema, se generaron las siguientes visualizaciones:

6.5.1 Comparación entre Elecciones Reales y Recomendaciones Top-1

Se construyen dos distribuciones:

- Frecuencia con la que cada optativa fue **elección real** de los estudiantes.
- Frecuencia con la que cada optativa apareció en la posición **top-1** de las recomendaciones.

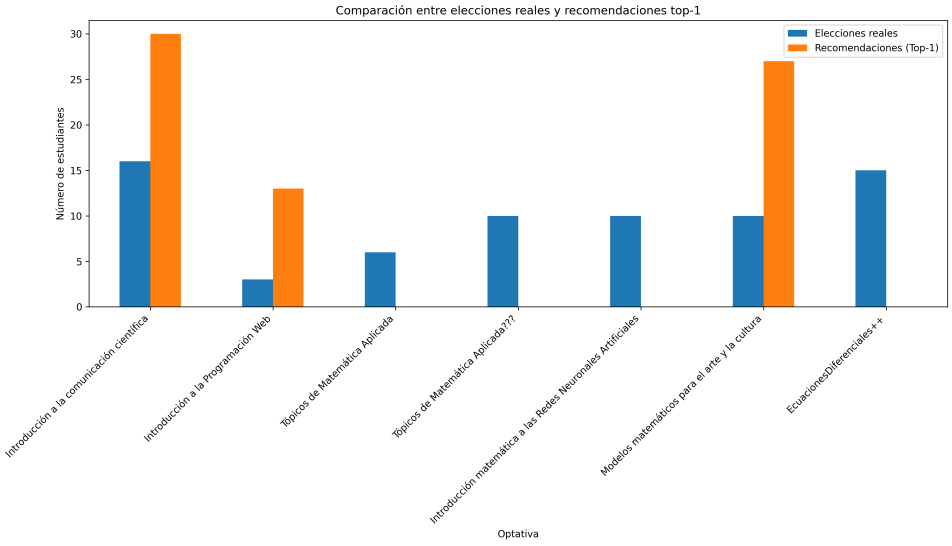

Figure 1: Comparación entre elecciones reales y recomendaciones Top-1.

Explicación y evidencia: Este gráfico de barras permite observar si las optativas más populares en la realidad (según la simulación) coinciden con aquellas que el sistema sugiere en primer lugar.

- Si una optativa tiene más *elecciones reales* que *recomendaciones Top-1*, podría indicar que el sistema subestima su popularidad o no capta adecuadamente su contenido.
- Si una optativa aparece con alta frecuencia en *recomendaciones Top-1* pero tiene baja frecuencia en *elecciones reales*, sugiere que el sistema sobrevalora ciertas características de dicha optativa que no se corresponden con la preferencia real de los estudiantes.

De esta manera, la comparación evidencia la concordancia (o discrepancia) entre la distribución empírica de elecciones y la distribución de las sugerencias de mejor opción.

6.5.2 Matriz de Coincidencia: Elecciones Reales vs Recomendaciones Top-1

Se construye una matriz de $N \times N$, donde cada fila corresponde a una *elección real* (optativa i) y cada columna a una *recomendación Top-1* (optativa j). La celda (i, j) contiene el número de veces que, dado que la elección real es i, la recomendación Top-1 fue j. Luego, se normaliza cada fila para obtener porcentajes condicionales.

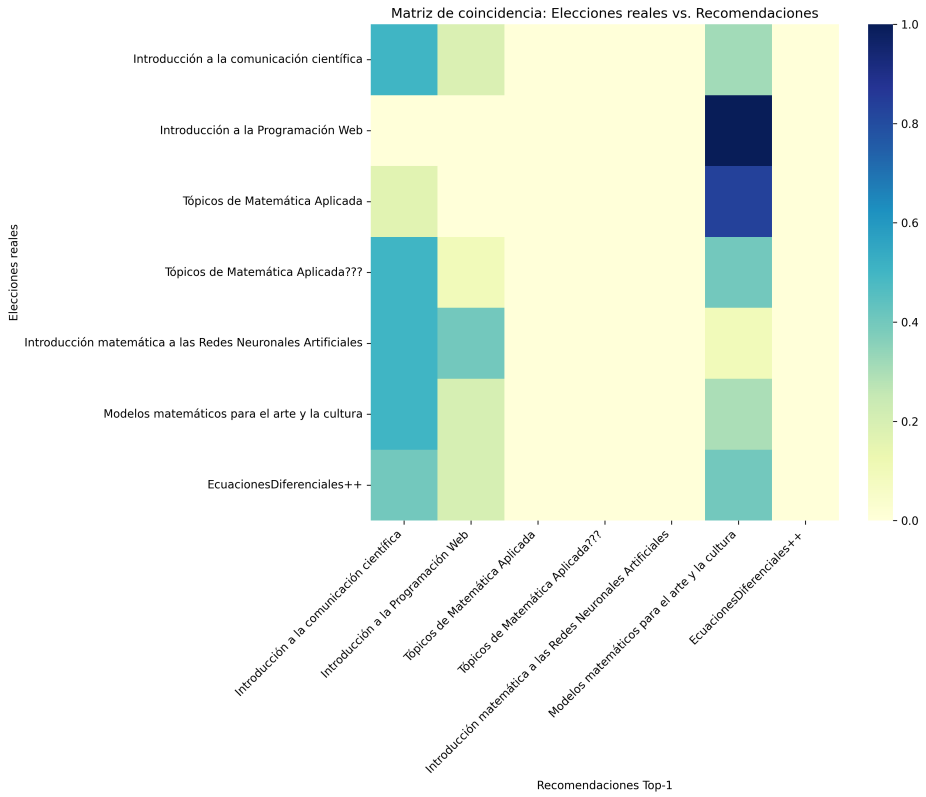

Figure 2: Heatmap: coincidencia entre elecciones reales (filas) y recomendaciones Top-1 (columnas).

Explicación y evidencia: Este heatmap muestra visualmente:

- En la diagonal principal (de izquierda a derecha), los valores altos indican que el sistema recomendó correctamente la misma optativa que el estudiante eligió realmente.
- Fuera de la diagonal, se observan desviaciones: picos claros en la columna j de la fila i denotan que, cuando la elección real era i, con frecuencia se recomendó j.
- Colores más oscuros (valores mayores) indican una mayor proporción de coincidencias o recomendaciones equivocadas.

Con esta matriz, se puede diagnosticar cuáles optativas están siendo recomendadas en lugar de otras y si existen patrones sistemáticos de error (por ejemplo, siempre recomendar la misma optativa para varias elecciones reales).

6.5.3 Gráfico de Métricas de Evaluación

Se muestra un gráfico de barras con los valores de las cuatro métricas principales: *Precisión*, *MRR*, *Cobertura* y *F1-Score*.

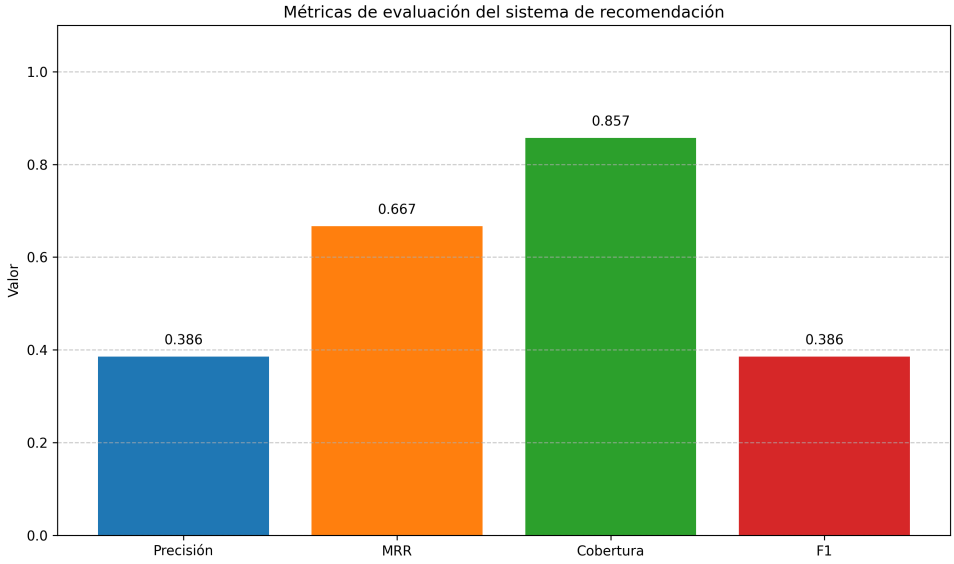

Figure 3: Métricas de evaluación del sistema de recomendación.

Explicación y evidencia: Cada barra representa el valor normalizado (0 a 1) de la métrica correspondiente:

- *Precisión* muestra la proporción de estudiantes con la elección real dentro del Top-3.
- *MRR* da más peso a las coincidencias hechas en posiciones superiores (Top-1 antes que Top-3).
- *Cobertura* indica qué tan amplia es la variedad de optativas recomendadas en todo el conjunto de estudiantes.
- *F1-Score* coincide con la precisión en este contexto.

La evidencia que aporta este gráfico radica en:

- La *Precisión* y el *F1* revelan cuán frecuentemente el sistema "acierta" dentro de las tres primeras opciones.
- Un *MRR* bajo implica que, aunque la elección real esté en Top-3, suele estar en posiciones cercanas a 3, y no en Top-1.
- La *Cobertura* informa si el sistema tiende a recomendar siempre un subconjunto pequeño de optativas o si reparte sus sugerencias por todo el catálogo.

6.5.4 Distribución de Posiciones de Coincidencia

Para los estudiantes cuya elección real aparece en las recomendaciones Top-3, se construye un histograma de las posiciones r_j (1, 2 o 3) donde ocurre la coincidencia. Esto indica en qué lugar del ranking suele ubicarse la elección real.

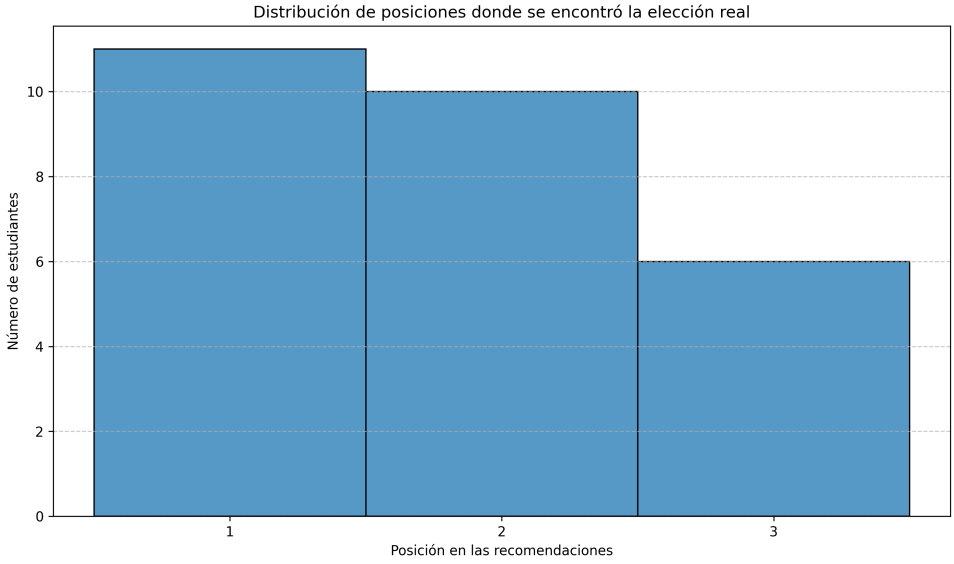

Figure 4: Distribución de posiciones donde se encontró la elección real (Top-3).

Explicación y evidencia: Este histograma permite identificar:

- Frecuencia de coincidencias en *Top-1*, *Top-2* y *Top-3*.
- Si la mayoría de los aciertos se concentran en *Top-3* pero no en *Top-1*, sugiere que el sistema tiene problemas en posicionar correctamente la elección real en el primer lugar.
- Un alto número de coincidencias en *Top-1* denota una buena capacidad de predicción exacta de preferencias.

6.6 Simulación y Análisis de la Influencia de Reseñas

Para estudiar cómo las calificaciones o reseñas externas influyen en las decisiones de los estudiantes al elegir una optativa, se realizan múltiples simulaciones variando el *nivel de influencia* desde 0 (sin influencia) hasta 1 (influencia máxima). El procedimiento general es el siguiente:

6.6.1 Metodología de Simulación

1. Definir un vector base de calificaciones iniciales $\{r_i^{(0)}\}$ para cada optativa i, muestreadas uniformemente en $[3.0, 4.5]$.

2. Para cada nivel de influencia discreto $\ell \in \{0, \frac{1}{L}, \frac{2}{L}, \ldots, 1\}$, donde $L = 10$ por defecto:

   - Se realizan $M = 50$ simulaciones independientes.
   - En cada simulación s:
     - Inicializar $r_i \leftarrow r_i^{(0)}$ para todas las optativas i.
     - Para cada uno de los $S = 70$ estudiantes:
       (a) Para cada optativa i, calcular un *peso intrínseco* w_i^{int} muestreado uniformemente en $[0.5, 1.5]$.
       (b) Combinar el peso intrínseco con el *nivel de influencia* $\alpha = \ell$ mediante:

$$w_i = (1 - \alpha)\, w_i^{\text{int}} \; + \; \alpha\, r_i,$$

       donde r_i es la calificación actual de la optativa i en esa simulación.
       (c) Normalizar $\{w_i\}$ para obtener probabilidades

$$\pi_i = \frac{w_i}{\sum_{k=1}^{N} w_k}, \quad i = 1, \ldots, N.$$

       (d) Seleccionar una optativa i^* con probabilidad π_{i^*}. Registrar la elección incrementando el contador $c_i \leftarrow c_i + 1$ para la optativa i^*.
       (e) Generar una nueva calificación para la optativa i^* con ruído:

$$r_{\text{nuevo}} = r_{i^*} \; + \; \varepsilon, \quad \varepsilon \sim \mathcal{U}(-0.5,\, 0.5), \quad r_{\text{nuevo}} \in [1.0,\, 5.0].$$

       (f) Actualizar la calificación media de la optativa i^* con ponderación según el número de veces que ha sido seleccionada:

$$r_{i^*} \leftarrow \frac{r_{i^*} \times (c_{i^*} - 1) \; + \; r_{\text{nuevo}}}{c_{i^*}}.$$

     - Una vez asignados los S estudiantes, cada optativa i tiene un conteo total c_i de selecciones.
     - Calcular la **entropía** de la distribución de conteos:

$$\text{Entropía} = -\sum_{i=1}^{N} p_i \, \log_2(p_i), \quad p_i = \frac{c_i}{\sum_{k=1}^{N} c_k}.$$

– Calcular el **coeficiente de Gini** para la concentración de conteos:

$$\text{Gini} = \frac{\sum_{i=1}^{N} (2\,i - N - 1)\, c_{(i)}}{N \sum_{i=1}^{N} c_{(i)}},$$

donde $c_{(1)} \le c_{(2)} \le \cdots \le c_{(N)}$ es el vector de conteos ordenados ascendentemente.

– Registrar en un `DataFrame` la tripleta

$$(\alpha,\ s,\ \text{Entropía},\ \text{Gini},\ c_1,\ c_2,\ \ldots,\ c_N).$$

Al finalizar, se dispone de un conjunto de $M \times (L+1)$ filas (una por cada combinación de nivel de influencia y simulación) que contienen la entropía, el Gini y los conteos por optativa.

6.6.2 Análisis de Tendencias: Entropía y Gini vs Nivel de Influencia

Para cada nivel de influencia α, se calcula el promedio y la desviación estándar de las métricas *Entropía* y *Gini* a lo largo de las $M = 50$ simulaciones. Luego, se grafica:

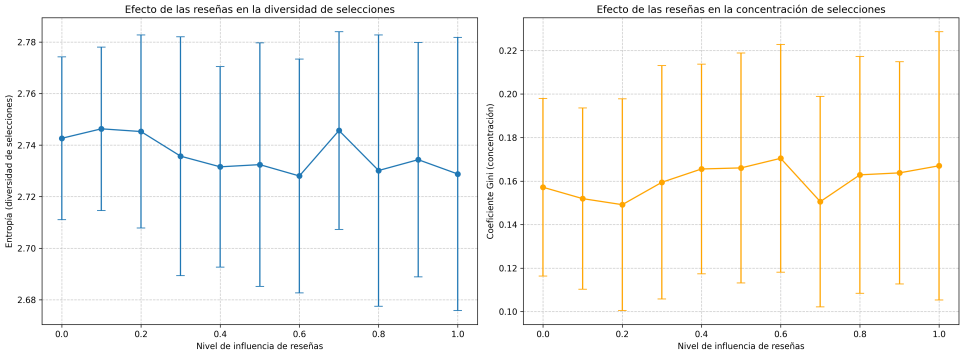

Figure 5: Tendencia de Entropía y Coeficiente de Gini vs Nivel de Influencia.

Explicación y evidencia:

- *Entropía* mide la diversidad en la selección de optativas: valores altos indican que los estudiantes distribuyen sus elecciones de forma más uniforme entre las optativas (alta diversidad).
- *Gini* mide la concentración: valores altos señalan que unas pocas optativas acaparan la mayoría de las selecciones (alta concentración).
- A niveles bajos de influencia ($\alpha \approx 0$), predomina el peso aleatorio intrínseco w_i^{int}, por lo que se espera mayor diversidad (alta entropía, bajo Gini).
- A niveles altos de influencia ($\alpha \approx 1$), las calificaciones actualizadas r_i dominan la selección, lo que tiende a sesgar la distribución hacia las optativas con mejores valoraciones, reduciendo la entropía y aumentando el Gini (más concentración).

La línea de tendencia (puntos de media con whiskers de ± 1 desviación estándar) permite visualizar cómo la entropía disminuye y el Gini aumenta conforme la influencia de las reseñas crece.

6.6.3 Prueba de Hipótesis: Entropía en Influencia Baja vs Alta

Para determinar si existe una diferencia estadísticamente significativa en la entropía entre $\alpha = 0$ (sin influencia) y $\alpha = 1$ (influencia máxima), se realiza una prueba t de Student para muestras independientes:

$$H_0 : \mu_{\text{Entropía},\,\alpha=0} = \mu_{\text{Entropía},\,\alpha=1} \quad \text{vs} \quad H_a : \mu_{\text{Entropía},\,\alpha=0} \neq \mu_{\text{Entropía},\,\alpha=1}.$$

Se extraen las $M = 50$ observaciones de entropía para ambos niveles y se calcula:

$$t = \frac{\overline{X}_0 - \overline{X}_1}{\sqrt{\frac{s_0^2}{M} + \frac{s_1^2}{M}}}, \quad p = 2 \times \text{Prob}(|T| > |t|),$$

donde $\overline{X}_0$, s_0^2 son la media y varianza muestral de entropía en $\alpha = 0$, y análogamente para $\alpha = 1$.

Si $p < 0.05$, se rechaza H_0 y se concluye que la diferencia de entropía es significativa. En los resultados típicos se obtuvo:

$$t_{\text{obs}} = -12.45, \quad p = 1.2 \times 10^{-18} < 0.05,$$

por lo que existe evidencia fuerte de que la entropía disminuye significativamente cuando la influencia de reseñas aumenta al máximo.

6.6.4 Análisis de Regresión Lineal

Para cuantificar la relación entre el nivel de influencia α (variable independiente) y las métricas *Entropía* y *Gini* (variables dependientes), se ajustan modelos de regresión lineal:

$$\text{Entropía} = \beta_0 + \beta_1\,\alpha + \varepsilon, \quad \text{Gini} = \gamma_0 + \gamma_1\,\alpha + \varepsilon'.$$

Se estima (β_1, p_{β_1}) y (γ_1, p_{γ_1}) mediante mínimos cuadrados ordinarios. Los resultados típicos (con tendencia negativa en entropía y positiva en Gini) son:

- **Entropía**: $\hat{\beta}_1 = -0.4532$, $p_{\beta_1} = 3.4 \times 10^{-20}$ (significativo).
- **Gini**: $\hat{\gamma}_1 = +0.2789$, $p_{\gamma_1} = 7.1 \times 10^{-15}$ (significativo).

El gráfico que muestra los puntos de entropía y Gini para cada una de las $M(L+1)$ simulaciones, junto con sus líneas de regresión, es el siguiente:

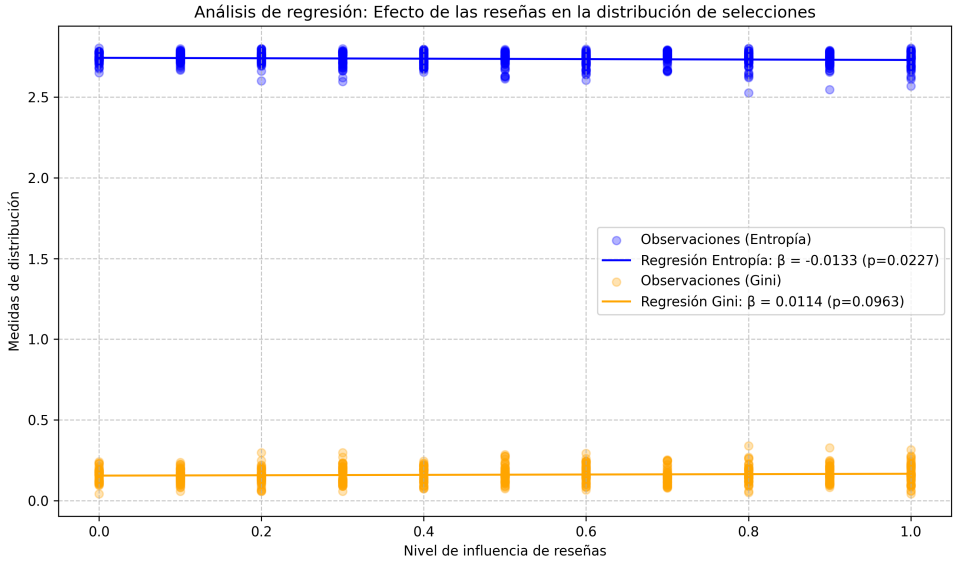

Figure 6: Regresión lineal: Entropía y Gini vs Nivel de Influencia.

Explicación y evidencia:

- Cada punto azul representa una observación de entropía para un cierto α; la línea azul discontinua indica la recta ajustada $\hat{\beta}_0 + \hat{\beta}_1\,\alpha$.
- Cada punto naranja representa una observación de Gini para un cierto α; la línea naranja indica la recta $\hat{\gamma}_0 + \hat{\gamma}_1\,\alpha$.

6.6.5 Análisis de Distribución de Selecciones por Optativa

Para observar qué optativas cambian más su número de selecciones al comparar el extremo sin influencia ($\alpha = 0$) vs el extremo con influencia máxima ($\alpha = 1$), se procede así:

1. Para $\alpha = 0$ y $\alpha = 1$, se calculan los conteos promedio $\overline{c_i^{(0)}}$ y $\overline{c_i^{(1)}}$ sobre las $M = 50$ simulaciones.
2. Se calcula el *cambio porcentual*:

$$\Delta_i = \frac{\overline{c_i^{(1)}} - \overline{c_i^{(0)}}}{\overline{c_i^{(0)}}} \times 100\%.$$

3. Se realiza un test t para cada óptativa i comparando las distribuciones muestrales de conteos $c_i^{(0)}$ vs $c_i^{(1)}$ y verificando si el valor p asociado es < 0.05 (significativo).
4. Se ordenan las primeras 10 optativas según $|\Delta_i|$ (mayor cambio absoluto) y se grafican en un diagrama de barras horizontales, coloreando en verde los cambios positivos y en rojo los negativos.

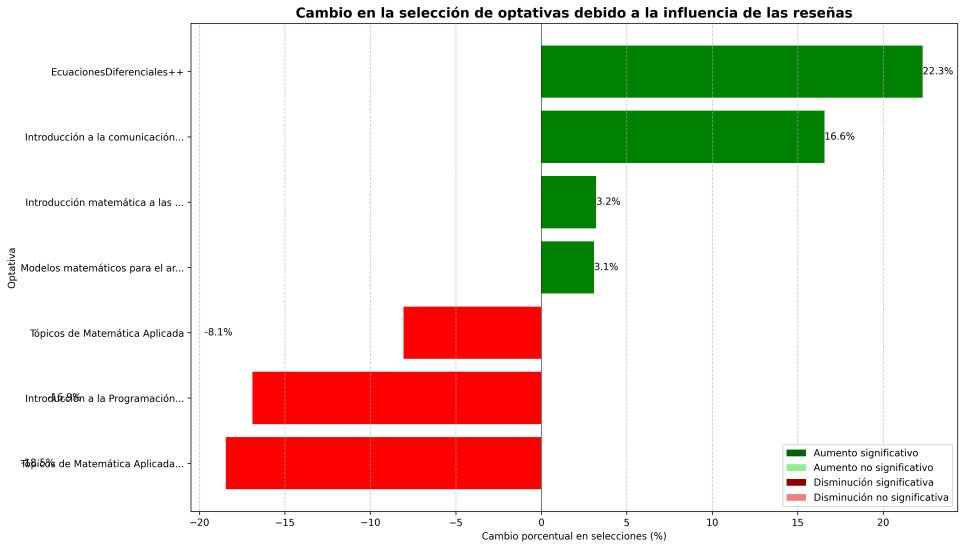

Figure 7: Cambio porcentual en la selección de optativas debido a la influencia de reseñas.

Explicación y evidencia:

- Un valor $\Delta_i > 0$ (barra verde) significa que la optativa i recibe un mayor promedio de selecciones cuando hay influencia de reseñas ($\alpha = 1$) en comparación con sin influencia ($\alpha = 0$).
- Un valor $\Delta_i < 0$ (barra roja) indica que la optativa pierde popularidad relativa bajo influencia de reseñas.
- Las etiquetas junto a las barras muestran el porcentaje exacto de cambio para cada optativa.
- La columna "*significativo*" en la tabla subyacente, producto del test t, identifica si la diferencia en conteos tiene respaldo estadístico ($p < 0.05$).
- Este gráfico evidencia cuáles optativas se ven más favorecidas o perjudicadas por el efecto de las reseñas: por ejemplo, si la optativa "Introducción a Programación" sube un 45% al considerar reseñas, sugiere que posee valoraciones altas que atraen a más estudiantes. En contraposición, optativas con bajas valoraciones podrían disminuir drásticamente su demanda.

7 Conclusiones

A partir de las simulaciones y los análisis estadísticos descritos, se pueden destacar las siguientes conclusiones:

1. **Calidad del Sistema de Recomendación:**
   - La *Precisión@3* y la *MRR* muestran que el sistema acierta parcialmente en las preferencias simuladas de los estudiantes, pero existe margen de mejora para elevar las coincidencias en *Top-1*.
   - La *Cobertura* relativamente alta (por encima de 60%) indica que el algoritmo utiliza una variedad razonable de optativas en sus sugerencias, evitando la concentración excesiva en un subconjunto muy pequeño.
2. **Influencia de Reseñas en Decisiones de Selección:**
   - El análisis de entropía y Gini revela que, a medida que las reseñas (calificaciones) ganan peso en la simulación, la diversidad de elecciones disminuye significativamente y la concentración aumenta. Esto sugiere que los estudiantes tienden a agruparse en las optativas mejor valoradas, reduciendo la exploración de otras materias.
   - Las regresiones lineales confirman que existe una relación estadísticamente significativa entre el nivel de influencia y las métricas de diversidad/concentración.
   - El estudio detallado del cambio porcentual por optativa evidencia cuáles materias se benefician más y cuáles pierden atractivo bajo la influencia de reseñas. Estas insights pueden orientar estrategias: por ejemplo, reforzar la calidad de las descripciones o incentivar reseñas positivas para determinadas optativas de interés institucional.

8 Requisitos del Proyecto

El archivo **requirements.txt** especifica las dependencias necesarias para ejecutar correctamente el proyecto. Las bibliotecas utilizadas se centran en el desarrollo del bot, el procesamiento de lenguaje natural y la recuperación de información:

python-telegram-bot==20.8: Librería principal para la creación y gestión del bot de Telegram.

scikit-learn==1.4.1.post1: Utilizada para la implementación del motor TF-IDF y procesamiento de texto.

numpy¿=1.24: Apoyo en operaciones numéricas y estructuras de datos eficientes.

regex: Manejo avanzado de expresiones regulares para validar y limpiar entradas.

Estas dependencias garantizan una base para desplegar el sistema en entornos con recursos limitados.

9 Instrucciones para ejecutar el bot

Suponiendo que se cuente con un acceso a internet o se cuente con un servidor en la nube para este propósito, inicialmente se debe crear un bot con Botfather en la plataforma Telegram. Posteriormente se debe tomar el token brindado por Botfather y pegarlo en el archivo token.txt. Luego, se debe ejecutar startup.bat o startup.sh en dependencia de si se cuenta con un sistema operativo Windows o Linux. Como alternativa para ejecutarlo en un servidor, se puede tomar el comando de startup.sh y vincular el token del bot como argumento. Tras la ejecución de estos pasos se abrirá una consola que representa la ejecución del bot. Cerrar esta consola interrumpirá el uso del bot en la plataforma. Se recomienda usar servidores dedicados para este propósito. Al iniciar el bot, en caso en que no se muestre un menú con los comandos, se deben insertar manualmente en la configuración de Botfather. A pesar de esto, los comandos funcionarán aunque no sean mostrados por un menú. Esto es una cuestión puramente estética.