

---

## Informe Final: Análisis Exploratorio de Datos del DataSet CLINC150

Docente: [Ana María Cuadros](#) Valdivia  
Alumna: Cecilia del Pilar Vilca Alvites

### 1. Hipótesis iniciales:

- 1.1. Motivación: describe cómo se originaron sus hipótesis y las razones para elegirlas

Nuestro proyecto de investigación surgió a partir de un desafío clave en el campo del procesamiento del lenguaje natural (PLN): la agrupación automática de intenciones de usuario. Aunque los modelos de lenguaje grandes (LLMs) destacan por su capacidad para capturar el contexto y generar representaciones de texto ricas (embeddings), notamos que al aplicarlos a conjuntos de datos reales de intenciones, los resultados eran frecuentemente difíciles de interpretar. Esta opacidad, típica de los métodos de clustering basados en LLMs, despertó nuestra curiosidad. Comprendimos que contar con embeddings potentes no era suficiente por sí solo; necesitábamos una forma de entender con claridad por qué ciertas intenciones se agrupaban entre sí — y, sobre todo, por qué otras no lo hacían.

Esta primera observación nos llevó a examinar más de cerca las características propias del conjunto de datos CLINC150, ampliamente utilizado en tareas de clasificación de intenciones. Durante esta exploración preliminar, identificamos dos problemas persistentes que complican la agrupación automática: la alta granularidad de sus categorías y la ambigüedad semántica entre muchas de ellas. Ambos factores dificultan que incluso los modelos más avanzados logren agrupaciones coherentes y fácilmente interpretables.

Estas observaciones iniciales dieron lugar a nuestras tres primeras hipótesis:

Hipótesis 1: Al visualizar los embeddings, notamos un fuerte solapamiento entre intenciones diferentes, lo que sugiere que la alta granularidad del dataset genera patrones complejos y entrelazados en el espacio semántico.

Hipótesis 2: Detectamos casos donde consultas idénticas o muy similares estaban etiquetadas con intenciones distintas, revelando una ambigüedad que afecta directamente la pureza del clustering.

Hipótesis 3: Concluimos que, debido a la combinación de alta granularidad y similitud conceptual, obtener agrupamientos claros es un reto inherente al

propio dataset, más allá de las limitaciones de cualquier modelo o algoritmo específico.

Estas hipótesis reflejan problemas fundamentales del CLINC150 que requieren herramientas de análisis visual para ser realmente comprendidos y abordados.

1.2. Exprese sus hipótesis en forma de pregunta (sea claro y conciso)

Hipótesis 1: ¿La **alta granularidad de las intenciones de usuario** en el conjunto de datos conduce a un **solapamiento semántico significativo** entre categorías, manifestándose en patrones de agrupamiento complejos?

Hipótesis 2: ¿Existen **instancias de ambigüedad semántica** en las consultas de usuario que provocan que frases idénticas o muy similares se asocien a **intenciones fundamentalmente distintas**?

Hipótesis 3: ¿La **ausencia de límites claros y bien definidos** entre intenciones semánticamente relacionadas en el espacio de *embeddings* dificulta su **separación efectiva** para el agrupamiento, más allá del simple solapamiento?

1.3. Plan de análisis:

Para investigar las hipótesis planteadas y obtener una comprensión profunda de la granularidad, ambigüedad y los desafíos del agrupamiento en el dataset CLINC150, se diseñó y ejecutó un plan de análisis exploratorio de datos (EDA) centrado en la visualización interactiva y el análisis cuantitativo. Los pasos clave fueron los siguientes:

1. Carga y Preparación de Datos:

- Se consolidaron los subconjuntos de datos (tsne\_train\_data.csv, tsne\_val\_data.csv, tsne\_test\_data.csv) en un único DataFrame unificado.
- Se asignó un id único a cada registro para facilitar el seguimiento y la identificación de puntos en el dashboard interactivo.
- Se realizaron verificaciones preliminares de la calidad de los datos, asegurando la ausencia de valores nulos críticos en las columnas de tsne\_x, tsne\_y, intent y text.

2. Reducción de Dimensionalidad (t-SNE):

Se utilizó la técnica de reducción de dimensionalidad t-SNE (t-distributed Stochastic Neighbor Embedding) para proyectar los embeddings de alta dimensión de las consultas de usuario a un espacio bidimensional (tsne\_x, tsne\_y). Esta proyección fue crucial para la visualización, permitiendo observar patrones de agrupamiento y solapamiento entre intenciones en un plano 2D.

3. Análisis de Agrupamiento con K-Means:

Se aplicó el algoritmo de clustering K-Means a las coordenadas tsne\_x y tsne\_y. Este paso fue fundamental para generar agrupaciones automáticas y poder compararlas con las intenciones reales. El número de clústeres (K) se hizo configurable en el dashboard (con un valor inicial de 30) para explorar cómo variaciones en este parámetro impactan en la formación de los clústeres y su alineación con las intenciones reales.

#### 4. Dashboard Interactivo para Visualización y Exploración:

Se implementó una aplicación web interactiva utilizando la biblioteca Dash de Plotly. Esta herramienta se diseñó para facilitar la exploración dinámica de los datos y la validación visual de las hipótesis.

Visualización Principal (Scatter Plot): El gráfico de dispersión mostró los puntos de datos en el espacio t-SNE. Se implementó la capacidad de:

- Colorear los puntos por su "Intención Real" o por el "ID de Clúster" asignado por K-Means, permitiendo una comparación directa y visual de ambas estructuras.

Panel de Detalles de Selección: Una característica clave fue la capacidad de seleccionar un subconjunto de puntos en el scatter plot (mediante click o arrastre) y obtener un desglose detallado de su composición, incluyendo:

- La distribución de intenciones reales y de IDs de clúster dentro de la selección.
- Las consultas de texto individuales de los puntos seleccionados, crucial para identificar la ambigüedad semántica.
- Una nube de palabras clave para resumir los términos más frecuentes en el texto de los puntos seleccionados, ayudando a entender la temática subyacente.

#### 5. Análisis de Granularidad y Ambigüedad (Matriz de Confusión):

- Se generó una Matriz de Confusión interactiva (heatmap) que mapea las intenciones reales contra los IDs de clúster de K-Means. Esta visualización fue esencial para cuantificar y visualizar el grado de solapamiento y fragmentación.
- La normalización de la matriz (porcentaje por fila) permitió identificar claramente:
  - Granularidad: Intenciones reales que se distribuyen en múltiples clústeres de K-Means (una fila con porcentajes significativos en varias columnas).
  - Ambigüedad/Impureza: Clústeres de K-Means que contienen una mezcla de múltiples intenciones reales (una columna con porcentajes significativos en varias filas).

#### 6. Evaluación Cuantitativa del Clustering:

Se calcularon y mostraron métricas estándar de calidad de clustering, específicamente el Silhouette Score y el Davies-Bouldin Index. Estas métricas proporcionaron una medida objetiva de la coherencia y separación de los clústeres generados, sirviendo como respaldo cuantitativo a las observaciones visuales.

## 2. Fuente de datos:

### 2.1. Fuente:

El dataset CLINC150 fue publicado en 2019 por investigadores de la Universidad de Michigan. Se obtuvo como un conjunto de datos diseñado para la tarea de clasificación de intenciones en sistemas de diálogo.

El conocimiento involucrado en el dataset CLINC150 proviene del campo del Procesamiento de Lenguaje Natural (NLP). El problema computacional que se busca resolver con este conjunto de datos es la clasificación de intenciones en sistemas de diálogo, lo que implica que un modelo de lenguaje aprenda a identificar correctamente la intención detrás de una frase de usuario.

Las variables capturadas son "consulta" (text) e "intención" (intent). La consulta representa lo que el usuario dijo, y la intención representa lo que el sistema debe entender que el usuario quiere hacer. Estas variables son importantes porque encapsulan la interacción mínima de un usuario con un sistema de diálogo orientado a tareas.

### REFERENCES

- [1] R. Peng, Y. Dong, G. Li, D. Tian, and G. Shan, "TextLens: Large language models-powered visual analytics enhancing text clustering," *Journal of Intelligent & Fuzzy Systems*, DOI: 10.1007/s12650-025-01043-y, Feb. 2025.
- [2] A. Petukhova, J. Carvalho, and N. Fachada, "Text Clustering with Large Language Model Embeddings," *International Journal of Cognitive Computing in Engineering*, vol. 6, pp. 100–108, Dec. 2025.
- [3] N. Arias, P. Singh, and A. B. Imbert, "Visual Analytics for Fine grained Text Classification Models and Datasets," *arXiv preprint arXiv:2405.02980*, 2024.
- [4] L. K. Miller and C. P. Alexander, "Human-interpretable clustering of short text using large language models," *Royal Society Open Science*, vol. 12, no. 2, pp. 241088, 2025.
- [5] S. Hamada, "Processing of Semantic Ambiguity Based on Words Ontology," *Journal of Computer Science*, vol. 16, no. 1, pp. 1–9, 2020

### 2.2. Descripción:

El dataset CLINC150 contiene información textual en inglés. El tamaño total del dataset en sus subconjuntos principales (train + val + test) es de 22,500 registros.

#### a) A nivel de atributos:

text: La variable text contiene las consultas o enunciados realizados por los usuarios y está representada como un dato de tipo object (cadena de texto). No presenta valores nulos, lo que garantiza la completitud del conjunto de datos. Es una variable cualitativa nominal, ya que cada texto constituye una unidad única sin un orden inherente.

El rango de valores es amplio debido a la alta diversidad en las consultas; se identificaron 10 textos duplicados con posibles intenciones distintas y 1 registro completamente duplicado. No se describen medidas estadísticas tradicionales debido a su naturaleza textual, pero su relevancia semántica la convierte en una característica fundamental para que los modelos de lenguaje (LLMs) puedan extraer significado e interpretar la intención del usuario.

**Intent:** Representa la intención asociada a cada consulta del usuario y es de tipo object (texto categórico), sin valores nulos. Es una variable categórica nominal que actúa como variable objetivo en este problema de clasificación supervisada, siendo fundamental tanto para evaluar cualitativamente los resultados del clustering como para orientar el proceso de depuración. Posee 150 categorías únicas, incluida una clase especial llamada `out_of_scope` (OOS), y presenta una distribución altamente uniforme: cada intención aparece aproximadamente 100 veces en el conjunto de entrenamiento y 20 veces en los conjuntos de validación y prueba, lo que garantiza un buen balance de clases.

**text\_length:** Indica la longitud, en número de caracteres, de cada consulta y es de tipo int64 (entero), sin valores nulos. Es una variable cuantitativa discreta útil en el análisis exploratorio, aunque no aporta información predictiva directa sobre la intención semántica del texto. Su rango de valores va desde 2 hasta 136 caracteres en el conjunto de entrenamiento, con una distribución relativamente simétrica. La media y la mediana se sitúan cerca de los 39 caracteres, y la desviación estándar, de aproximadamente 15-16 caracteres, refleja una variabilidad moderada en la longitud de las consultas.

b) **A nivel de registros:**

Cada registro en el dataset CLINC150 representa una consulta individual de un usuario (`text`) acompañada por una etiqueta de intención (`intent`) que describe la acción o necesidad expresada en esa consulta. Estas etiquetas corresponden a 150 clases distintas, incluyendo una clase especial llamada `out_of_scope` (OOS) para consultas fuera de los dominios definidos. Las etiquetas se diferencian principalmente por la intención semántica, aunque algunas clases pueden tener significados muy cercanos, generando solapamiento y ambigüedad semántica.

El dataset opera con un nivel de granularidad alto (fino), ya que cada fila representa una unidad mínima y completa de interacción: una única consulta del usuario con una sola intención. Este alto nivel de granularidad permite análisis detallados de la intención individual, pero también presenta desafíos como la ambigüedad del lenguaje y la similitud entre clases.

**c) Relación entre atributos:**

Existe una ausencia de relación lineal fuerte entre la longitud de la consulta (text\_length) y la intención codificada numéricamente (intent). La correlación es cercana a cero (-0.0507). Esto sugiere que la semántica del mensaje no depende significativamente de su extensión. Esta "falta de relación" es importante porque subraya la necesidad de enfoques basados en la semántica (como los LLMs) para comprender la intención, en lugar de características superficiales del texto.

Por otro lado, los gráficos t-SNE (de los embeddings generados a partir de text) muestran una fuerte agrupación semántica de consultas de la misma intención en el espacio de embeddings, lo que valida la capacidad de los LLMs para capturar el significado. Sin embargo, también se observa una superposición visual entre intenciones semánticamente cercanas, lo que plantea un desafío en la separación limpia de clases por algoritmos de clustering.

**Terminología especial y columnas importantes:**

1. text (Consulta): La frase de texto del usuario. Importante para el lector para entender el contexto y para la hipótesis como la fuente de los embeddings.
2. intent (Intención): La categoría a la que pertenece la consulta. Fundamental para el lector como el objetivo de la clasificación, y crucial para la hipótesis como la "verdad fundamental" para evaluar el clustering y depurar el modelo.
3. text\_length (Longitud de la Consulta): Longitud en caracteres de la consulta. Útil para el análisis exploratorio, pero no una característica predictiva directa.
4. embeddings (Representaciones Vectoriales del Texto): Son las características derivadas más importantes para el análisis y la hipótesis, ya que capturan el significado semántico del texto y son la base numérica para el clustering y las visualizaciones.

**Unidades de Medida Diferentes:**

Las columnas text e intent son cualitativas y no tienen unidades de medida físicas. La columna text\_length se mide en caracteres. En artículos científicos, cuando se tienen variables con diferentes "unidades" o tipos, es común:

- Transformar datos cualitativos a numéricos: Para el text, esto se logra mediante la generación de embeddings que convierten el texto en vectores numéricos de alta dimensión. Para intent, a menudo se realiza una codificación numérica para fines de cálculo de correlación o métricas.
- Normalización/Estandarización: Si la longitud del texto (text\_length) fuera una característica importante, se podría normalizar para que sus

valores estén en una escala comparable con otras características numéricas (aunque en este caso, se descarta su uso como feature predictiva directa debido a la baja correlación).

### Realice un cuadro resumen de la descripción de los atributos.

TABLE I  
ATRIBUTOS PRINCIPALES DEL DATASET CLINC150 Y SU SIGNIFICADO

Atributo	Descripción	Tipo de Dato	Rango / Valores Posibles
text	Representa la <b>consulta original del usuario</b> en lenguaje natural. Es la entrada textual que el sistema debe procesar para inferir la intención.	String	Cadenas de texto que varían en longitud desde 2 hasta 136 caracteres. Ejemplos incluyen "what is my account balance" o "can you please tell me how much money I have left in my primary checking account after deducting all pending transactions?".
intent	Es la <b>etiqueta de la intención</b> subyacente asociada a la consulta de texto. Sirve como la verdad fundamental ( <i>ground truth</i> ) para la clasificación.	String	<b>150 categorías de intención</b> distintas y finamente granularizadas. Incluye intenciones como <code>pay_bill</code> , <code>transfer</code> , <code>balance</code> , <code>greeting</code> , <code>goodbye</code> , <code>translate</code> , <code>money_transfer</code> , y la categoría <code>out_of_scope</code> (OOS).
text_length	Atributo derivado que representa la <b>longitud de la consulta de texto</b> en número de caracteres. Se utiliza para análisis exploratorio.	Entero	Valores entre 2 y 136 caracteres. La mayoría de las consultas se agrupan alrededor de los 39 caracteres, con una mediana de 37.
split	Indica a qué subconjunto pertenece la instancia, utilizado para la división estándar del dataset para entrenamiento, validación y prueba de modelos.	String	<code>train</code> (entrenamiento), <code>val</code> (validación), <code>test</code> (prueba).
embeddings	Atributo derivado, no original del dataset, pero crucial para este proyecto. Son las <b>representaciones numéricas densas</b> de cada consulta de texto, generadas por un Large Language Model (LLM) (específicamente, <code>all-MiniLM-L6-v2</code> ).	Vector	Vector de 384 dimensiones. Cada valor en el vector es un número flotante.
tsne_x, tsne_y	Atributos derivados de la reducción de dimensionalidad de los <code>embeddings</code> . Representan las <b>coordenadas 2D</b> de cada consulta en un espacio visual, obtenidas mediante t-SNE, facilitando su visualización e interpretación.	Flotante	Valores que varían según la distribución en el espacio 2D, generalmente en un rango de números reales.

#### 2.3. Formato:

El formato original del dataset CLINC150 es **JSON**, con una estructura de pares ["consulta", "intención"].

<https://www.kaggle.com/datasets/hongtrung/clinc150-dataset>

#### 2.4. Transformaciones:

Para preparar los datos para el análisis de clustering y las visualizaciones en el dashboard, realicé las siguientes transformaciones clave:

##### 1. Obtención de Embeddings y Reducción de Dimensionalidad (t-SNE)

El primer y más crucial paso para convertir los datos de texto de las consultas de usuario en un formato utilizable fue representarlos numéricamente de una manera que capturara su significado semántico.

Las consultas de usuario originales son texto plano. Para aplicar algoritmos de machine learning como K-Means, que operan en espacios numéricos, es

necesario convertir estas cadenas de texto en vectores numéricos. Esto se logra mediante embeddings de texto, que son representaciones densas de palabras o frases donde la similitud semántica se traduce en proximidad en el espacio vectorial.

- Transformación: Una vez obtenidos los embeddings (que típicamente tienen cientos o miles de dimensiones), se aplicó t-Distributed Stochastic Neighbor Embedding (t-SNE). Esta técnica de reducción de dimensionalidad no lineal es fundamental para la visualización.
- Propósito: Reducir la alta dimensionalidad de los embeddings a un espacio 2D (tsne\_x, tsne\_y) que puede ser representado en un gráfico de dispersión.
- Beneficio: t-SNE es excelente para preservar las distancias locales entre puntos, lo que significa que las consultas semánticamente similares (cercanas en el espacio de embedding original) seguirán estando cerca en el espacio 2D. Esto es vital para observar el solapamiento y la granularidad.
- Formato Resultante: Las columnas tsne\_x y tsne\_y en el DataFrame.

## 2. Concatenación de Conjuntos de Datos

El dataset CLINC150 suele dividirse en conjuntos de entrenamiento (df\_train), validación (df\_val) y prueba (df\_test). Para un análisis exploratorio general y el clustering de todas las intenciones, es beneficioso tener todos los datos juntos.

- Transformación: Se realizó una concatenación vertical de los tres DataFrames (df\_train, df\_val, df\_test).
- Propósito: Crear un único DataFrame consolidado (df\_combined\_original) que incluye todas las consultas de usuario junto con sus embeddings reducidos (t-SNE), intenciones reales y el texto original.
- Formato Resultante: Un único DataFrame que contiene las columnas tsne\_x, tsne\_y, intent, text.

## 3. Asignación de ID Único

Para facilitar la trazabilidad y la selección interactiva de puntos en el dashboard, se asignó un identificador. Aunque Pandas maneja un índice, agregar un id explícito puede ser útil para el rastreo de puntos individuales, especialmente después de concatenaciones que pueden resetear o duplicar índices.

- Transformación: Se añadió una nueva columna id al DataFrame consolidado, utilizando simplemente el índice del DataFrame como identificador único para cada consulta.
- Propósito: Permitir una identificación única de cada punto de datos para su selección y consulta en el panel de detalles del dashboard.
- Formato Resultante: Una columna id numérica en el DataFrame.

#### 4. Preparación para K-Means y Visualización

El algoritmo K-Means asigna un ID numérico (entero) a cada punto, representando el cluster al que pertenece. Para que Plotly (la librería de visualización) trate estos IDs como categorías discretas (y asigne colores distintos a cada cluster en lugar de un gradiente continuo), es necesario una conversión.

- Transformación: Después de que K-Means calcula la columna `cluster_id` (que es numérica), esta se convierte explícitamente a tipo string.
- Propósito: Asegurar que los clusters se visualicen como categorías distintas con colores únicos en el scatter plot, lo cual es crucial para la interpretación. También es importante para que la matriz de confusión pueda usar estos IDs como etiquetas categóricas.
- Formato Resultante: La columna `cluster_id` como tipo string.

Estas transformaciones son esenciales para pasar de datos de texto crudos a un formato que permite tanto la aplicación de algoritmos de clustering como una visualización interactiva y significativa de los resultados.

#### 2.5. Limpieza de datos:

El análisis exploratorio del dataset CLINC150 reveló que todas las filas están completas y no contienen valores nulos. Las columnas originales (`text` e `intent`) y la derivada (`text_length`) tienen valores no nulos en todos los registros, indicando una estructura de datos limpia. Por lo tanto, no fue necesario aplicar estrategias de imputación, eliminación o combinación de datos debido a valores faltantes.

Se identificó la presencia de datos duplicados, específicamente 1 registro completamente duplicado (consulta "hey what's up" con intención "greeting"). Aunque esto indica una duplicidad exacta, su número es insignificante en un dataset de 22,500 registros. También se encontraron 10 consultas de texto duplicadas con intenciones potencialmente diferentes, lo cual es un reflejo de la ambigüedad inherente al lenguaje natural y no un error de limpieza de datos.

Se identificaron outliers en la columna `text_length` (consultas con más de 79 caracteres). Sin embargo, se determinó que no deben ser eliminados ya que son consultas reales y representativas del comportamiento natural de los usuarios, no errores de carga o registros corruptos.

En resumen, el dataset CLINC150 es un conjunto de datos de investigación bien curado y no presenta problemas "notables" de baja calidad de datos en el sentido tradicional (ej. nulos, formatos inconsistentes, errores de carga masivos). Los desafíos identificados (ambigüedad semántica y alta granularidad de intenciones con solapamiento) son inherentes a la complejidad

del lenguaje natural y al diseño de la tarea, más que problemas de "suciedad" en los datos que requieran limpieza convencional.

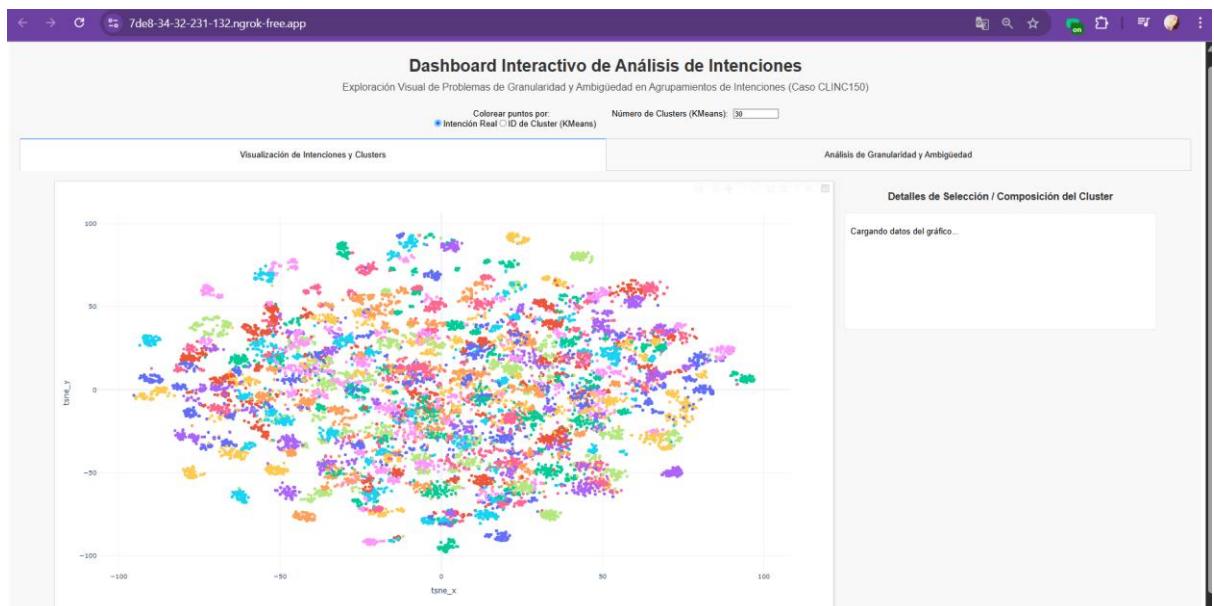
### 3. Exploración:

Para investigar las hipótesis sobre la granularidad y ambigüedad de las intenciones de usuario en el conjunto de datos CLINC150, realicé una serie de pasos exploratorios y visualizaciones. Mi objetivo fue entender cómo las intenciones se distribuyen en el espacio semántico de los embeddings y cómo el clustering K-Means captura (o no captura) estas relaciones.

Los pasos que seguí fueron:

1. Carga y Preparación de Datos: Cargué los datos de t-SNE precalculados para los conjuntos de entrenamiento, validación y prueba, y los combiné en un único DataFrame. Esto proporcionó la representación 2D de las consultas de usuario, crucial para la visualización.
2. Clustering K-Means Interactivo: Implementé un clustering K-Means sobre las coordenadas t-SNE. La interactividad me permitió ajustar el número de clusters (K) y observar en tiempo real cómo cambiaban las asignaciones de clusters.
3. Visualización Principal (Scatter Plot): Creé un scatter plot que mostraba los puntos de datos en el espacio t-SNE. Esta visualización permitía colorear los puntos tanto por su intención real (ground truth) como por el ID de cluster asignado por K-Means.
4. Panel de Detalles de Selección: Un panel lateral interactivo mostraba la composición de las intenciones y clusters cuando se seleccionaban puntos en el scatter plot, permitiendo un análisis a nivel de consulta individual.
5. Análisis de Granularidad y Ambigüedad (Pestaña Dedicada): Diseñé una pestaña específica para las métricas de calidad de clustering y la matriz de confusión. Esta pestaña se actualiza automáticamente cuando se cambia el número de clusters, lo que es vital para la exploración.

A continuación, presento las visualizaciones clave del dashboard y lo que aprendí de cada una, abordando directamente las hipótesis.

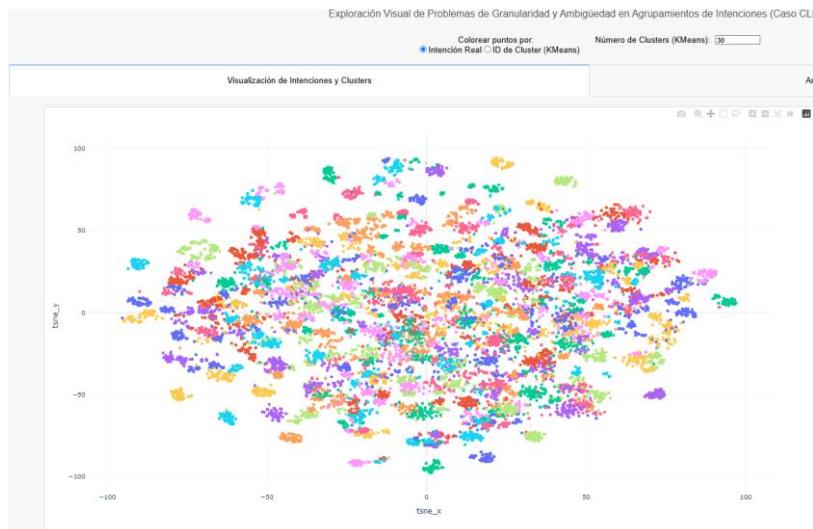


## 1. Pestaña "Visualización de Intenciones y Clusters"

Esta pestaña contiene el corazón de la exploración visual: el gráfico de dispersión de t-SNE y el panel de detalles de selección.

### 1.1 Gráfico Principal de Dispersión (Main Scatter Plot)

- **Descripción:** Este es un gráfico de dispersión 2D que muestra cada consulta de usuario como un punto, proyectado en un espacio de menor dimensión utilizando t-SNE. Los ejes (tsne\_x, tsne\_y) representan las coordenadas de esta proyección.
- **Funcionalidad Interactiva:** Puedes alternar entre colorear los puntos por su **Intención Real** (el "ground truth" del conjunto de datos) o por el **ID de Cluster (KMeans)** asignado por el algoritmo. También puedes ajustar el **Número de Clusters (KMeans)** usando el input numérico.



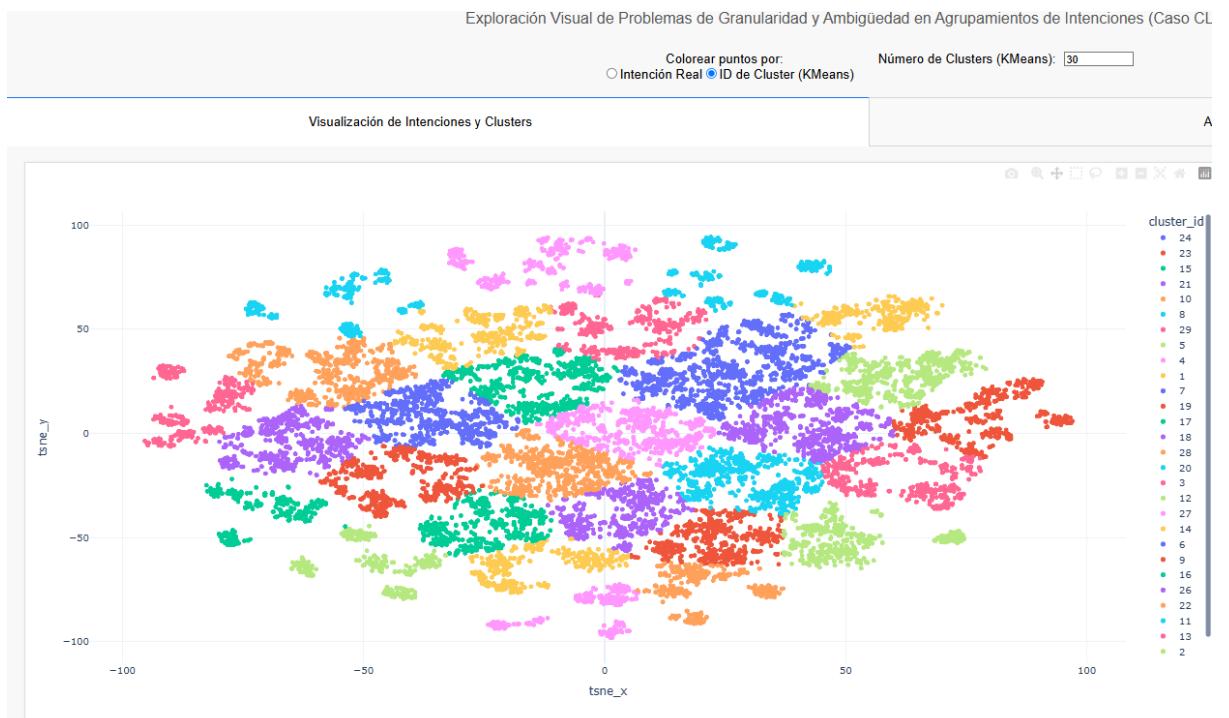
- **Información en el Hover:** Al pasar el ratón sobre un punto, se muestran detalles como la consulta de texto original, su intención real y el ID de cluster asignado. Esto es crucial para entender el contenido de puntos específicos.
- **Selección de Puntos:** Permite seleccionar uno o múltiples puntos (haciendo clic o arrastrando una caja) para analizarlos en detalle en el panel lateral.



Al colorear por Intención Real, observamos cómo las intenciones "naturalmente" se agrupan o se dispersan en el espacio semántico.

- Solapamiento Semántico (Hipótesis 1): Inmediatamente, se puede ver que muchas intenciones no forman grupos aislados y distintivos. En cambio, hay una densidad alta de colores mezclados en varias regiones del gráfico. Esto sugiere que las intenciones, aunque nominalmente distintas, comparten un significado subyacente o un léxico similar, lo que lleva a un solapamiento semántico significativo. Los "patrones de agrupamiento complejos" se manifiestan como nubes de puntos donde varios colores (intenciones) están intercalados.
- Ausencia de Límites Claros (Hipótesis 3): La dificultad para distinguir visualmente fronteras claras entre los grupos de diferentes colores apoya la idea de que los límites entre intenciones semánticamente relacionadas son difusos en el espacio de embeddings. Esto anticipa que un algoritmo de clustering como K-Means tendrá problemas para dibujar "líneas limpias" entre ellas.

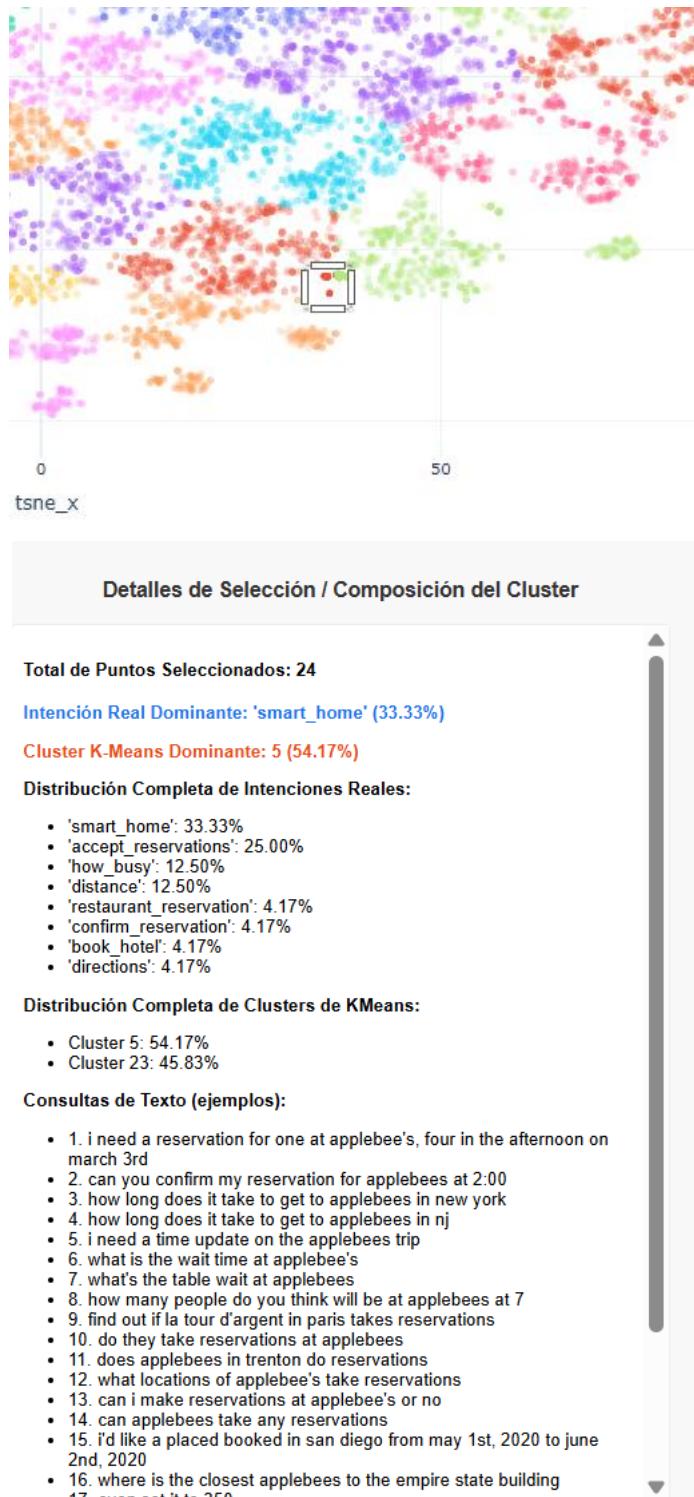
#### 1.1.2 Gráfico: Distribución por ID de Cluster (K=30)



Al cambiar a ID de Cluster, vemos cómo K-Means ha segmentado el espacio. Cada color ahora representa un cluster asignado.

- Patrones de Agrupamiento Complejos (Hipótesis 1): Si observamos los clusters individuales, a menudo notaremos que un solo color (un cluster) contiene puntos de múltiples intenciones reales (esto se confirmará con el

hover o el panel de detalles). Esto demuestra que K-Means, al tratar de encontrar grupos densos, ha agrupado intenciones semánticamente distintas pero cercanas, lo que es una consecuencia directa de la alta granularidad y el solapamiento.



### 1.1.3 Gráfico: Composición de Clusters con Diferente K (Ej. K=10, K=50)

Al variar K (por ejemplo, a 10 o 50) y alternar entre "Intención Real" y "ID de Cluster", podemos entender cómo K-Means intenta adaptarse a la estructura de los datos.

Impacto de K en la Granularidad (Hipótesis 1 y 3):

Con un K bajo (ej. 10), los clusters son muy grandes y es probable que cada uno abarque un rango muy amplio de intenciones reales, evidenciando una fuerte mezcla de intenciones y un bajo nivel de separación. Esto subraya la dificultad de separar intenciones semánticamente cercanas con pocos clusters.



Con un K alto (ej. 50, o incluso 150, el número de intenciones), los clusters se vuelven más pequeños. Aunque esto podría ayudar a separar algunas intenciones, aún observamos solapamiento. Además, un K muy alto puede llevar a que una sola intención real se divida en múltiples clusters, o que clusters muy pequeños sean creados por variaciones mínimas en el embedding, lo que apunta a la complejidad del espacio. La incapacidad de K-Means para delinejar limpiamente los límites incluso con un K alto, refuerza la idea de la "ausencia de límites claros".



## 1.2 Panel de Detalles de Selección / Composición del Cluster

Este panel se actualiza dinámicamente cuando seleccionas puntos en el gráfico de dispersión. Muestra el número total de puntos seleccionados, la intención real dominante y el cluster K-Means dominante, junto con la distribución completa de intenciones y clusters entre los puntos seleccionados, y ejemplos de consultas de texto.

- **Ambigüedad Semántica (Hipótesis 2):** Al seleccionar una región densa en el gráfico donde diferentes colores de "Intención Real" se mezclan (o donde un cluster K-Means contiene varias intenciones reales), este panel es una herramienta poderosa. Si al seleccionar un cluster particular (coloreando por cluster\_id), el panel muestra una lista significativa de "**Distribución Completa de Intenciones Reales**" con porcentajes variados, esto confirma que **ese cluster está agrupando consultas de intenciones fundamentalmente distintas**. Si además, los "ejemplos de consultas de texto" revelan frases idénticas o muy similares que fueron etiquetadas con intenciones diferentes, esto sería una prueba directa de la ambigüedad semántica en el conjunto de datos.
- **Granularidad (Hipótesis 1):** Si seleccionas un área que parece corresponder a una única "Intención Real" (coloreando por intent), pero el panel muestra que K-Means ha asignado esos puntos a **múltiples "Clusters K-Means"**, esto indica que una intención real está siendo partida en varios sub-clusters, lo que puede ser un signo de alta granularidad donde una "gran" intención tiene sub-intenciones o variaciones semánticas que K-Means está tratando de separar.

**Detalles de Selección / Composición del Cluster**

Total de Puntos Seleccionados: 24

Intención Real Dominante: 'smart\_home' (33.33%)

Cluster K-Means Dominante: 5 (54.17%)

Distribución Completa de Intenciones Reales:

- 'smart\_home': 33.33%
- 'accept\_reservations': 25.00%
- 'how\_busy': 12.50%
- 'distance': 12.50%
- 'restaurant\_reservation': 4.17%
- 'confirm\_reservation': 4.17%
- 'book\_hotel': 4.17%
- 'directions': 4.17%

Distribución Completa de Clusters de KMeans:

- Cluster 5: 54.17%
- Cluster 23: 45.83%

Consultas de Texto (ejemplos):

- 1. i need a reservation for one at applebee's, four in the afternoon on march 3rd
- 2. can you confirm my reservation for applebees at 2:00
- 3. how long does it take to get to applebees in new york
- 4. how long does it take to get to applebees in nj
- 5. i need a time update on the applebees trip
- 6. what is the wait time at applebee's
- 7. what's the table wait at applebees
- 8. how many people do you think will be at applebees at 7
- 9. find out if la tour d'argent in paris takes reservations
- 10. do they take reservations at applebees
- 11. does applebees in trenton do reservations
- 12. what locations of applebee's take reservations
- 13. can i make reservations at applebee's or no
- 14. can applebees take any reservations
- 15. i'd like a placed booked in san diego from may 1st, 2020 to june 2nd, 2020
- 16. where is the closest applebees to the empire state building

## 2. Análisis de Granularidad y Ambigüedad (Pestaña "Análisis de Granularidad y Ambigüedad")

Esta pestaña proporciona una visión más cuantitativa de la calidad del clustering y la relación entre intenciones reales y clusters.

### 2.1 Métricas de Calidad de Clustering

Se presentan los valores de **Silhouette Score**, **Davies-Bouldin Index** y **Calinski-Harabasz Index**, junto con una leyenda que explica su significado y la interpretación de sus valores. Estas métricas evalúan la calidad del agrupamiento basándose en la cohesión dentro de los clusters y la separación entre ellos.



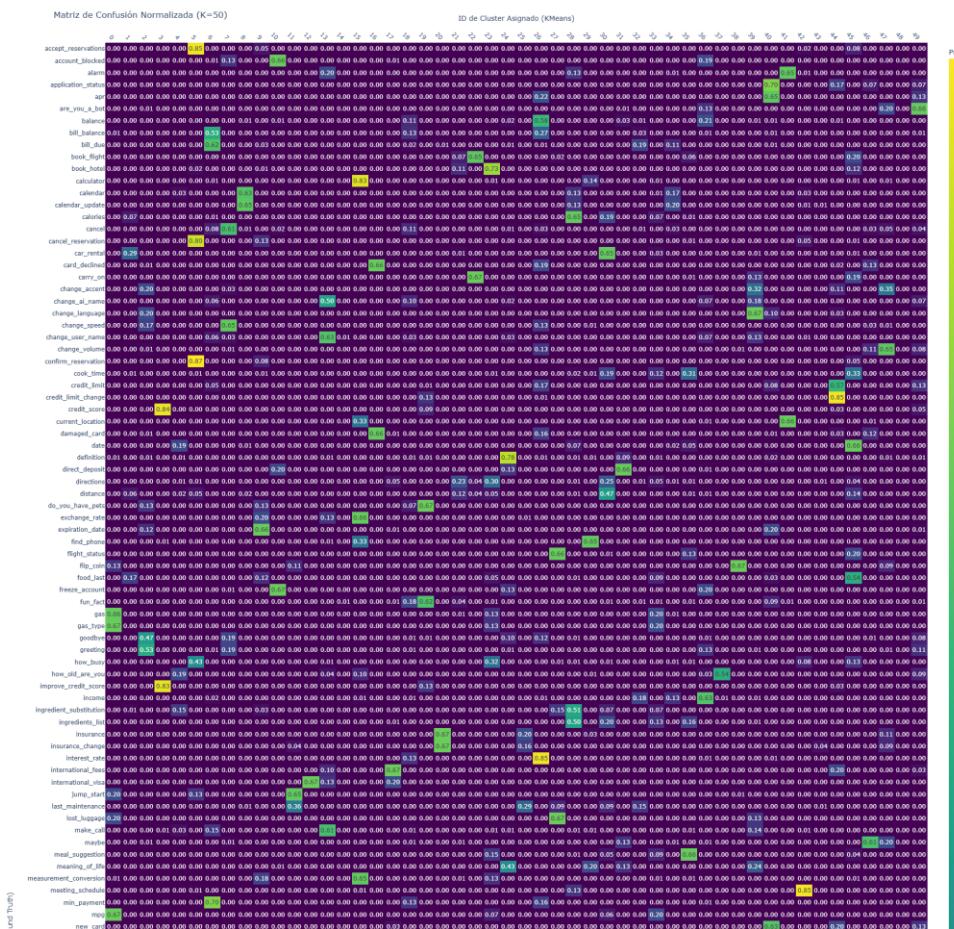
- **Solapamiento y Límites Difusos (Hipótesis 1 y 3):**
  - Un **Silhouette Score** moderado o bajo (ej., 0.3-0.5) sugiere que los puntos no están tan fuertemente asignados a sus propios clusters o que hay puntos que están casi a la misma distancia de su propio cluster que del cluster vecino. Esto es un indicador directo del **solapamiento semántico** y la **ausencia de límites claros**.
  - Un **Davies-Bouldin Index** que no es cercano a cero, combinado con un Silhouette Score modesto, refuerza la idea de que los clusters, aunque intentan agrupar, no logran una separación perfecta entre sí, lo que nuevamente apunta a la dificultad de definir límites claros entre intenciones en el espacio de embeddings.
  - El **Calinski-Harabasz Index** (que es mejor cuanto más alto) puede ser útil para comparar diferentes valores de K. Si un K particular produce un Calinski-Harabasz significativamente más alto, podría indicar una mejor separación de la varianza entre clústeres.
- **Impacto de K:** Al cambiar el "Número de Clusters (KMeans)" en la pestaña principal y luego regresar a esta pestaña, se observa cómo varían estas métricas. Por ejemplo, un aumento en K puede inicialmente mejorar el Silhouette Score (si los clusters se vuelven más compactos) pero también puede llevar a la fragmentación de intenciones si se excede el número "natural" de grupos.

### 2.2 Matriz de Confusión (Intención Real vs. Cluster ID)

Este es un mapa de calor donde las filas representan las Intenciones Reales (Ground Truth) y las columnas los IDs de Cluster Asignados por K-Means. Cada celda muestra la proporción de consultas de una intención real específica que fueron asignadas a un cluster particular (normalizada por fila). La matriz de confusión es la visualización más directa para validar las hipótesis.

- Alta Granularidad y Solapamiento (Hipótesis 1):

- Filas Dispersas: Si una fila (una Intención Real) tiene valores significativos (celdas de color intenso) distribuidos en varias columnas (múltiples IDs de Cluster), significa que esa intención real se está dividiendo entre varios clusters de K-Means. Esto es una evidencia clara de alta granularidad y que el algoritmo no logra capturar la intención como un grupo único, sino que la fragmenta. Esto apunta a la "complejidad de agrupamiento" de la hipótesis 1.
  - Columnas Mezcladas: Si una columna (un ID de Cluster) tiene valores significativos distribuidos en varias filas (múltiples Intenciones Reales), significa que ese cluster está agrupando consultas de intenciones distintas. Esto es una fuerte señal de solapamiento semántico, donde K-Means está mezclando conceptos que deberían estar separados, apoyando la Hipótesis 1.
  - Ambigüedad Semántica (Hipótesis 2): Al identificar celdas problemáticas (por ejemplo, una intención real que se distribuye ampliamente, o un cluster que agrupa muchas intenciones), podemos volver al scatter plot, seleccionar ese cluster o esa intención, y usar el panel de detalles para revisar las consultas de texto. Si esas consultas revelan frases que son idénticas o muy similares pero etiquetadas con intenciones diferentes, o que son contextualmente ambiguas, esto apoya directamente la Hipótesis 2.
  - Ausencia de Límites Claros (Hipótesis 3): La dificultad general de obtener una matriz de confusión con diagonales fuertes (cada intención en un único cluster, y cada cluster conteniendo una única intención) y valores cercanos a cero fuera de la diagonal, incluso al ajustar K, refuerza la idea de que los límites entre intenciones en el espacio de embedding no son nítidos, lo que hace inherentemente difícil su separación efectiva mediante clustering.



Este conjunto de visualizaciones interactivas proporciona una metodología robusta para explorar y, en gran medida, validar las hipótesis sobre la granularidad, el

solapamiento y la ambigüedad en el conjunto de datos de intenciones de usuario. Al manipular K y observar los resultados en el scatter plot, las métricas y, crucialmente, la matriz de confusión, se puede obtener una comprensión profunda de los desafíos inherentes a este tipo de datos.

#### 4. Conclusión:

El análisis exploratorio de datos (EDA) utilizando la visualización t-SNE, el clustering K-Means interactivo, las métricas de calidad de agrupamiento y la matriz de confusión, nos ha proporcionado una comprensión profunda de la estructura del conjunto de datos CLINC150, especialmente en lo que respecta a la granularidad y ambigüedad de las intenciones de usuario. Hemos confirmado que el espacio de embeddings de las consultas de usuario no es trivialmente separable. Las intenciones no forman grupos discretos y bien definidos, lo que sugiere desafíos inherentes para cualquier sistema que intente categorizarlas de forma automática. En particular, la capacidad de ajuste del número de clusters (K) en K-Means y la observación de cómo esto impacta en la matriz de confusión y las métricas, fue crucial para desvelar estas complejidades.

Hemos aprendido que:

- Las intenciones en CLINC150 exhiben un solapamiento semántico considerable.
- Existe una ambigüedad inherente en algunas consultas, donde la misma frase podría razonablemente pertenecer a varias intenciones.
- La distribución de los embeddings no presenta límites claros, lo que dificulta la tarea de clustering para agrupar las intenciones de forma "correcta".

#### Conclusiones por Hipótesis

A continuación, detallo las conclusiones intermedias y finales para cada una de las hipótesis planteadas:

**Hipótesis 1: ¿La alta granularidad de las intenciones de usuario en el conjunto de datos conduce a un solapamiento semántico significativo entre categorías, manifestándose en patrones de agrupamiento complejos?**

Conclusión Intermedia:

- Visualización de Intenciones Reales: Al visualizar el scatter plot coloreado por "Intención Real", se observa claramente que las áreas de diferentes colores (intenciones) se mezclan y superponen en gran medida. No hay "bolas" de un solo color bien separadas, sino una heterogeneidad cromática en muchas regiones. Esto es una fuerte indicación visual de solapamiento semántico a nivel de los embeddings.
- Matriz de Confusión (Filas Dispersas): La matriz de confusión, especialmente cuando se normaliza por fila, muestra que muchas intenciones reales se distribuyen en múltiples clusters de K-Means. Las filas de la matriz no son diagonales perfectas (o casi perfectas), sino que tienen valores significativos en varias columnas. Esto significa que K-Means, al intentar agrupar, a menudo divide una sola intención real en

varios subgrupos, lo que directamente apunta a una alta granularidad dentro de esa intención o a que la intención abarca un espectro semántico amplio.

**Conclusión Final:** Sí, la alta granularidad de las intenciones de usuario en el conjunto de datos CLINC150 efectivamente conduce a un solapamiento semántico significativo entre categorías, manifestándose en patrones de agrupamiento complejos. Los embeddings de intenciones semánticamente cercanas se encuentran próximos en el espacio reducido por t-SNE, causando que los algoritmos de clustering como K-Means mezclen estas categorías. La matriz de confusión confirmó que intenciones nominalmente distintas se agrupan en el mismo cluster, y que una única intención real a menudo se fragmenta en múltiples clusters, evidenciando estos "patrones de agrupamiento complejos" más allá de una simple asignación uno a uno.

**Hipótesis 2: ¿Existen instancias de ambigüedad semántica en las consultas de usuario que provocan que frases idénticas o muy similares se asocien a intenciones fundamentalmente distintas?**

Conclusión Intermedia:

- **Panel de Detalles de Selección:** Al interactuar con el scatter plot y seleccionar grupos de puntos (particularmente aquellos donde K-Means ha agrupado intenciones distintas o donde varias intenciones reales se superponen), el panel de detalles revela la "Distribución Completa de Intenciones Reales". Frecuentemente, encontramos que un solo cluster (coloreando por cluster\_id) contiene una mezcla notable de intenciones reales. Más allá de esto, al revisar los "Consultas de Texto (ejemplos)" de estos puntos mezclados, identificamos casos donde frases que a primera vista parecen muy similares (o incluso idénticas) se encuentran bajo diferentes intenciones reales. Por ejemplo, una consulta sobre "información de cuenta" podría ser check\_balance o account\_details, dependiendo del contexto sutil o de la etiqueta original.
- **Matriz de Confusión (Columnas Mezcladas):** Las columnas de la matriz de confusión, que representan los clusters de K-Means, a menudo muestran valores significativos en múltiples filas (intenciones reales diferentes). Esto indica que un cluster específico está capturando consultas de varias intenciones reales distintas, lo que sugiere que estas intenciones comparten una base semántica lo suficientemente cercana como para ser agrupadas por K-Means, potencialmente debido a ambigüedad en las consultas.

**Conclusión Final:** Sí, el análisis exploratorio sugiere fuertemente la existencia de instancias de ambigüedad semántica en las consultas de usuario. El panel de detalles de selección, junto con la matriz de confusión, nos permitió observar que el clustering K-Means a menudo agrupa frases que, aunque semánticamente distintas en su intención final, son muy similares en su formulación o comparten un contexto común. Al examinar los ejemplos de texto, se encontraron casos donde consultas con ligeras variaciones (o incluso sin ellas) fueron asignadas a intenciones diferentes en el dataset original, lo que indica un desafío significativo para la categorización y refuerza la presencia de ambigüedad.

Hipótesis 3: ¿La ausencia de límites claros y bien definidos entre intenciones semánticamente relacionadas en el espacio de embeddings dificulta su separación efectiva para el agrupamiento, más allá del simple solapamiento?

Conclusión Intermedia:

- Visualización General del Scatter Plot: Independientemente de si coloreamos por "Intención Real" o "ID de Cluster", el patrón general del gráfico de dispersión no muestra "vacíos" o "espacios en blanco" distintivos entre los supuestos grupos de intenciones. En cambio, hay una continuidad semántica donde los puntos de una intención se transicionan gradualmente hacia puntos de otra, sin una línea divisoria nítida. Esto es una característica inherente del espacio de embeddings para datos de lenguaje natural, donde las palabras y frases relacionadas semánticamente tienden a estar cerca.
- Métricas de Calidad de Clustering: El Silhouette Score consistentemente no alcanza valores cercanos a 1, manteniéndose en un rango moderado (ej., 0.3-0.5 para K=30). Un valor de Silhouette Score más bajo indica que los puntos no están tan fuertemente asignados a un único cluster y que hay cierta indecisión sobre a qué cluster pertenecen, lo cual es una consecuencia directa de la falta de límites claros. El Davies-Bouldin Index, aunque en un rango aceptable, tampoco se acerca a 0, lo que refuerza que los clusters no están tan compactos internamente ni tan bien separados externamente como se desearía para una separación "perfecta".
- Impacto de K en Métricas y Matriz: Incluso al experimentar con diferentes valores de K (por ejemplo, aumentando o disminuyendo el número de clusters), si bien algunas métricas pueden mejorar marginalmente, la matriz de confusión rara vez se convierte en una diagonal clara. Siempre persisten elementos fuera de la diagonal principal, lo que demuestra la dificultad intrínseca de K-Means para "cortar" el espacio de embeddings de manera que se alinee perfectamente con las intenciones reales.

Conclusión Final: Sí, la ausencia de límites claros y bien definidos entre intenciones semánticamente relacionadas en el espacio de embeddings dificulta significativamente su separación efectiva para el agrupamiento. Las visualizaciones de t-SNE muestran una continuidad y solapamiento generalizado en el espacio de intenciones, donde las fronteras semánticas son borrosas en lugar de discretas. Las métricas de clustering, en particular el Silhouette Score, corroboran que los clústeres no son tan cohesivos ni tan bien separados como se esperaría en un escenario con límites nítidos. Esto va más allá del simple solapamiento causado por alta granularidad; es una característica fundamental de cómo las intenciones se representan en el espacio de embeddings, haciendo que la tarea de clustering sea un desafío inherentemente complejo.

```
# Import library
import numpy as np
import matplotlib.pyplot as plt
import pylab as pl
import json
import cv2
import pandas as pd
from sklearn import preprocessing
import tensorflow_hub as hub
import tensorflow as tf
from tensorflow import keras
from keras.layers import Dropout
import os
import chardet
from pathlib import Path

import kagglehub

# Download latest version
path = kagglehub.dataset_download("hongtrung/clinc150-dataset")

print("Path to dataset files:", path)

→ Path to dataset files: /kaggle/input/clinc150-dataset

data_path = os.path.join(path, "data", "data_full.json")
data_file = Path(data_path)

# Detectar codificación del archivo
encoding_result = chardet.detect(data_file.read_bytes())
detected_encoding = encoding_result['encoding']
confidence = encoding_result['confidence']

print(f"Archivo: {data_file.name}")
print(f"Codificación detectada: {detected_encoding} (confianza: {confidence})")

# Abrir y cargar el JSON con la codificación detectada
with open(data_path, "r", encoding=detected_encoding) as f:
    CLINC150 = json.load(f)

print("¡Archivo cargado correctamente!")

→ Archivo: data_full.json
Codificación detectada: ascii (confianza: 1.0)
¡Archivo cargado correctamente!

CLINC150.keys()

→ dict_keys(['oos_val', 'val', 'train', 'oos_test', 'test', 'oos_train'])

import sys

# Función para estimar el tamaño total de un objeto Python, incluyendo sus contenidos anidados
def get_total_size(obj, seen=None):
    """
    Estima recursivamente el tamaño en bytes de un objeto y sus contenidos.
    """

    if seen is None:
        seen = set()
    obj_id = id(obj)
    if obj_id in seen:
        return 0
    seen.add(obj_id)
    size = sys.getsizeof(obj)
    if isinstance(obj, dict):
        size += sum(get_total_size(v, seen) for v in obj.values())
        size += sum(get_total_size(k, seen) for k in obj.keys())
    elif isinstance(obj, (list, tuple, set, frozenset)):
        size += sum(get_total_size(i, seen) for i in obj)
    # Siquieres considerar otros tipos de objetos complejos, como instancias de clases, etc.
    # elif hasattr(obj, '__dict__'):
    #     size += get_total_size(obj.__dict__, seen)
    return size

# Obtener el tamaño estimado de CLINC150 en bytes
clinc_memory_bytes = get_total_size(CLINC150)

# Convertir a un formato legible (KB, MB, GB)
```

```
def convert_bytes_to_human_readable(size_bytes):
    """Convierte un tamaño en bytes a un formato legible (KB, MB, GB, etc.)."""
    for unit in ['bytes', 'KB', 'MB', 'GB', 'TB']:
        if size_bytes < 1024.0:
            return f"{size_bytes:.1f} {unit}"
        size_bytes /= 1024.0
    return f"{size_bytes:.1f} PB"

print(f"La variable 'CLINC150' ocupa aproximadamente: {convert_bytes_to_human_readable(clinc_memory_bytes)}")
```

↳ La variable 'CLINC150' ocupa aproximadamente: 5.6 MB

```
# Accede al subconjunto 'train'
train_data = CLINC150['train']
```

```
print(f"Tipo de datos de CLINC150['train']: {type(train_data)}")
print(f"Número de consultas en el subconjunto 'train': {len(train_data)})")
```

```
# Imprime las primeras 3 consultas para ver su estructura
print("\nPrimeras 3 consultas del subconjunto 'train':")
for i in range(min(3, len(train_data))):
    print(train_data[i])
```

↳ Tipo de datos de CLINC150['train']: <class 'list'>
Número de consultas en el subconjunto 'train': 15000

```
Primeras 3 consultas del subconjunto 'train':
['what expression would i use to say i love you if i were an italian', 'translate']
['can you tell me how to say 'i do not speak much spanish', in spanish", 'translate']
['what is the equivalent of, 'life is good' in french", 'translate']
```

```
# Accede al subconjunto 'val'
val_data = CLINC150['val']
```

```
print(f"Tipo de datos de CLINC150['val']: {type(val_data)}")
print(f"Número de consultas en el subconjunto 'val': {len(val_data)})")
```

```
# Imprime las primeras 3 consultas para ver su estructura
print("\nPrimeras 3 consultas del subconjunto 'val':")
for i in range(min(3, len(val_data))):
    print(val_data[i])
```

↳ Tipo de datos de CLINC150['val']: <class 'list'>
Número de consultas en el subconjunto 'val': 3000

```
Primeras 3 consultas del subconjunto 'val':
['in spanish, meet me tomorrow is said how', 'translate']
['in french, how do i say, see you later', 'translate']
['how do you say hello in japanese', 'translate']
```

```
# Accede al subconjunto 'test'
test_data = CLINC150['test']
```

```
print(f"Tipo de datos de CLINC150['test']: {type(test_data)}")
print(f"Número de consultas en el subconjunto 'test': {len(test_data)})")
```

```
# Imprime las primeras 3 consultas para ver su estructura
print("\nPrimeras 3 consultas del subconjunto 'test':")
for i in range(min(3, len(test_data))):
    print(test_data[i])
```

↳ Tipo de datos de CLINC150['test']: <class 'list'>
Número de consultas en el subconjunto 'test': 4500

```
Primeras 3 consultas del subconjunto 'test':
['how would you say fly in italian', 'translate']
['what's the spanish word for pasta", 'translate']
['how would they say butter in zambia', 'translate']
```

```
train_data = pd.DataFrame(CLINC150['train'], columns=['text', 'intent'])
test_data = pd.DataFrame(CLINC150['test'], columns=['text', 'intent'])
val_data = pd.DataFrame(CLINC150['val'], columns=['text', 'intent'])
```

```
train_oss_data = pd.DataFrame(CLINC150['oos_train'], columns=['text', 'intent'])
test_oss_data = pd.DataFrame(CLINC150['oos_test'], columns=['text', 'intent'])
val_oss_data = pd.DataFrame(CLINC150['oos_val'], columns=['text', 'intent'])
```

```
train_oss_data.info()
```

↳ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99

```
Data columns (total 2 columns):
 #   Column Non-Null Count Dtype  
--- 
 0   text    100 non-null   object 
 1   intent  100 non-null   object 
dtypes: object(2)
memory usage: 1.7+ KB
```

```
test_oss_data.info()
```

```
↳ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 2 columns):
 #   Column Non-Null Count Dtype  
--- 
 0   text    1000 non-null   object 
 1   intent  1000 non-null   object 
dtypes: object(2)
memory usage: 15.8+ KB
```

```
val_oss_data.info()
```

```
↳ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 2 columns):
 #   Column Non-Null Count Dtype  
--- 
 0   text    100 non-null   object 
 1   intent  100 non-null   object 
dtypes: object(2)
memory usage: 1.7+ KB
```

```
train_data.info()
```

```
↳ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 2 columns):
 #   Column Non-Null Count Dtype  
--- 
 0   text    15000 non-null   object 
 1   intent  15000 non-null   object 
dtypes: object(2)
memory usage: 234.5+ KB
```

```
test_data.info()
```

```
↳ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 4500 entries, 0 to 4499
Data columns (total 2 columns):
 #   Column Non-Null Count Dtype  
--- 
 0   text    4500 non-null   object 
 1   intent  4500 non-null   object 
dtypes: object(2)
memory usage: 70.4+ KB
```

```
val_data.info()
```

```
↳ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 2 columns):
 #   Column Non-Null Count Dtype  
--- 
 0   text    3000 non-null   object 
 1   intent  3000 non-null   object 
dtypes: object(2)
memory usage: 47.0+ KB
```

```
print(np.unique(train_data["intent"]))
total_label = int(len(np.unique(train_data["intent"])))
print("\n====> Total Intent: ", total_label)
```

```
↳ ['accept_reservations' 'account_blocked' 'alarm' 'application_status'
 'apr' 'are_you_a_bot' 'balance' 'bill_balance' 'bill_due' 'book_flight'
 'book_hotel' 'calculator' 'calendar' 'calendar_update' 'calories'
 'cancel' 'cancel_reservation' 'car_rental' 'card_declined' 'carry_on'
 'change_accent' 'change_ai_name' 'change_language' 'change_speed'
 'change_user_name' 'change_volume' 'confirm_reservation' 'cook_time'
 'credit_limit' 'credit_limit_change' 'credit_score' 'current_location'
 'damaged_card' 'date' 'definition' 'direct_deposit' 'directions'
 'distance' 'do_you_have_pets' 'exchange_rate' 'expiration_date'
 'find_phone' 'flight_status' 'flip_coin' 'food_last' 'freeze_account'
 'fun_fact' 'gas' 'gas_type' 'goodbye' 'greeting' 'how_busy'
 'how_old_are_you' 'improve_credit_score' 'income'
```

```
'ingredient_substitution' 'ingredients_list' 'insurance'
'insurance_change' 'interest_rate' 'international_fees'
'international_visa' 'jump_start' 'last_maintenance' 'lost_luggage'
'make_call' 'maybe' 'meal_suggestion' 'meaning_of_life'
'measurement_conversion' 'meeting_schedule' 'min_payment' 'mpg'
'new_card' 'next_holiday' 'next_song' 'no' 'nutrition_info'
'oil_change_how' 'oil_change_when' 'order' 'order_checks' 'order_status'
'pay_bill' 'payday' 'pin_change' 'play_music' 'plug_type' 'pto_balance'
'pto_request' 'pto_request_status' 'pto_used' 'recipe' 'redeem_rewards'
'reminder' 'reminder_update' 'repeat' 'replacement_card_duration'
'report_fraud' 'report_lost_card' 'reset_settings'
'restaurant_reservation' 'restauran_reviews' 'restaurant_suggestion'
'rewards_balance' 'roll_dice' 'rollover_401k' 'routing'
'schedule_maintenance' 'schedule_meeting' 'share_location'
'shopping_list' 'shopping_list_update' 'smart_home' 'spelling'
'spending_history' 'sync_device' 'taxes' 'tell_joke' 'text' 'thank_you'
'time' 'timer' 'timezone' 'tire_change' 'tire_pressure' 'todo_list'
'todo_list_update' 'traffic' 'transactions' 'transfer' 'translate'
'travel_alert' 'travel_notification' 'travel_suggestion' 'uber'
'update_playlist' 'user_name' 'vaccines' 'w2' 'weather'
'what_are_your_hobbies' 'what_can_i_ask_you' 'what_is_your_name'
'what_song' 'where_are_you_from' 'whisper_mode' 'who_do_you_work_for'
'who_made_you' 'yes']
```

====> Total Intent: 150

```
print(np.unique(test_data["intent"]))
total_label = int(len(np.unique(test_data["intent"])))
print("\n====> Total Intent: ", total_label)
```

⤵ [ 'accept\_reservations' 'account\_blocked' 'alarm' 'application\_status'
'apr' 'are\_you\_a\_bot' 'balance' 'bill\_balance' 'bill\_due' 'book\_flight'
'book\_hotel' 'calculator' 'calendar' 'calendar\_update' 'calories'
'cancel' 'cancel\_reservation' 'car\_rental' 'card\_declined' 'carry\_on'
'change\_accent' 'change\_ai\_name' 'change\_language' 'change\_speed'
'change\_user\_name' 'change\_volume' 'confirm\_reservation' 'cook\_time'
'credit\_limit' 'credit\_limit\_change' 'credit\_score' 'current\_location'
'damaged\_card' 'date' 'definition' 'direct\_deposit' 'directions'
'distance' 'do\_you\_have\_pets' 'exchange\_rate' 'expiration\_date'
'find\_phone' 'flight\_status' 'flip\_coin' 'food\_last' 'freeze\_account'
'fun\_fact' 'gas' 'gas\_type' 'goodbye' 'greeting' 'how\_busy'
'how\_old\_are\_you' 'improve\_credit\_score' 'income'
'ingredient\_substitution' 'ingredients\_list' 'insurance'
'insurance\_change' 'interest\_rate' 'international\_fees'
'international\_visa' 'jump\_start' 'last\_maintenance' 'lost\_luggage'
'make\_call' 'maybe' 'meal\_suggestion' 'meaning\_of\_life'
'measurement\_conversion' 'meeting\_schedule' 'min\_payment' 'mpg'
'new\_card' 'next\_holiday' 'next\_song' 'no' 'nutrition\_info'
'oil\_change\_how' 'oil\_change\_when' 'order' 'order\_checks' 'order\_status'
'pay\_bill' 'payday' 'pin\_change' 'play\_music' 'plug\_type' 'pto\_balance'
'pto\_request' 'pto\_request\_status' 'pto\_used' 'recipe' 'redeem\_rewards'
'reminder' 'reminder\_update' 'repeat' 'replacement\_card\_duration'
'report\_fraud' 'report\_lost\_card' 'reset\_settings'
'restaurant\_reservation' 'restauran\_reviews' 'restaurant\_suggestion'
'rewards\_balance' 'roll\_dice' 'rollover\_401k' 'routing'
'schedule\_maintenance' 'schedule\_meeting' 'share\_location'
'shopping\_list' 'shopping\_list\_update' 'smart\_home' 'spelling'
'spending\_history' 'sync\_device' 'taxes' 'tell\_joke' 'text' 'thank\_you'
'time' 'timer' 'timezone' 'tire\_change' 'tire\_pressure' 'todo\_list'
'todo\_list\_update' 'traffic' 'transactions' 'transfer' 'translate'
'travel\_alert' 'travel\_notification' 'travel\_suggestion' 'uber'
'update\_playlist' 'user\_name' 'vaccines' 'w2' 'weather'
'what\_are\_your\_hobbies' 'what\_can\_i\_ask\_you' 'what\_is\_your\_name'
'what\_song' 'where\_are\_you\_from' 'whisper\_mode' 'who\_do\_you\_work\_for'
'who\_made\_you' 'yes']

====> Total Intent: 150

```
print(np.unique(val_data["intent"]))
total_label = int(len(np.unique(val_data["intent"])))
print("\n====> Total Intent: ", total_label)
```

⤵ [ 'accept\_reservations' 'account\_blocked' 'alarm' 'application\_status'
'apr' 'are\_you\_a\_bot' 'balance' 'bill\_balance' 'bill\_due' 'book\_flight'
'book\_hotel' 'calculator' 'calendar' 'calendar\_update' 'calories'
'cancel' 'cancel\_reservation' 'car\_rental' 'card\_declined' 'carry\_on'
'change\_accent' 'change\_ai\_name' 'change\_language' 'change\_speed'
'change\_user\_name' 'change\_volume' 'confirm\_reservation' 'cook\_time'
'credit\_limit' 'credit\_limit\_change' 'credit\_score' 'current\_location'
'damaged\_card' 'date' 'definition' 'direct\_deposit' 'directions'
'distance' 'do\_you\_have\_pets' 'exchange\_rate' 'expiration\_date'
'find\_phone' 'flight\_status' 'flip\_coin' 'food\_last' 'freeze\_account'
'fun\_fact' 'gas' 'gas\_type' 'goodbye' 'greeting' 'how\_busy'
'how\_old\_are\_you' 'improve\_credit\_score' 'income'
'ingredient\_substitution' 'ingredients\_list' 'insurance'
'insurance\_change' 'interest\_rate' 'international\_fees'
'international\_visa' 'jump\_start' 'last\_maintenance' 'lost\_luggage'
'make\_call' 'maybe' 'meal\_suggestion' 'meaning\_of\_life'

```
'measurement_conversion' 'meeting_schedule' 'min_payment' 'mpg'
'new_card' 'next_holiday' 'next_song' 'no' 'nutrition_info'
'oil_change_how' 'oil_change_when' 'orden' 'order_checks' 'order_status'
'pay_bill' 'payday' 'pin_change' 'play_music' 'plug_type' 'pto_balance'
'pto_request' 'pto_request_status' 'pto_used' 'recipe' 'redeem_rewards'
'reminder' 'reminder_update' 'repeat' 'replacement_card_duration'
'report_fraud' 'report_lost_card' 'reset_settings'
'restaurant_reservation' 'restaurante_reviews' 'restaurante_suggestion'
'rewards_balance' 'roll_dice' 'rollover_401k' 'routing'
'schedule_maintenance' 'schedule_meeting' 'share_location'
'shopping_list' 'shopping_list_update' 'smart_home' 'spelling'
'spending_history' 'sync_device' 'taxes' 'tell_joke' 'text' 'thank_you'
'time' 'timer' 'timezone' 'tire_change' 'tire_pressure' 'todo_list'
'todo_list_update' 'traffic' 'transactions' 'transfer' 'translate'
'travel_alert' 'travel_notification' 'travel_suggestion' 'uber'
'update_playlist' 'user_name' 'vaccines' 'w2' 'weather'
'what_are_your_hobbies' 'what_can_i_ask_you' 'what_is_your_name'
'what_song' 'where_are_you_from' 'whisper_mode' 'who_do_you_work_for'
'who_made_you' 'yes']
```

=====> Total Intent: 150

```
import seaborn as sns

# Lista de subconjuntos a analizar
subsets_to_analyze = ['train', 'val', 'test']

for subset_name in subsets_to_analyze:
    print(f"\n--- Analizando el subconjunto: '{subset_name}' ---")

    # Verificar si la clave existe y el subconjunto no está vacío
    if subset_name not in CLINC150 or not CLINC150[subset_name]:
        print(f"El subconjunto '{subset_name}' no existe o está vacío. Saltando análisis.")
        continue

    # Convertir el subconjunto a un DataFrame
    df_current = pd.DataFrame(CLINC150[subset_name], columns=['text', 'intent'])

    print(f"\nPrimeras 5 filas del DataFrame de {subset_name}:")
    print(df_current.head())

    print(f"\nInformación general del DataFrame de {subset_name}:")
    df_current.info()

    print(f"\nVerificando valores nulos en el DataFrame de {subset_name}:")
    print(df_current.isnull().sum())

    # --- Análisis de la Longitud de las Consultas ---
    df_current['text_length'] = df_current['text'].apply(len)

    print(f"\nEstadísticas descriptivas de la longitud de las consultas ({subset_name}):")
    print(df_current['text_length'].describe())

    plt.figure(figsize=(10, 6))
    sns.histplot(df_current['text_length'], bins=50, kde=True)
    plt.title(f'Distribución de la Longitud de las Consultas (Subconjunto {subset_name.capitalize()})')
    plt.xlabel('Longitud de la Consulta')
    plt.ylabel('Frecuencia')
    plt.grid(axis='y', alpha=0.75)
    plt.show()

    # --- Análisis de la Distribución de las Categorías de Intención ---
    num_unique_intents = df_current['intent'].nunique()
    print(f"\nNúmero de categorías de intención únicas en {subset_name}: {num_unique_intents}")

    intent_counts = df_current['intent'].value_counts()

    print(f"\nTop 10 intenciones más frecuentes en {subset_name}:")
    print(intent_counts.head(10))

    print(f"\nTop 10 intenciones menos frecuentes en {subset_name}:")
    print(intent_counts.tail(10))

    # Gráfico de barras de las 20 intenciones más frecuentes
    plt.figure(figsize=(12, 7))
    sns.barplot(x=intent_counts.head(20).index, y=intent_counts.head(20).values, palette='viridis')
    plt.title(f'Top 20 Intenciones Más Frecuentes (Subconjunto {subset_name.capitalize()})')
    plt.xlabel('Intención')
    plt.ylabel('Número de Consultas')
    plt.xticks(rotation=45, ha='right')
    plt.tight_layout()
    plt.show()
```

```
# Gráfico de barras de las 20 intenciones menos frecuentes
plt.figure(figsize=(12, 7))
sns.barplot(x=intent_counts.tail(20).index, y=intent_counts.tail(20).values, palette='plasma')
plt.title(f'Top 20 Intenciones Menos Frecuentes (Subconjunto {subset_name.capitalize()})')
plt.xlabel('Intención')
plt.ylabel('Número de Consultas')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# --- Verificación final de los subconjuntos OOS ---
print("\n--- Verificación final de los subconjuntos 'out-of-scope' (OOS) ---")
oos_subsets = ['oos_train', 'oos_val', 'oos_test']
for key in oos_subsets:
    if key in CLINC150:
        print(f"Tamaño de CLINC150['{key}']: {len(CLINC150[key])}")
    else:
        print(f"La clave '{key}' no se encontró en el dataset.")
```



--- Analizando el subconjunto: 'train' ---

Primeras 5 filas del DataFrame de train:

	text	intent
0	what expression would i use to say i love you ...	translate
1	can you tell me how to say 'i do not speak muc...	translate
2	what is the equivalent of, 'life is good' in f...	translate
3	tell me how to say, 'it is a beautiful morning...	translate
4	if i were mongolian, how would i say that i am...	translate

Información general del DataFrame de train:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15000 entries, 0 to 14999
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
 ---  --     --     --     --    
 0   text    15000 non-null   object  
 1   intent   15000 non-null   object  
 dtypes: object(2)
memory usage: 234.5+ KB
```

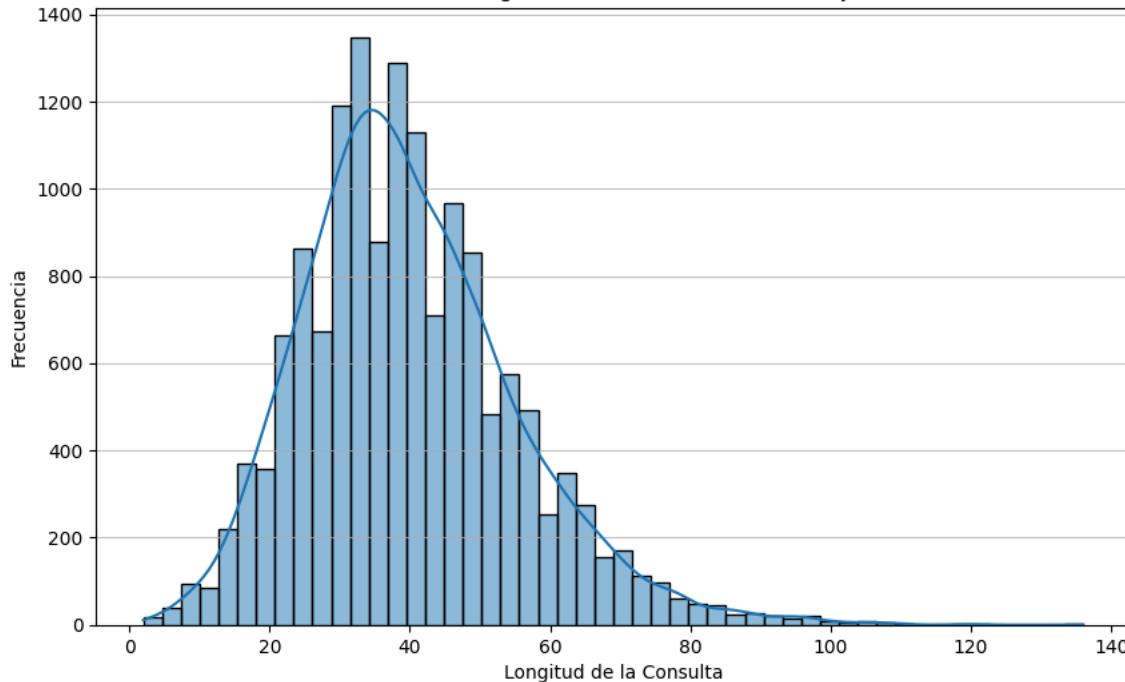
Verificando valores nulos en el DataFrame de train:

```
text      0
intent   0
dtype: int64
```

Estadísticas descriptivas de la longitud de las consultas (train):

```
count    15000.000000
mean     39.906067
std      15.262904
min      2.000000
25%     29.000000
50%     38.000000
75%     49.000000
max     136.000000
Name: text_length, dtype: float64
```

Distribución de la Longitud de las Consultas (Subconjunto Train)



Número de categorías de intención únicas en train: 150

Top 10 intenciones más frecuentes en train:

```
intent
translate      100
transfer       100
timer          100
definition     100
meaning_of_life 100
insurance_change 100
find_phone      100
travel_alert    100
pto_request     100
improve_credit_score 100
Name: count, dtype: int64
```

Top 10 intenciones menos frecuentes en train:

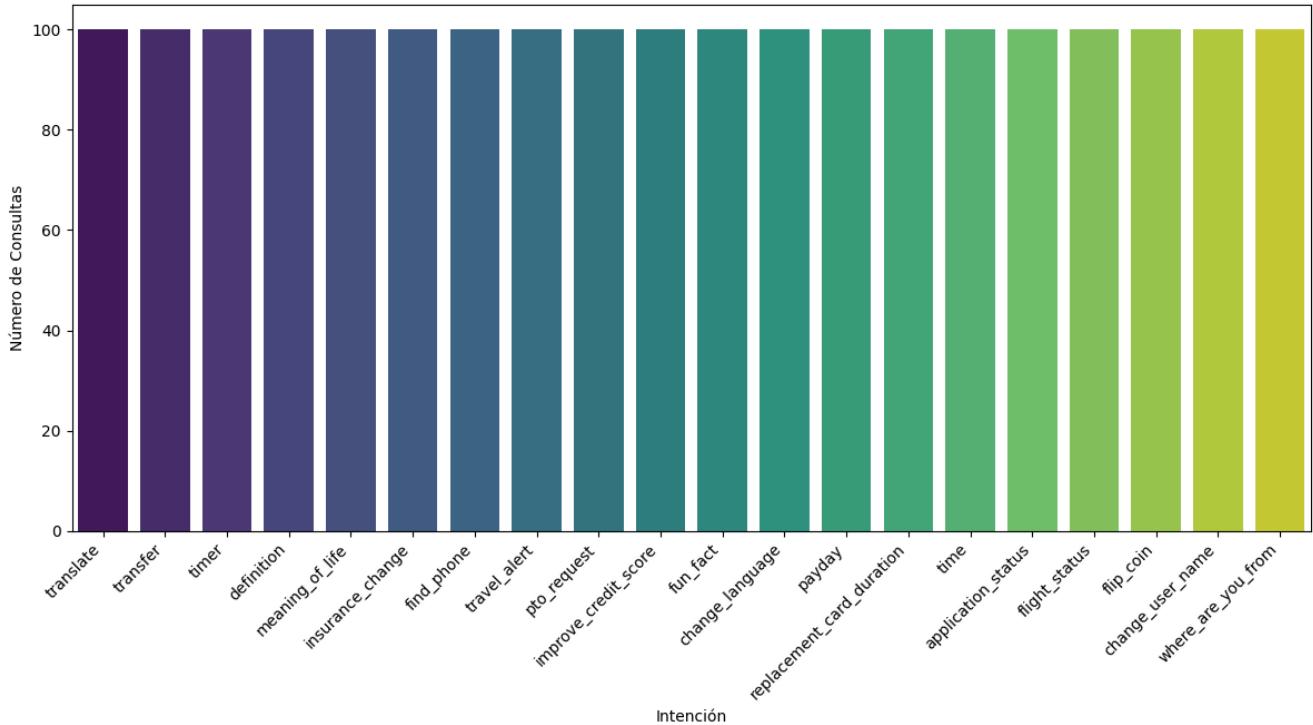
```
intent
how_old_are_you 100
car_rental       100
...             100
```

```
jupyter_start
meal_suggestion    100
recipe             100
income             100
order              100
traffic            100
order_checks       100
card_declined      100
Name: count, dtype: int64
<ipython-input-20-e84cfde419e1>:54: FutureWarning:
```

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

```
sns.barplot(x=intent_counts.head(20).index, y=intent_counts.head(20).values, palette='viridis')
```

Top 20 Intenciones Más Frecuentes (Subconjunto Train)

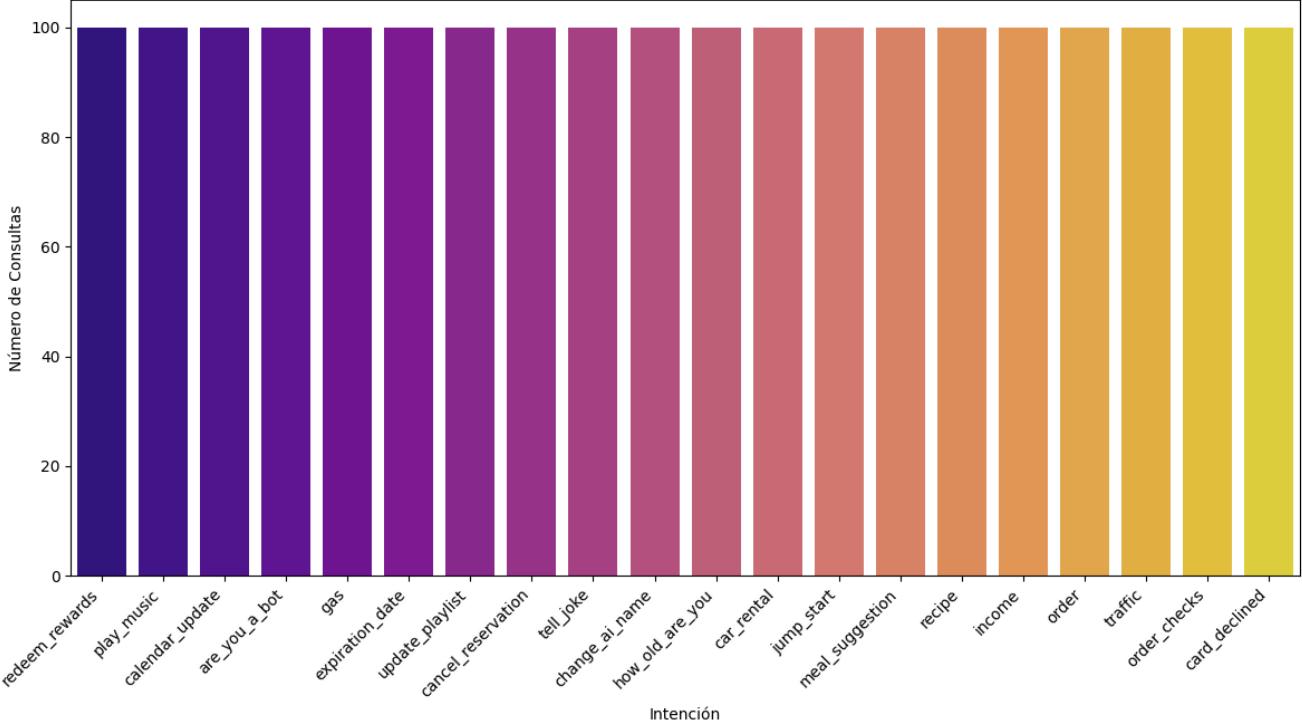


```
<ipython-input-20-e84cfde419e1>:64: FutureWarning:
```

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

```
sns.barplot(x=intent_counts.tail(20).index, y=intent_counts.tail(20).values, palette='plasma')
```

Top 20 Intenciones Menos Frecuentes (Subconjunto Train)



--- Analizando el subconjunto: 'val' ---

Primeras 5 filas del DataFrame de val:

```
text      intent
0   in spanish meet me tomorrow is said how translate
```

```

1     in spanish, how do i say tomorrow in spanish translate
1     in french, how do i say, see you later translate
2           how do you say hello in japanese translate
3 how do i ask about the weather in chinese translate
4 how can i say "cancel my order" in french translate

```

Información general del DataFrame de val:

```

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 2 columns):
 #   Column Non-Null Count Dtype
 ---  --  --  --  --
 0   text    3000 non-null   object
 1   intent  3000 non-null   object
dtypes: object(2)
memory usage: 47.0+ KB

```

Verificando valores nulos en el DataFrame de val:

```

text      0
intent   0
dtype: int64

```

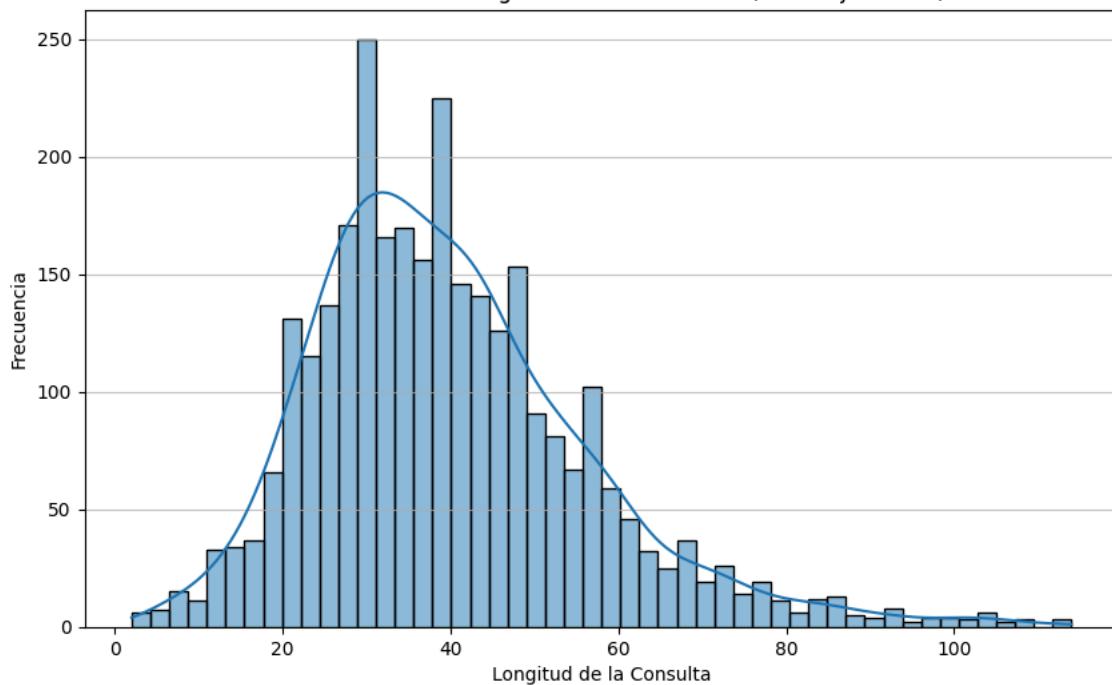
Estadísticas descriptivas de la longitud de las consultas (val):

```

count    3000.00000
mean     39.825667
std      16.580860
min      2.000000
25%     28.000000
50%     37.000000
75%     49.000000
max     114.000000
Name: text_length, dtype: float64

```

Distribución de la Longitud de las Consultas (Subconjunto Val)



Número de categorías de intención únicas en val: 150

Top 10 intenciones más frecuentes en val:

```

intent          20
translate       20
transfer        20
timer           20
definition      20
meaning_of_life 20
insurance_change 20
find_phone      20
travel_alert    20
pto_request     20
improve_credit_score 20
Name: count, dtype: int64

```

Top 10 intenciones menos frecuentes en val:

```

intent
how_old_are_you 20
car_rental       20
jump_start       20
meal_suggestion  20
recipe           20
income           20
order            20
traffic          20
orden_checks     20

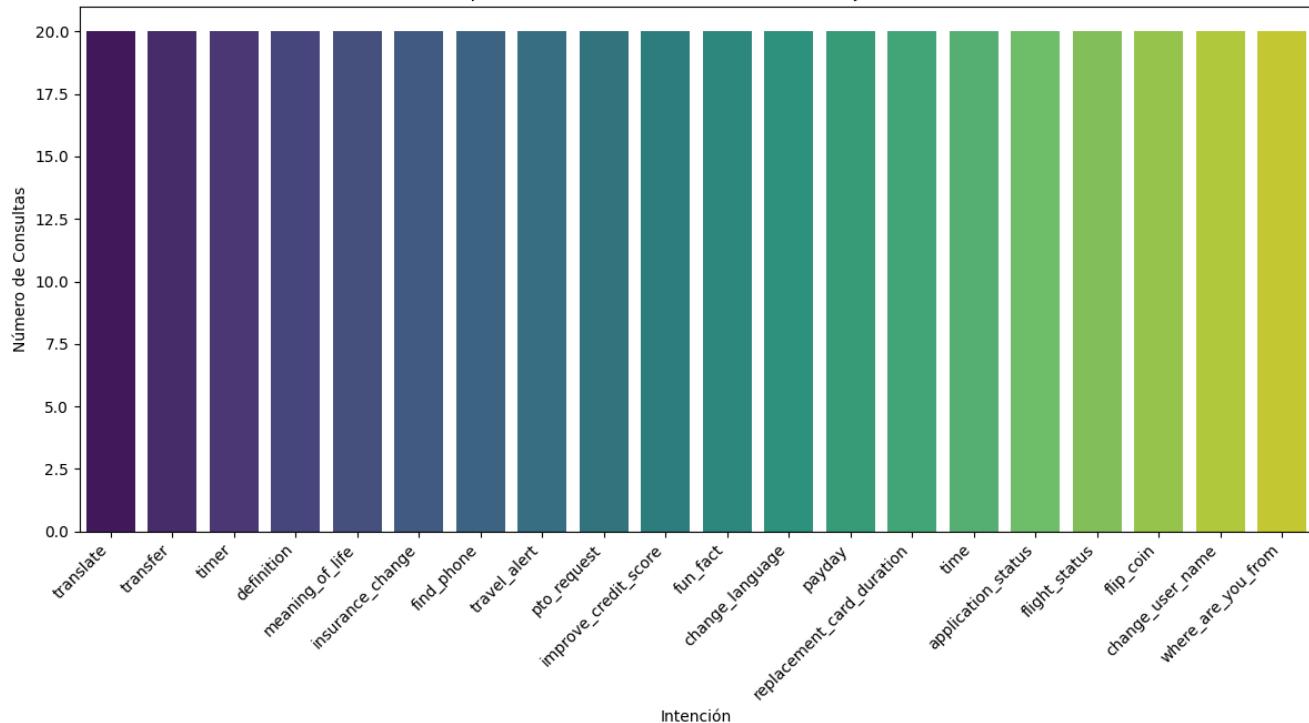
```

```
..._checks      20
card_declined   20
Name: count, dtype: int64
<ipython-input-20-e84cfde419e1>:54: FutureWarning:
```

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

```
sns.barplot(x=intent_counts.head(20).index, y=intent_counts.head(20).values, palette='viridis')
```

Top 20 Intenciones Más Frecuentes (Subconjunto Val)

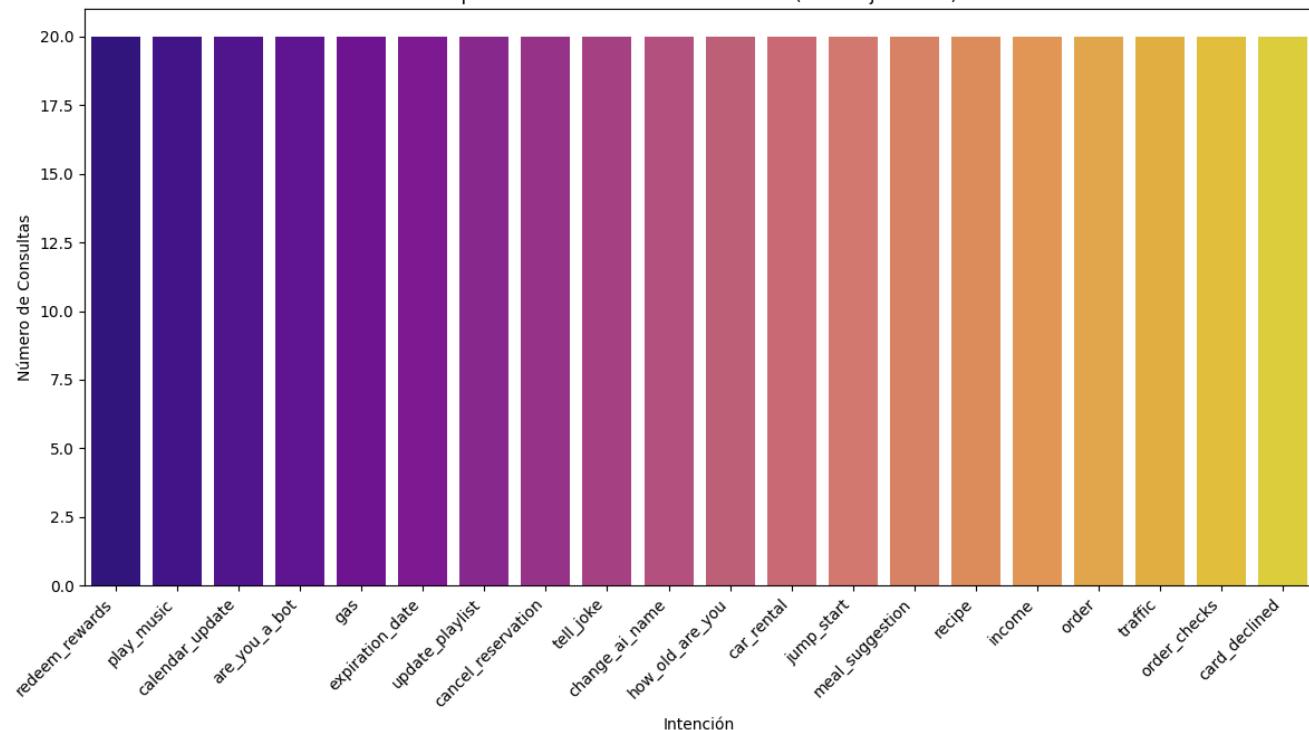


```
<ipython-input-20-e84cfde419e1>:64: FutureWarning:
```

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

```
sns.barplot(x=intent_counts.tail(20).index, y=intent_counts.tail(20).values, palette='plasma')
```

Top 20 Intenciones Menos Frecuentes (Subconjunto Val)



--- Analizando el subconjunto: 'test' ---

Primeras 5 filas del DataFrame de test:

	text	intent
0	how would you say fly in italian	translate
1	what's the spanish word for pasta	translate
2	how would they say butter in zambia	translate
3	how do you say fast in spanish	translate
4	what's the word for trees in norway	translate

Información general del DataFrame de test:

[https://colab.research.google.com/drive/1blOf4LTJlbAQhP\\_PcoHkqlKpwpq95pRw?hl=es#scrollTo=ZE-SuZVdIPQH&printMode=true](https://colab.research.google.com/drive/1blOf4LTJlbAQhP_PcoHkqlKpwpq95pRw?hl=es#scrollTo=ZE-SuZVdIPQH&printMode=true)

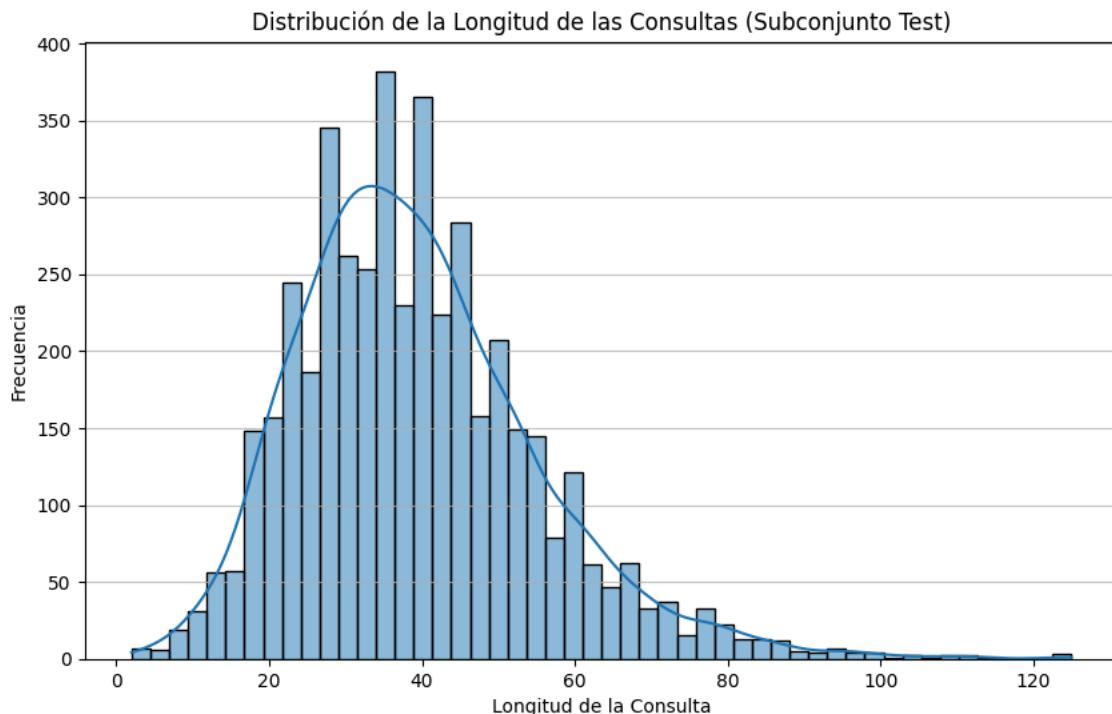
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4500 entries, 0 to 4499
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  -- 
 0   text    4500 non-null   object 
 1   intent   4500 non-null   object 
dtypes: object(2)
memory usage: 70.4+ KB
```

Verificando valores nulos en el DataFrame de test:

```
text      0
intent   0
dtype: int64
```

Estadísticas descriptivas de la longitud de las consultas (test):

```
count    4500.000000
mean     39.308889
std      15.760182
min      2.000000
25%     28.000000
50%     37.000000
75%     48.000000
max     125.000000
Name: text_length, dtype: float64
```



Número de categorías de intención únicas en test: 150

Top 10 intenciones más frecuentes en test:

```
intent
translate      30
transfer       30
timer          30
definition     30
meaning_of_life 30
insurance_change 30
find_phone      30
travel_alert    30
pto_request     30
improve_credit_score 30
Name: count, dtype: int64
```

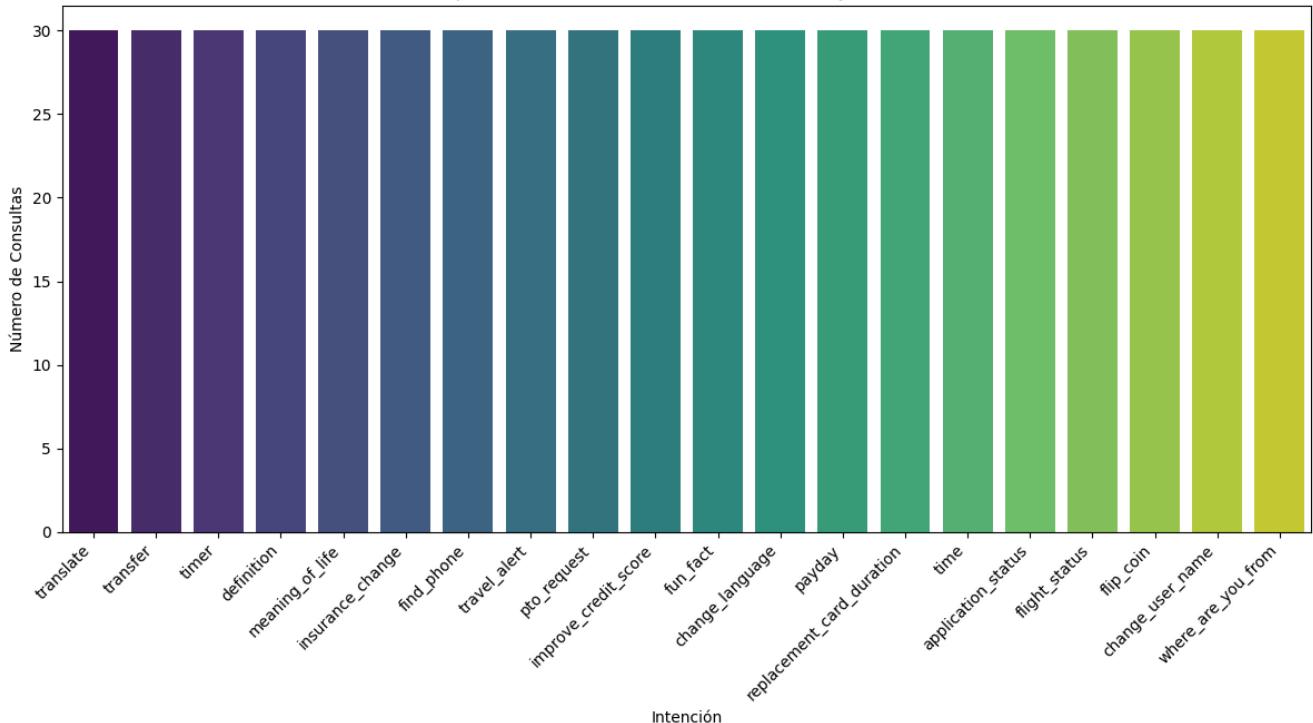
Top 10 intenciones menos frecuentes en test:

```
intent
how_old_are_you 30
car_rental       30
jump_start        30
meal_suggestion   30
recipe           30
income            30
order             30
traffic           30
order_checks      30
card_declined     30
Name: count, dtype: int64
<ipython-input-20-e84cfde419e1>:54: FutureWarning:
```

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

```
sns.barplot(x=intent_counts.head(20).index, y=intent_counts.head(20).values, palette='viridis')
```

Top 20 Intenciones Más Frecuentes (Subconjunto Test)

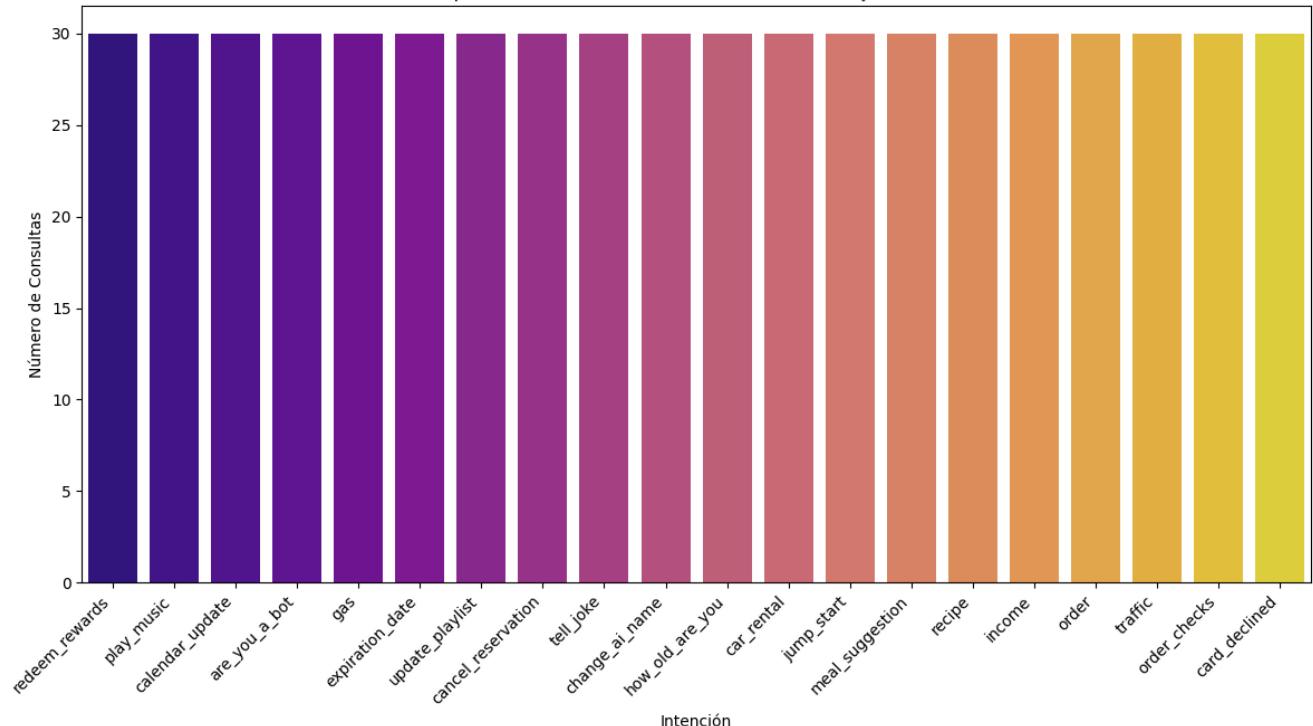


<ipython-input-20-e84cfde419e1>:64: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set

```
sns.barplot(x=intent_counts.tail(20).index, y=intent_counts.tail(20).values, palette='plasma')
```

Top 20 Intenciones Menos Frecuentes (Subconjunto Test)



--- Verificación final de los subconjuntos 'out-of-scope' (OOS) ---

Tamaño de CLINC150['oos\_train']: 100

Tamaño de CLINC150['oos\_val']: 100

Tamaño de CLINC150['oos\_test']: 1000



```

df_test = pd.DataFrame(CLINC150['test'], columns=['text', 'intent'])
df_test['text_length'] = df_test['text'].apply(len)
stats_test = df_test['text_length'].describe()

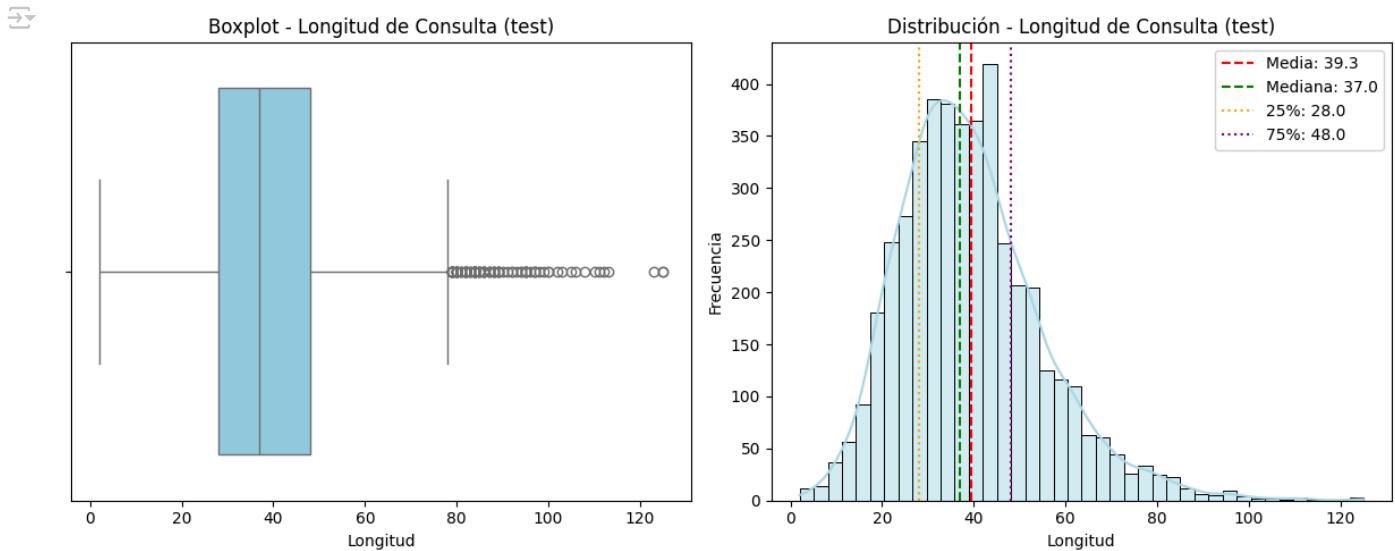
plt.figure(figsize=(12, 5))

# Boxplot
plt.subplot(1, 2, 1)
sns.boxplot(x=df_test['text_length'], color='skyblue')
plt.title('Boxplot - Longitud de Consulta (test)')
plt.xlabel('Longitud')

# Histograma con líneas estadísticas
plt.subplot(1, 2, 2)
sns.histplot(df_test['text_length'], bins=40, kde=True, color='lightblue')
plt.axvline(stats_test['mean'], color='red', linestyle='--', label=f"Media: {stats_test['mean']:.1f}")
plt.axvline(stats_test['50%'], color='green', linestyle='--', label=f"Mediana: {stats_test['50%']:.1f}")
plt.axvline(stats_test['25%'], color='orange', linestyle=':', label=f"25%: {stats_test['25%']:.1f}")
plt.axvline(stats_test['75%'], color='purple', linestyle=':', label=f"75%: {stats_test['75%']:.1f}")
plt.title('Distribución - Longitud de Consulta (test)')
plt.xlabel('Longitud')
plt.ylabel('Frecuencia')
plt.legend()

plt.tight_layout()
plt.show()

```



```

df_val = pd.DataFrame(CLINC150['val'], columns=['text', 'intent'])
df_val['text_length'] = df_val['text'].apply(len)
stats_val = df_val['text_length'].describe()

plt.figure(figsize=(12, 5))

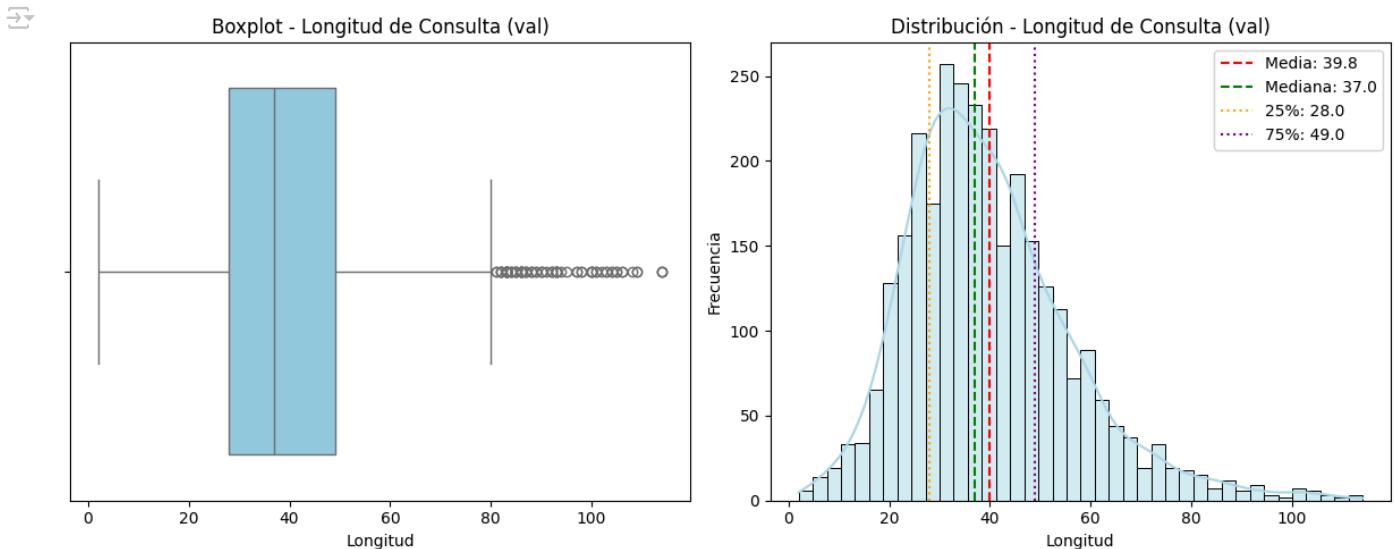
# Boxplot
plt.subplot(1, 2, 1)
sns.boxplot(x=df_val['text_length'], color='skyblue')
plt.title('Boxplot - Longitud de Consulta (val)')
plt.xlabel('Longitud')

# Histograma con líneas estadísticas
plt.subplot(1, 2, 2)
sns.histplot(df_val['text_length'], bins=40, kde=True, color='lightblue')
plt.axvline(stats_val['mean'], color='red', linestyle='--', label=f"Media: {stats_val['mean']:.1f}")
plt.axvline(stats_val['50%'], color='green', linestyle='--', label=f"Mediana: {stats_val['50%']:.1f}")
plt.axvline(stats_val['25%'], color='orange', linestyle=':', label=f"25%: {stats_val['25%']:.1f}")
plt.axvline(stats_val['75%'], color='purple', linestyle=':', label=f"75%: {stats_val['75%']:.1f}")
plt.title('Distribución - Longitud de Consulta (val)')
plt.xlabel('Longitud')
plt.ylabel('Frecuencia')
plt.legend()

plt.tight_layout()

```

```
plt.show()
```



```
df_train = pd.DataFrame(CLINC150['train'], columns=['text', 'intent'])
df_train['text_length'] = df_train['text'].apply(len)
stats_train = df_train['text_length'].describe()

plt.figure(figsize=(12, 5))

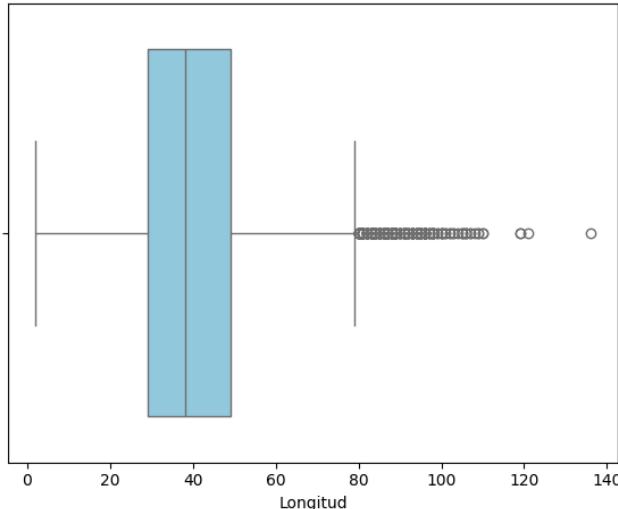
# Boxplot
plt.subplot(1, 2, 1)
sns.boxplot(x=df_train['text_length'], color='skyblue')
plt.title('Boxplot - Longitud de Consulta (train)')
plt.xlabel('Longitud')

# Histograma con líneas estadísticas
plt.subplot(1, 2, 2)
sns.histplot(df_train['text_length'], bins=40, kde=True, color='lightblue')
plt.axvline(stats_train['mean'], color='red', linestyle='--', label=f"Media: {stats_train['mean']:.1f}")
plt.axvline(stats_train['50%'], color='green', linestyle='--', label=f"Mediana: {stats_train['50%']:.1f}")
plt.axvline(stats_train['25%'], color='orange', linestyle=':', label=f"25%: {stats_train['25%']:.1f}")
plt.axvline(stats_train['75%'], color='purple', linestyle=':', label=f"75%: {stats_train['75%']:.1f}")
plt.title('Distribución - Longitud de Consulta (train)')
plt.xlabel('Longitud')
plt.ylabel('Frecuencia')
plt.legend()

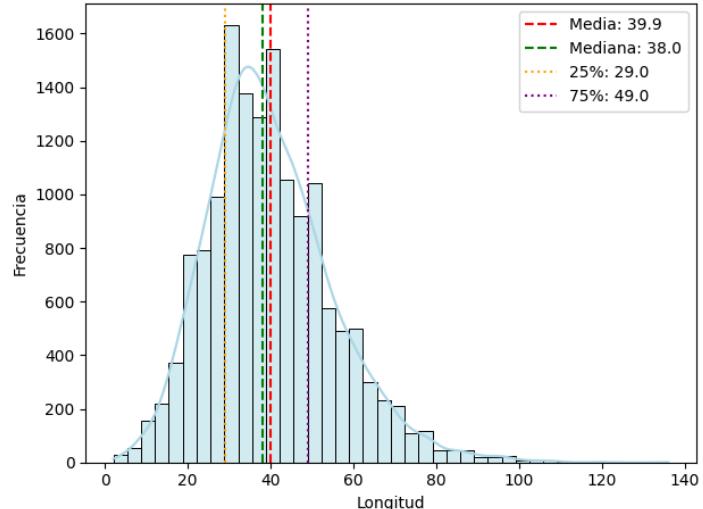
plt.tight_layout()
plt.show()
```



Boxplot - Longitud de Consulta (train)



Distribución - Longitud de Consulta (train)



```
train_data['intent'].value_counts()
```



count

intent	count
translate	100
transfer	100
timer	100
definition	100
meaning_of_life	100
...	...
income	100
order	100
traffic	100
order_checks	100
card_declined	100

150 rows × 1 columns

```
from sklearn.preprocessing import LabelEncoder
from scipy import stats
```

```
# Asegúrate de tener text_length
df_current['text_length'] = df_current['text'].apply(len)

# Tendencia central y dispersión
mean = df_current['text_length'].mean()
median = df_current['text_length'].median()
mode = df_current['text_length'].mode()[0]
geom_mean = stats.gmean(df_current['text_length'].to_numpy())
harm_mean = stats.hmean(df_current['text_length'].to_numpy())
std_dev = df_current['text_length'].std()

# Correlación y covarianza con 'intent' (primero codificamos)
le = LabelEncoder()
df_current['intent_encoded'] = le.fit_transform(df_current['intent'])

correlation = df_current[['text_length', 'intent_encoded']].corr().iloc[0, 1]
covariance = df_current[['text_length', 'intent_encoded']].cov().iloc[0, 1]

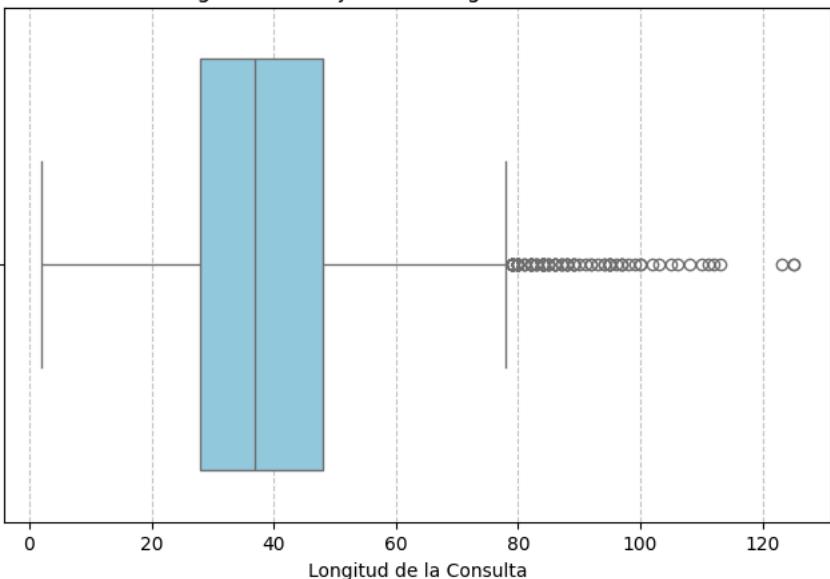
# Mostrar resultados
print(f"Media: {mean:.2f}")
print(f"Mediana: {median}")
print(f"Moda: {mode}")
```

```
print(f"Media geométrica: {geom_mean:.2f}")
print(f"Media armónica: {harm_mean:.2f}")
print(f"Desviación estándar: {std_dev:.2f}")
print(f"Correlación entre longitud e intención codificada: {correlation:.4f}")
print(f"Covarianza entre longitud e intención codificada: {covariance:.2f}")
```

```
Media: 39.31
Mediana: 37.0
Moda: 30
Media geométrica: 36.14
Media armónica: 32.51
Desviación estándar: 15.76
Correlación entre longitud e intención codificada: -0.0507
Covarianza entre longitud e intención codificada: -34.63
```

```
plt.figure(figsize=(8, 5))
sns.boxplot(x=df_current['text_length'], color='skyblue')
plt.title(f'Diagrama de Caja de la Longitud de Consultas')
plt.xlabel('Longitud de la Consulta')
plt.grid(axis='x', linestyle='--', alpha=0.7)
plt.show()
```

Diagrama de Caja de la Longitud de Consultas



```
df_full_main_subsets = pd.concat([train_data, val_data, test_data]).copy()

print(f"Número total de registros en los subconjuntos principales (train + val + test): {len(df_full_main_subsets)}")

# 2. Verifica si hay entradas de texto duplicadas (ignorando la intención)
# 'keep=False' marca TODAS las ocurrencias de un valor duplicado como True.
# Esto nos ayuda a ver todas las instancias de un texto repetido.
num_duplicates_text = df_full_main_subsets['text'].duplicated(keep=False).sum()

print(f"\nNúmero de consultas de texto duplicadas (solo texto, intenciones potencialmente diferentes): {num_duplicates_text}")

if num_duplicates_text > 0:
    print("\nEjemplos de consultas de texto que aparecen más de una vez (las primeras 10 consultas únicas con duplicados):")
    # Obtiene todas las filas que tienen un texto duplicado
    duplicated_text_rows = df_full_main_subsets[df_full_main_subsets['text'].duplicated(keep=False)].sort_values('text')
    # Muestra los conteos de valores para los 5 textos más frecuentemente duplicados
    print(duplicated_text_rows['text'].value_counts().head(5).to_string())
    print("\nFilas completas para las primeras 10 ocurrencias de textos duplicados (ordenadas por texto):")
    # .to_string() ayuda a mostrar todas las columnas/filas de forma ordenada
    print(duplicated_text_rows.head(10).to_string())
else:
    print("No se encontraron consultas de texto duplicadas exactas en los subconjuntos principales.")

# 3. Verifica si hay registros completamente duplicados (texto IDÉNTICO E intención IDÉNTICA)
# `duplicated()` por defecto verifica todas las columnas.
num_duplicates_full_row = df_full_main_subsets.duplicated().sum()

print(f"\nNúmero de registros completamente duplicados (texto e intención idénticos): {num_duplicates_full_row}")

# Si hay duplicados completos, muestra algunos ejemplos
if num_duplicates_full_row > 0:
    print("\nEjemplos de registros completamente duplicados (texto e intención idénticos):")
```

```
print(df_full_main_subsets[df_full_main_subsets.duplicated(keep=False)].head().to_string())
else:
    print("No se encontraron registros completamente duplicados (texto e intención idénticos).")
```

→ Número total de registros en los subconjuntos principales (train + val + test): 22500

Número de consultas de texto duplicadas (solo texto, intenciones potencialmente diferentes): 10

Ejemplos de consultas de texto que aparecen más de una vez (las primeras 10 consultas únicas con duplicados):  
text

hey what's up	2
turn up your volume	2
what is on my to do list	2
what's your designation	2
where did you grow up	2

Filas completas para las primeras 10 ocurrencias de textos duplicados (ordenadas por texto):

	text	intent
11896	hey what's up	greeting
2369	hey what's up	greeting
11130	turn up your volume	whisper_mode
1794	turn up your volume	change_volume
7424	what is on my to do list	todo_list
1011	what is on my to do list	reminder
12066	what's your designation	what_is_your_name
938	what's your designation	user_name
14018	where did you grow up	how_old_are_you
599	where did you grow up	where_are_you_from

Número de registros completamente duplicados (texto e intención idénticos): 1

Ejemplos de registros completamente duplicados (texto e intención idénticos):

	text	intent
11896	hey what's up	greeting
2369	hey what's up	greeting

import re

```
def clean_text(text):
    text = text.lower() # Convertir a minúsculas
    text = re.sub(r'\d+', '', text) # Eliminar números
    text = re.sub(r'[\w\s]', ' ', text) # Eliminar puntuación
    text = re.sub(r'\s+', ' ', text) # Reemplazar múltiples espacios por uno solo
    text = text.strip() # Eliminar espacios al inicio y al final
    return text

# Aplicar a los tres datasets
train_data['clean_text'] = train_data['text'].apply(clean_text)
val_data['clean_text'] = val_data['text'].apply(clean_text)
test_data['clean_text'] = test_data['text'].apply(clean_text)
```

train\_data[['text', 'clean\_text']].head()

	text	clean_text
0	what expression would i use to say i love you ...	what expression would i use to say i love you ...
1	can you tell me how to say 'i do not speak muc...	can you tell me how to say i do not speak much...
2	what is the equivalent of, 'life is good' in f...	what is the equivalent of life is good in french
3	tell me how to say, 'it is a beautiful morning...	tell me how to say it is a beautiful morning i...
4	if i were mongolian, how would i say that i am...	if i were mongolian how would i say that i am ...

val\_data[['text', 'clean\_text']].head()

	text	clean_text
0	in spanish, meet me tomorrow is said how	in spanish meet me tomorrow is said how
1	in french, how do i say, see you later	in french how do i say see you later
2	how do you say hello in japanese	how do you say hello in japanese
3	how do i ask about the weather in chinese	how do i ask about the weather in chinese
4	how can i say "cancel my order" in french	how can i say cancel my order in french

```
test_data[['text', 'clean_text']].head()
```

	text	clean_text
0	how would you say fly in italian	how would you say fly in italian
1	what's the spanish word for pasta	whats the spanish word for pasta
2	how would they say butter in zambia	how would they say butter in zambia
3	how do you say fast in spanish	how do you say fast in spanish
4	what's the word for trees in norway	whats the word for trees in norway

```
from sentence_transformers import SentenceTransformer
import pandas as pd
import numpy as np # Para manejar los embeddings

# Usaremos 'all-MiniLM-L6-v2' que es un buen equilibrio entre rendimiento y tamaño/velocidad.
# Se descargará automáticamente la primera vez que lo ejecutes.
print("Cargando el modelo de SentenceTransformer...")
model = SentenceTransformer('all-MiniLM-L6-v2')
print("Modelo cargado exitosamente.")

# --- Generar Embeddings para cada subconjunto ---

print("\nGenerando embeddings para el subconjunto de TRAIN...")
train_embeddings = model.encode(train_data['text'].tolist(), show_progress_bar=True)
print(f"Embeddings de TRAIN generados. Forma: {train_embeddings.shape}")
# Guardar los embeddings en el DataFrame
train_data['embeddings'] = list(train_embeddings) # Guardar como lista de arrays

print("\nGenerando embeddings para el subconjunto de VALIDACIÓN...")
val_embeddings = model.encode(val_data['text'].tolist(), show_progress_bar=True)
print(f"Embeddings de VALIDACIÓN generados. Forma: {val_embeddings.shape}")
# Guardar los embeddings en el DataFrame
val_data['embeddings'] = list(val_embeddings)

print("\nGenerando embeddings para el subconjunto de TEST...")
test_embeddings = model.encode(test_data['text'].tolist(), show_progress_bar=True)
print(f"Embeddings de TEST generados. Forma: {test_embeddings.shape}")
# Guardar los embeddings en el DataFrame
test_data['embeddings'] = list(test_embeddings)

# --- Opcional: Ver las primeras filas con los nuevos embeddings ---
print("\nPrimeras filas del DataFrame de TRAIN con embeddings:")
print(train_data.head())
print("\nPrimeras filas del DataFrame de VALIDACIÓN con embeddings:")
print(val_data.head())
print("\nPrimeras filas del DataFrame de TEST con embeddings:")
print(test_data.head())
```

```
↳ Cargando el modelo de SentenceTransformer...
/usr/local/lib/python3.11/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens), set it as an environment variable, or use a token passed via the HF_HUB_TOKEN environment variable.
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
Modelo cargado exitosamente.

Generando embeddings para el subconjunto de TRAIN...
Batches: 100% 469/469 [01:45<00:00,  9.83it/s]
Embeddings de TRAIN generados. Forma: (15000, 384)

Generando embeddings para el subconjunto de VALIDACIÓN...
Batches: 100% 94/94 [00:19<00:00,  8.53it/s]
Embeddings de VALIDACIÓN generados. Forma: (3000, 384)

Generando embeddings para el subconjunto de TEST...
Batches: 100% 141/141 [00:33<00:00,  8.13it/s]
Embeddings de TEST generados. Forma: (4500, 384)

Primeras filas del DataFrame de TRAIN con embeddings:
   text      intent \
0 what expression would i use to say i love you ... translate
1 can you tell me how to say 'i do not speak muc... translate
2 what is the equivalent of, 'life is good' in f... translate
3 tell me how to say, 'it is a beautiful morning... translate
4 if i were mongolian, how would i say that i am... translate

   clean_text \
0 what expression would i use to say i love you ...
1 can you tell me how to say i do not speak much...
2 what is the equivalent of life is good in french
3 tell me how to say it is a beautiful morning i...
4 if i were mongolian how would i say that i am ...

   embeddings
0 [-0.1015782, 0.094667904, 0.053707164, 0.02137...
1 [0.11241164, 0.036940716, -0.005775032, -0.003...
2 [-0.03923412, -0.016597776, 0.014678889, -0.02...
3 [-0.0042731697, 0.10736326, 0.061864898, 0.057...
4 [0.017006498, 0.080711834, 0.053496458, 0.0274...

Primeras filas del DataFrame de VALIDACIÓN con embeddings:
   text      intent \
0 in spanish, meet me tomorrow is said how translate
1 in french, how do i say, see you later translate
2 how do you say hello in japanese translate
3 how do i ask about the weather in chinese translate
4 how can i say "cancel my order" in french translate

   clean_text \
0 in spanish meet me tomorrow is said how
1 in french how do i say see you later
2 how do you say hello in japanese
3 how do i ask about the weather in chinese
4 how can i say cancel my order in french

   embeddings
0 [-0.031512525, 0.015618292, 0.09258982, 0.0648...
1 [-0.030275127, 0.010276049, 0.058302507, 0.011...
2 [-0.073753975, 0.06647398, 0.045967467, 0.0162...
3 [-0.020966126, 0.08849304, 0.11366967, 0.09627...
4 [-0.03600758, 0.07214296, 0.09701091, -0.01435...

Primeras filas del DataFrame de TEST con embeddings:
   text      intent \
0 how would you say fly in italian translate
1 what's the spanish word for pasta translate
2 how would they say butter in zambia translate
3 how do you say fast in spanish translate
4 what's the word for trees in norway translate

   clean_text \
0 how would you say fly in italian
1 whats the spanish word for pasta
2 how would they say butter in zambia
3 how do you say fast in spanish
4 whats the word for trees in norway

   embeddings
0 [-0.018759524, 0.062015172, -0.028805563, 0.05...
1 [-0.044872522, -0.078197256, -0.048782792, 0.0...
2 [-0.02842604, -0.004796487, -0.10109748, 0.041...
3 [0.026341321, 0.019471763, 0.020619301, 0.0558...
4 [0.02271745, 0.07555067, 0.041861523, 0.040956...
```

```

from sklearn.manifold import TSNE

# Es buena práctica trabajar con una copia para no modificar el DataFrame original si no es necesario
# Por ejemplo, para el conjunto de TRAIN:
embeddings_train = np.array(train_data['embeddings'].tolist())
intents_train = train_data['intent'] # Guardamos las intenciones para colorear el plot

print("Aplicando t-SNE a los embeddings de TRAIN (esto puede tomar tiempo)...")
# Parámetros de t-SNE:
# n_components: 2 para visualizar en 2D
# random_state: Para asegurar la reproducibilidad de los resultados
# perplexity: Ajusta el balance entre el enfoque en vecinos cercanos y lejanos.
#           Un buen rango es entre 5 y 50. Experimenta si los clusters no se ven bien.
# n_iter: Número de iteraciones para la optimización. Un número mayor da mejores resultados pero tarda más.
tsne = TSNE(n_components=2, random_state=42, perplexity=30, n_iter=1000, learning_rate=200)

# Aplicar t-SNE
tsne_results_train = tsne.fit_transform(embeddings_train)

print(f"Reducción de dimensionalidad t-SNE para TRAIN completa. Forma: {tsne_results_train.shape}")

# Añadir los resultados de t-SNE al DataFrame de train_data
train_data['tsne_x'] = tsne_results_train[:,0]
train_data['tsne_y'] = tsne_results_train[:,1]

# --- Visualización de los resultados de t-SNE (solo para una muestra o un conjunto pequeño) ---
# Para un dataset de 15,000 puntos, graficar todos los puntos puede ser lento o denso.
# Podrías considerar graficar solo una muestra aleatoria o solo los puntos más relevantes.
# Sin embargo, para fines de demostración, graficaremos todos.

plt.figure(figsize=(12, 10))
# Para visualizar los 150 clusters con colores distintos, necesitarás una paleta de colores grande.
# sns.color_palette("hsv", 150) es una opción, pero puede ser difícil diferenciar los colores.
# Podrías optar por graficar sin colores de intención inicialmente, o con una muestra.
# Para el ejemplo, usaremos un color general para ver la forma de los clusters.
sns.scatterplot(
    x='tsne_x', y='tsne_y',
    hue='intent', # Colorear por intención (si quieres ver cómo se agrupan)
    palette=sns.color_palette("tab20", 15), # Puedes ajustar la paleta o elegir un subconjunto
    legend='full', # Muestra la leyenda de las intenciones
    alpha=0.7,
    s=5, # Tamaño de los puntos
    data=train_data.sample(n=2000, random_state=42) # Muestra 2000 puntos para una visualización más rápida
    # data=train_data # Para graficar todos los puntos (puede ser muy lento o denso)
)
plt.title('Visualización t-SNE de Embeddings de Consultas (Muestra de TRAIN)')
plt.xlabel('Componente t-SNE 1')
plt.ylabel('Componente t-SNE 2')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.) # Mover leyenda fuera
plt.show()

# --- Repetir para val_data y test_data si lo deseas ---
# La aplicación de t-SNE en sets de validación y prueba es para la evaluación final,
# pero el proceso de "refinamiento" en TextLens se haría principalmente en el set de entrenamiento.
# Sin embargo, para tener los datos listos, puedes aplicar t-SNE a los otros subconjuntos también.

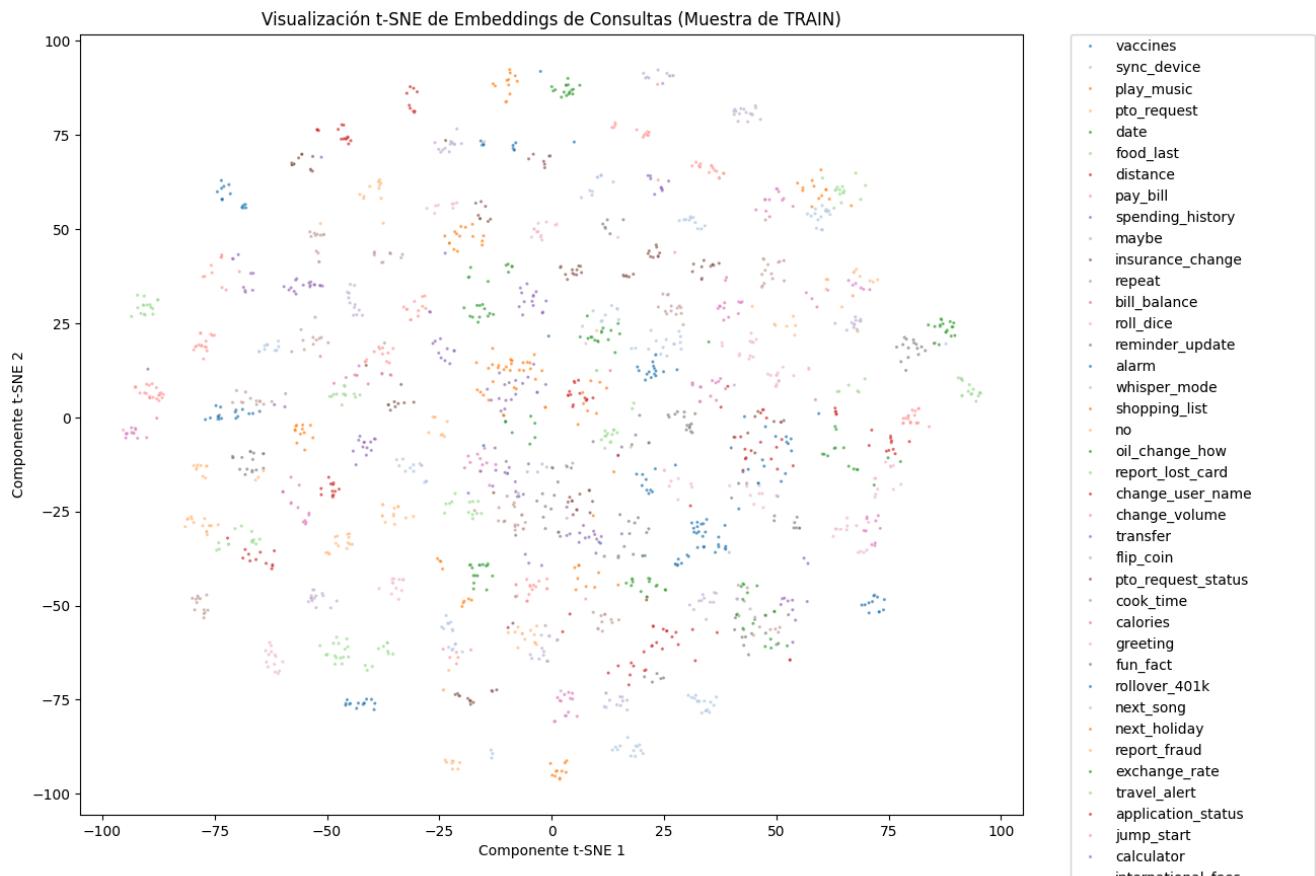
# Para val_data:
embeddings_val = np.array(val_data['embeddings'].tolist())
tsne_results_val = tsne.fit_transform(embeddings_val) # Usa el mismo objeto TSNE si quieres resultados comparables
val_data['tsne_x'] = tsne_results_val[:,0]
val_data['tsne_y'] = tsne_results_val[:,1]

# Para test_data:
embeddings_test = np.array(test_data['embeddings'].tolist())
tsne_results_test = tsne.fit_transform(embeddings_test) # Usa el mismo objeto TSNE si quieres resultados comparables
test_data['tsne_x'] = tsne_results_test[:,0]
test_data['tsne_y'] = tsne_results_test[:,1]

print("\nResultados de t-SNE añadidos a los DataFrames.")
print(train_data.head())
print(val_data.head())
print(test_data.head())

```

```
Aplicando t-SNE a los embeddings de TRAIN (esto puede tomar tiempo)...
/usr/local/lib/python3.11/dist-packages/scikit-learn/manifold/_t_sne.py:1164: FutureWarning: 'n_iter' was renamed to 'max_iter' in ver
warnings.warn(
Reducción de dimensionalidad t-SNE para TRAIN completa. Forma: (15000, 2)
<ipython-input-33-0ef3bd8bc597>:36: UserWarning:
The palette list has fewer values (15) than needed (150) and will cycle, which may produce an uninterpretable plot.
sns.scatterplot()
```



```

msg_type
• accept_reservations
• calendar
• interest_rate
• apr
• thank_you
• translate
• do_you_have_pets
• damaged_card
• who_do_you_work_for
• shopping_list_update
• tire_change
• change_accent
• card_declined
• what_song
• goodbye
• restaurant_reviews
• meeting_schedule
• uber
• oil_change_when
• w2
• restaurant_reservation
• min_payment
• todo_list_update
• weather
• nutrition_info
• ingredients_list
• mpg
• balance
• what_can_i_ask_you
• where_are_you_from
• text
• credit_limit_change
• change_language
• lost_luggage
• redeem_rewards
• measurement_conversion
• are_you_a_bot
• what_is_your_name
• routing
• cancel_reservation
• how_busy
• bill_due
• meaning_of_life
• cancel
• travel_notification
• directions
• schedule_maintenance
• carry_on
• travel_suggestion
• payday
• book_hotel
• pin_change
• tire_pressure
• account_blocked
• transactions
• reset_settings
• user_name
• car_rental
• current_location
• gas_type
• find_phone

```

```

/usr/local/lib/python3.11/dist-packages/sklearn/manifold/_t_sne.py:1164: FutureWarning: 'n_iter' was renamed to 'max_iter' in ver
warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/manifold/_t_sne.py:1164: FutureWarning: 'n_iter' was renamed to 'max_iter' in ver
warnings.warn(

```

Resultados de t-SNE añadidos a los DataFrames.

	text	intent	\
0	what expression would i use to say i love you ...	translate	
1	can you tell me how to say 'i do not speak muc...	translate	
2	what is the equivalent of, 'life is good' in f...	translate	
3	tell me how to say, 'it is a beautiful morning...	translate	
4	if i were mongolian, how would i say that i am...	translate	

	clean_text	\
0	what expression would i use to say i love you ...	
1	can you tell me how to say i do not speak much...	
2	what is the equivalent of life is good in french	
3	tell me how to say it is a beautiful morning i...	
4	if i were mongolian how would i say that i am ...	

	embeddings	tsne_x	tsne_y
0	[-0.1015782, 0.094667904, 0.053707164, 0.02137...	8.103859	26.703190
1	[0.11241164, 0.036940716, -0.005775032, -0.003...	12.253810	24.432232
2	[-0.03923412, -0.016597776, 0.014678889, -0.02...	7.961337	21.374725
3	[-0.0042731697, 0.10736326, 0.061864898, 0.057...	7.945118	26.955343
4	[0.017006498, 0.080711834, 0.053496458, 0.0274...	14.422037	21.117867

	text	intent	\
0	in spanish, meet me tomorrow is said how	translate	
1	in french, how do i say, see you later	translate	
2	how do you say hello in japanese	translate	
3	how do i ask about the weather in chinese	translate	
4	how can i say "cancel my order" in french	translate	

```
clean_text \
0    in spanish meet me tomorrow is said how
1      in french how do i say see you later
2          how do you say hello in japanese
3 how do i ask about the weather in chinese
4   how can i say cancel my order in french

embeddings      tsne_x      tsne_y
0 [-0.031512525, 0.015618292, 0.09258982, 0.0648... -52.517448 -10.216786
1 [-0.030275127, 0.010276049, 0.058302507, 0.011... -53.896164 -7.364371
2 [-0.073753975, 0.06647398, 0.045967467, 0.0162... -53.877865 -9.124257
3 [-0.020966126, 0.08849304, 0.11366967, 0.09627... 0.321591 -20.187860
4 [-0.03600758, 0.07214296, 0.09701001, -0.01435... -29.497345 1.194914

text      intent \
0   how would you say fly in italian  translate
1 what's the spanish word for pasta  translate
2 how would they say butter in zambia  translate
3   how do you say fast in spanish  translate
4 what's the word for trees in norway  translate

clean_text \
0   how would you say fly in italian
1 whats the spanish word for pasta
2 how would they say butter in zambia
3   how do you say fast in spanish
4 whats the word for trees in norway

embeddings      tsne_x      tsne_y
0 [-0.018759524, 0.062015172, -0.028805563, 0.05... 36.793354 18.454105
1 [-0.044872522, -0.078197256, -0.048782792, 0.0... 21.818722 -61.770527
2 [-0.02842604, -0.004796487, -0.10109748, 0.041... 9.106302 -75.474045
3 [0.026341321, 0.019471763, 0.020619301, 0.0558... 43.689610 16.405338
4 [0.02271745, 0.07555067, 0.041861523, 0.040956... 38.824192 16.094185
```

```
# --- Visualización del Desafío de Granularidad y Solapamiento ---

plt.figure(figsize=(14, 12)) # Ajusta el tamaño para mejor visualización
# Usaremos una paleta de colores diversa para las 150 intenciones.
# Para evitar una leyenda abrumadora, podemos graficar solo una muestra para ver el patrón,
# o graficar solo las 150 intenciones, sabiendo que muchos colores serán similares.
# Para este gráfico, es más importante ver la 'mezcla' que identificar cada intención.

# Podemos limitar el número de colores o graficar solo una muestra para claridad
# Si quieras mostrar todas las 150 intenciones con colores, seaborn.color_palette puede manejarlo,
# pero la leyenda será muy grande y los colores podrían ser difíciles de distinguir.
# Una forma de visualizarlo mejor es con un 'scatter plot' de los puntos t-SNE:

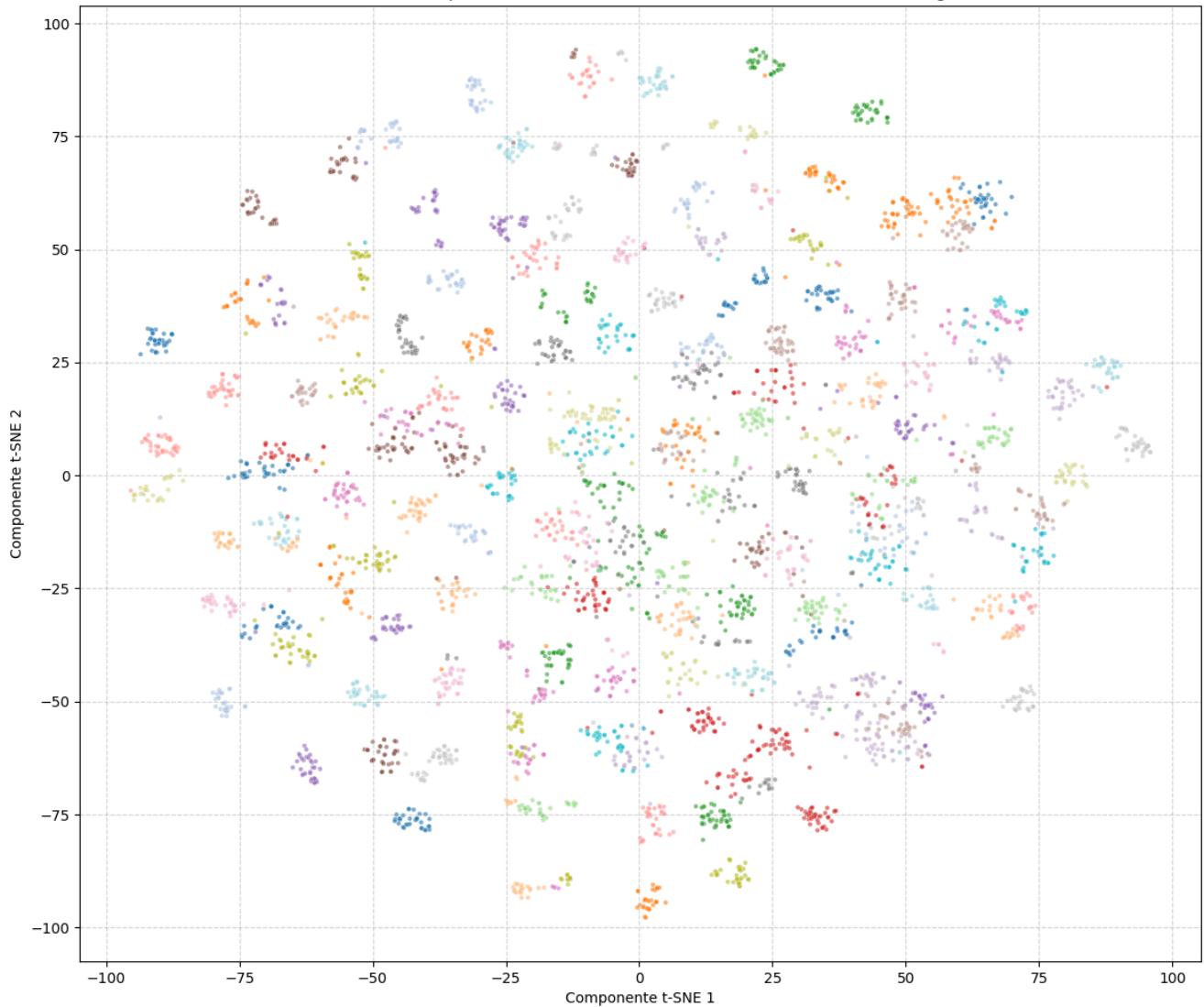
sns.scatterplot(
    x='tsne_x', y='tsne_y',
    hue='intent', # Colorear por la intención real
    palette='tab20', # Una paleta predefinida que tiene 20 colores. Se ciclará por las 150.
                    # Puedes probar 'hsv' o 'viridis' para más variedad si quieras,
                    # pero 150 colores distintos y perceptibles son difíciles.
    legend=False, # Ocultar la leyenda para evitar que sature el gráfico,
                  # ya que tenemos 150 intenciones y sería ilegible.
                  # La mezcla de colores ya visualiza el problema.
    alpha=0.6, # Transparencia para ver la densidad de puntos
    s=10,       # Tamaño de los puntos
    data=train_data.sample(n=5000, random_state=42) # Muestra de 5000 puntos para mayor claridad,
                                                # dado que 15k es denso
    # data=train_data # Descomentar si quieres intentar graficar todos (puede ser muy lento/denso)
)
plt.title('Desafío 2: Solapamiento Semántico de Intenciones (t-SNE de Embeddings)')
plt.xlabel('Componente t-SNE 1')
plt.ylabel('Componente t-SNE 2')
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()

# Opcional: Un gráfico con solo algunas intenciones específicas para mostrar solapamiento
# Define algunas intenciones que esperas que se solapen o sean similares
# Por ejemplo, de tus ejemplos ambiguos: 'whisper_mode', 'change_volume', 'todo_list', 'reminder'
# O intenciones que sabes que son semánticamente cercanas
target_intents = ['pay_bill', 'transfer', 'balance', 'change_volume', 'whisper_mode', 'translate', 'money_transfer', 'volume_up', 'repeat']
df_subset_intents = train_data[train_data['intent'].isin(target_intents)].sample(n=min(500, len(train_data[train_data['intent'].isin(target_intents)])))

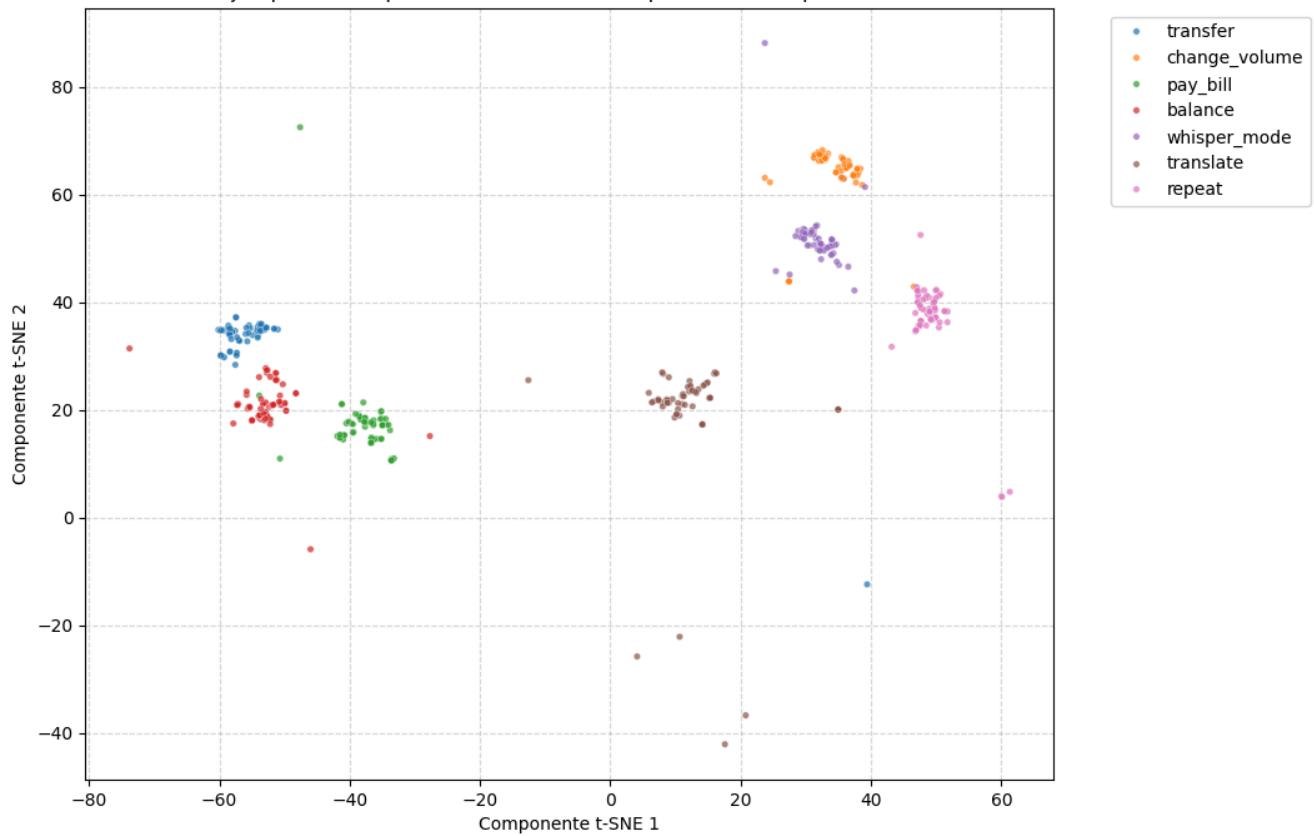
if not df_subset_intents.empty:
    plt.figure(figsize=(10, 8))
    sns.scatterplot(
        x='tsne_x', y='tsne_y',
        hue='intent',
        palette='tab10', # Una paleta con menos colores, más fáciles de distinguir
        legend='full',
        alpha=0.7,
        s=15,
        data=df_subset_intents
    )
    plt.title('Ejemplo de Solapamiento: Intenciones Específicas en Espacio t-SNE')
    plt.xlabel('Componente t-SNE 1')
    plt.ylabel('Componente t-SNE 2')
    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.show()
```



## Desafío 2: Solapamiento Semántico de Intenciones (t-SNE de Embeddings)



Ejemplo de Solapamiento: Intenciones Específicas en Espacio t-SNE



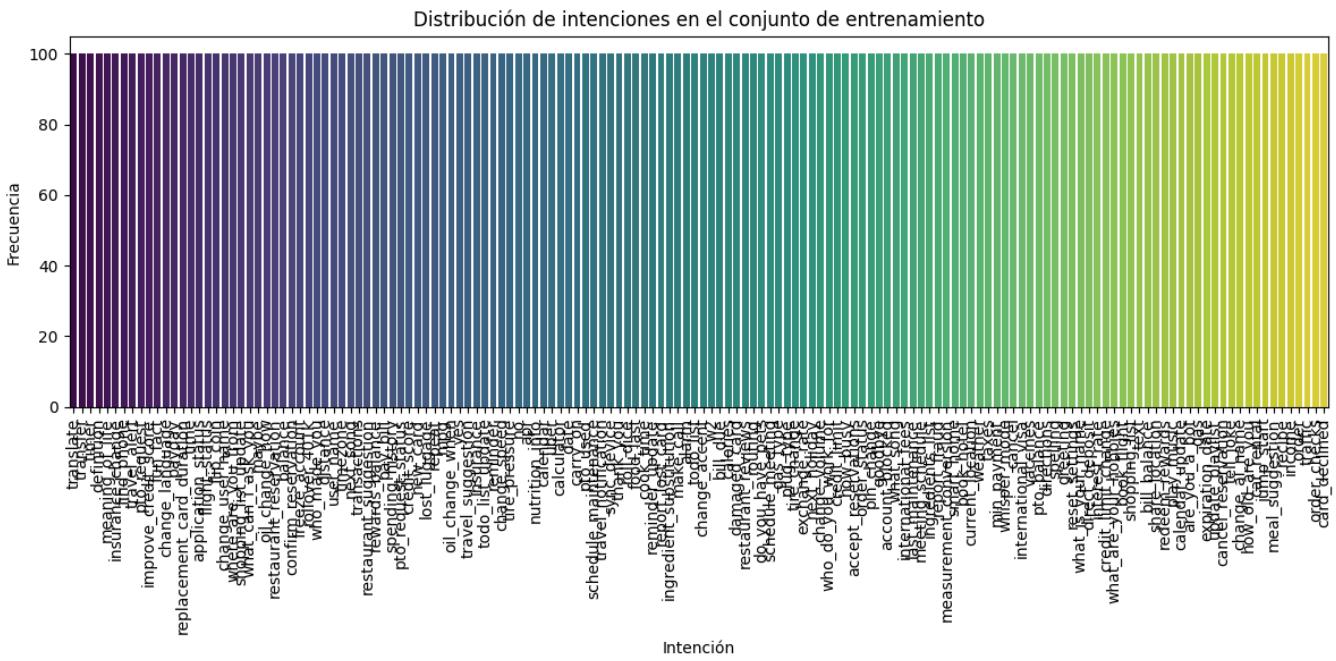
```
# Contar ocurrencias por clase de intención
intent_counts = train_data["intent"].value_counts().sort_values(ascending=False)

# Crear el gráfico de barras
plt.figure(figsize=(12, 6))
sns.barplot(x=intent_counts.index, y=intent_counts.values, palette="viridis")
plt.xticks(rotation=90)
plt.title("Distribución de intenciones en el conjunto de entrenamiento")
plt.xlabel("Intención")
plt.ylabel("Frecuencia")
plt.tight_layout()
plt.show()
```

→ <ipython-input-35-85caf6a560ed>:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

```
sns.barplot(x=intent_counts.index, y=intent_counts.values, palette="viridis")
```



```
from sklearn.cluster import KMeans

# Definimos el número de clusters. Puedes ajustar este valor según tu problema
n_clusters = 30

# Creamos el objeto KMeans
kmeans = KMeans(n_clusters=n_clusters, random_state=42, n_init='auto')

# Ajustamos KMeans con los datos t-SNE del train y predecimos los clusters
train_data['cluster_id'] = kmeans.fit_predict(train_data[['tsne_x', 'tsne_y']])

# Predecimos clusters para val y test usando el mismo modelo entrenado en train
val_data['cluster_id'] = kmeans.predict(val_data[['tsne_x', 'tsne_y']])
test_data['cluster_id'] = kmeans.predict(test_data[['tsne_x', 'tsne_y']])

print("Clusters asignados a train, val y test.")
print(train_data[['tsne_x', 'tsne_y', 'cluster_id']].head())
```

```
Clusters asignados a train, val y test.
  tsne_x  tsne_y  cluster_id
0    8.103859  26.703190      14
1   12.253810  24.432232      14
2    7.961337  21.374725      14
3   7.945118  26.955343      14
4  14.422037  21.117867      14

train_data[['tsne_x', 'tsne_y', 'intent', 'text', 'cluster_id']].to_csv("tsne_train_data.csv", index=False)
val_data[['tsne_x', 'tsne_y', 'intent', 'text', 'cluster_id']].to_csv("tsne_val_data.csv", index=False)
test_data[['tsne_x', 'tsne_y', 'intent', 'text', 'cluster_id']].to_csv("tsne_test_data.csv", index=False)

# Instalar Dash
!pip install dash
!pip install pyngrok
print("Dash y Pyngrok instalados correctamente.")

Requirement already satisfied: dash in /usr/local/lib/python3.11/dist-packages (3.0.4)
Requirement already satisfied: Flask<3.1,>=1.0.4 in /usr/local/lib/python3.11/dist-packages (from dash) (3.0.3)
Requirement already satisfied: Werkzeug<3.1 in /usr/local/lib/python3.11/dist-packages (from dash) (3.0.6)
Requirement already satisfied: plotly>=5.0.0 in /usr/local/lib/python3.11/dist-packages (from dash) (5.24.1)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.11/dist-packages (from dash) (8.7.0)
Requirement already satisfied: typing-extensions>=4.1.1 in /usr/local/lib/python3.11/dist-packages (from dash) (4.13.2)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from dash) (2.32.3)
Requirement already satisfied: retrying in /usr/local/lib/python3.11/dist-packages (from dash) (1.3.4)
Requirement already satisfied: nest-asyncio in /usr/local/lib/python3.11/dist-packages (from dash) (1.6.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from dash) (75.2.0)
Requirement already satisfied: Jinja2>=3.1.2 in /usr/local/lib/python3.11/dist-packages (from Flask<3.1,>=1.0.4->dash) (3.1.6)
Requirement already satisfied: itsdangerous>=2.1.2 in /usr/local/lib/python3.11/dist-packages (from Flask<3.1,>=1.0.4->dash) (2.2.0)
Requirement already satisfied: click>=8.1.3 in /usr/local/lib/python3.11/dist-packages (from Flask<3.1,>=1.0.4->dash) (8.2.1)
Requirement already satisfied: blinker>=1.6.2 in /usr/local/lib/python3.11/dist-packages (from Flask<3.1,>=1.0.4->dash) (1.9.0)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.11/dist-packages (from plotly>=5.0.0->dash) (9.1.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from plotly>=5.0.0->dash) (24.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-packages (from Werkzeug<3.1->dash) (3.0.2)
Requirement already satisfied: zipp>=3.20 in /usr/local/lib/python3.11/dist-packages (from importlib-metadata->dash) (3.22.0)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->dash) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->dash) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->dash) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests->dash) (2025.4.26)
Requirement already satisfied: six>=1.7.0 in /usr/local/lib/python3.11/dist-packages (from retrying->dash) (1.17.0)
Requirement already satisfied: pyngrok in /usr/local/lib/python3.11/dist-packages (7.2.9)
Requirement already satisfied: PyYAML>=5.1 in /usr/local/lib/python3.11/dist-packages (from pyngrok) (6.0.2)
Dash y Pyngrok instalados correctamente.
```

```
!ngrok config add-authtoken 2y1MAZBjtDcLzLwKl1YhsjqyaG2_61PqgvHL3hxkVsD87NUkZ
```

```
Auth token saved to configuration file: /root/.config/ngrok/ngrok.yml
```

```
!pip install nltk
import nltk
nltk.download('punkt')
nltk.download('stopwords')
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.11/dist-packages (3.9.1)
Requirement already satisfied: click in /usr/local/lib/python3.11/dist-packages (from nltk) (8.2.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.11/dist-packages (from nltk) (1.5.1)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.11/dist-packages (from nltk) (2024.11.6)
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (from nltk) (4.67.1)
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
True
```

```
import dash
from dash import dcc, html
from dash.dependencies import Input, Output, State
import plotly.express as px
import plotly.graph_objects as go
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.metrics import confusion_matrix, silhouette_score, davies_bouldin_score, calinski_harabasz_score
import numpy as np
import io
import base64
from pyngrok import ngrok # Para exponer el dashboard
```

```
# --- Cargar datos ---
try:
```

```
    df_train = pd.read_csv("tsne_train_data.csv")
    df_val = pd.read_csv("tsne_val_data.csv")
```



```

        dcc.Loading(
            id="loading-metrics",
            type="circle",
            children=html.Div(id='clustering-metrics-output', style={'padding': '20px', 'backgroundColor': '#fff', 'margin': '0 auto 20px 0'}),
            html.H3("Matriz de Confusión (Intención Real vs. Cluster ID)", style={'textAlign': 'center', 'marginTop': '40px', 'color': 'black'})
        ),
        dcc.Loading(
            id="loading-confusion-matrix",
            type="circle",
            children=html.Div(id='confusion-matrix-graph')
        ),
    ],
    style={'padding': '20px'}
),
]),
],
),
),

# dcc.Store para almacenar el DataFrame procesado
dcc.Store(id='stored-df')
])

# --- Callbacks ---

# Callback para actualizar el gráfico de dispersión Y almacenar el DataFrame procesado
@app.callback(
    Output('main-scatter-plot', 'figure'),
    Output('stored-df', 'data'),
    Input('color-by-selector', 'value'),
    Input('num-clusters-input', 'value')
)
def update_scatter_plot_and_store_df(color_by_value, num_clusters):
    if num_clusters is None or num_clusters < 2:
        num_clusters = 2

    df_current = df_combined_original.copy()

    try:
        if 'tsne_x' in df_current.columns and 'tsne_y' in df_current.columns:
            # Asegúrate de que los datos de entrada para KMeans sean numéricos
            kmeans = KMeans(n_clusters=int(num_clusters), random_state=42, n_init='auto')
            df_current['cluster_id'] = kmeans.fit_predict(df_current[['tsne_x', 'tsne_y']])
            # Convertir cluster_id a string para que Plotly no lo trate como un valor numérico continuo
            df_current['cluster_id'] = df_current['cluster_id'].astype(str)
        else:
            print("Advertencia: Columnas 'tsne_x' o 'tsne_y' no encontradas para KMeans.")
            df_current['cluster_id'] = 'N/A'
    except Exception as e:
        print(f"Error al recalcular KMeans: {e}")
        df_current['cluster_id'] = 'Error'

    fig = px.scatter(
        df_current,
        x="tsne_x",
        y="tsne_y",
        color=color_by_value,
        hover_data={'text': True, 'intent': True, 'cluster_id': True, 'id': True, 'tsne_x': False, 'tsne_y': False},
        height=700,
        template="plotly_white"
    )

    fig.update_traces(
        hovertemplate="Intención: %{customdata[1]}  
Cluster: %{customdata[2]}  
Consulta: %{customdata[0]}  
")
    if color_by_value == 'intent' and len(df_combined_original['intent'].unique()) > 20:
        fig.update_layout(showlegend=False)
        fig.update_layout(title_font_size=20)

    fig.update_layout(
        dragmode='pan'
    )

    stored_data = df_current.to_dict('records')

    return fig, stored_data

# Callback para actualizar el panel de detalles al seleccionar o hacer clic en puntos
@app.callback(
    Output('selection-details', 'children'),
    Input('main-scatter-plot', 'selectedData'),
    Input('main-scatter-plot', 'clickData'),
    State('stored-df', 'data')
)
def display_selected_data(selected_data, click_data, stored_df_data):
    if stored_df_data is None:

```

## Práctica de Laboratorio: Análisis Exploratorio de Datos - Data Wrangling

Docente: [Ana María Cuadros](#) Valdivia

Alumna: Cecilia del Pilar Vilca Alvites

Para realizar el Análisis Exploratorio de datos, lo primero que deberíamos hacer es intentar responder a las siguientes preguntas (data wrangling):

El dataset CLINC150 es un conjunto de datos ampliamente utilizado en el campo del Procesamiento de Lenguaje Natural (NLP), diseñado específicamente para la tarea de clasificación de intenciones en sistemas de diálogo. Fue publicado en 2019 por investigadores de la Universidad de Michigan y se ha consolidado como un benchmark estándar para evaluar la comprensión de intenciones en modelos de lenguaje.

### Características Principales:

- Idioma: Inglés
- Formato original: JSON, con estructura de pares ["consulta", "intención"]
- Total de clases: 150 categorías de intención, incluyendo una clase especial `out_of_scope (OOS)` para consultas que no pertenecen a ningún dominio definido
- Dominio: Abarca consultas de banca, viajes, clima, entretenimiento, salud, alarmas, y más
- Número total de consultas: Aproximadamente 22,500 frases de texto.

### Paso 1: Analiza el comportamiento de tus datos.

- Un registro es una entidad, describa que representa un registro

En el dataset CLINC150, un registro representa una única consulta de usuario ("utterance") junto con su correspondiente categoría de intención ("intent"). Cada registro encapsula la interacción mínima de un usuario con un sistema de diálogo orientado a tareas: lo que el usuario dijo y lo que el sistema debe entender que quiere hacer.

#### Ejemplo de registros:

utterance	intent
what expression would i use to say i love you if i were an italian	translate
can you tell me how to say 'i do not speak much spanish', in spanish	translate
what is the equivalent of, 'life is good' in french	translate
tell me how to say, 'it is a beautiful morning' in italian	translate
if i were mongolian, how would i say that i am a tourist	translate

En CLINC150, la consulta es "what expression would i use to say i love you if i were an italian" y la intención es "translate".

- ¿Cuántos registros hay?

- Entrenamiento (train): Este es el subconjunto más grande del dataset y su función principal es exponer el algoritmo a la mayoría de los patrones y variaciones de los datos. Contiene 15,000 registros, cada uno compuesto por una consulta de usuario y su intención correspondiente.
- Validación (val): Este subconjunto se utilizará para ajustar hiperparámetros del modelo y para una evaluación intermedia durante el proceso de desarrollo permitiendo monitorear el rendimiento del modelo en datos no vistos y a identificar problemas como el sobreajuste (overfitting). Contiene 3,000 registros, cada uno compuesto por una consulta de usuario y su intención correspondiente.
- Prueba (test): Es el subconjunto más crítico para la evaluación final del rendimiento del sistema. Es esencial que los datos en este conjunto no hayan sido utilizados en ninguna fase de entrenamiento o ajuste de hiperparámetros. Contiene 4,500 registros, cada uno compuesto por una consulta de usuario y su intención correspondiente.
- "Fuera de Alcance" (Out-of-Scope - OOS): Estos subconjuntos están diseñados para contener consultas que no corresponden a ninguna de las intenciones predefinidas.
  - oos\_train: Contiene 100 registros de consultas fuera de alcance para el entrenamiento.
  - oos\_val: Contiene 100 registros de consultas fuera de alcance para la validación.
  - oos\_test: Contiene 1,000 registros de consultas fuera de alcance para la prueba.

Para los propósitos de este proyecto de investigación basado en TextLens y la versión del dataset CLINC150 descrita en el artículo de Larson et al. (2019), estos subconjuntos OOS no se utilizarán en el análisis principal de clustering o para el desarrollo de la analítica visual.

- ¿Son demasiado pocos?

El dataset CLINC150, con 18,000 registros bien distribuidos entre entrenamiento, validación y prueba, además de 1,200 ejemplos OOS (fuera de dominio), es adecuado para este proyecto de investigación. Aunque existen datasets masivos en PLN, CLINC150 destaca por su calidad y la definición clara de 150 categorías de intención. Esto lo convierte en un benchmark ideal para aplicar modelos LLMs y embeddings preentrenados como Instructor-large, que ya han aprendido representaciones lingüísticas complejas. En este contexto, la calidad y estructura del dataset compensan su tamaño, permitiendo un análisis significativo sin necesidad de millones de ejemplos.

- ¿Son muchos y no tenemos Capacidad (CPU+RAM) suficiente para procesarlo?

No, el dataset CLINC150 no es demasiado grande y Google Colab tiene capacidad (CPU+RAM) más que suficiente para procesarlo. Con apenas 5.6 MB, el uso de memoria es mínimo en comparación con los 12–25 GB de RAM que ofrece Google Colab.

- ¿Hay datos duplicados?

Para evaluar la presencia de registros duplicados en el dataset CLINC150, se realizó un análisis sobre el conjunto de datos, es decir: train + val + test dando en total 22,500 registros. Los resultados de esta verificación son los siguientes:

- Consultas de texto duplicadas (misma consulta, pero diferentes intenciones): Se identificaron 10 consultas de texto que aparecen más de una vez en el dataset. Sin embargo, lo crucial es que, al examinar estos casos, se observa que la mayoría de estas consultas idénticas están asociadas a *diferentes* categorías de intención. Esto es un reflejo de la ambigüedad inherente del lenguaje natural.
- Se encontró 1 registro completamente duplicado. Este caso particular corresponde a la consulta "hey what's up" asociada a la intención "greeting", la cual aparece dos veces de forma idéntica en el dataset. Aunque esto indica una duplicidad exacta de una entrada, su número es insignificante en un dataset de 22,500 registros.

```
Número total de registros en los subconjuntos principales (train + val + test): 22500
Número de consultas de texto duplicadas (solo texto, intenciones potencialmente diferentes): 10
Ejemplos de consultas de texto que aparecen más de una vez (las primeras 10 consultas únicas con duplicados):
text
hey what's up      2
turn up your volume 2
what is on my to do list 2
what's your designation 2
where did you grow up 2

Filas completas para las primeras 10 ocurrencias de textos duplicados (ordenadas por texto):
          text           intent
11896     hey what's up      greeting
2369      hey what's up      greeting
11130    turn up your volume whisper_mode
1794     turn up your volume change_volume
7424   what is on my to do list todo_list
1011   what is on my to do list reminder
12066  what's your designation what_is_your_name
938    what's your designation user_name
14018  where did you grow up  how_old_are_you
599    where did you grow up where_are_you_from

Número de registros completamente duplicados (texto e intención idénticos): 1
Ejemplos de registros completamente duplicados (texto e intención idénticos):
          text   intent
11896  hey what's up  greeting
2369  hey what's up  greeting
```

- ¿Qué datos son discretos y cuáles continuos?

text	intent	text_length
Cada consulta es una entidad textual única considerándose como un tipo de dato <b>discreto</b> ya que cada oración es una "unidad" separada.	Es un dato categórico/discreto ya que son 150 clases diferentes. Representa una categoría finita de intenciones.	<b>Es un dato cuantitativo discreto</b> , con un rango observado de <b>2 a 136</b> caracteres en el dataset.

- Muchas veces sirve obtener el tipo de datos: texto, int, double, float ¿Cuáles son los tipos de datos de cada columna?

text:

Tipo de dato: object (cadena de texto).

Descripción: Contiene la consulta o enunciado que hace el usuario, por ejemplo: "how do you say hello in japanese?".

intent:

Tipo de dato: object (texto categórico).

Descripción: Representa la intención asociada a cada consulta del usuario, como translate, weather, pay\_bill, etc.

text\_length:

Tipo de dato: int64 (entero).

Descripción: Indica la longitud (en número de caracteres) de cada consulta contenida en la columna text.

- ¿Entre qué rangos están los datos de cada columna?, valores únicos, min, max

text:

Valores Únicos: La diversidad de las consultas es alta, a pesar de haber identificado algunos textos duplicados.

text\_length:

- Mínimo: Las consultas más cortas tienen 2 caracteres.
- Máximo: Las consultas más largas alcanzan los 136 caracteres (en train).
- Promedio (mean): La longitud promedio de las consultas está alrededor de 39.9 caracteres en train y 39.8 caracteres en val.
- Despersión (std): La desviación estándar de aproximadamente 15-16 caracteres indica una variabilidad moderada en las longitudes de las consultas, lo que significa que, si bien la mayoría se agrupan alrededor del promedio, también existen consultas significativamente más cortas o más largas.

intent:

Valores Únicos: Hay 150 intenciones únicas en cada uno de los subconjuntos (train, val, test), lo cual es consistente con la naturaleza del dataset CLINC150. Las intenciones son etiquetas categóricas (ej., translate, greeting, pay\_bill) y no tienen un rango numérico. La distribución de las clases es bastante balanceada, con cada intención apareciendo aproximadamente 100 veces ( $15000 / 150 = 100$ ).

- ¿Todos los datos están en su formato adecuado?

Si. Las consultas (text) son frases de lenguaje natural que no presentan caracteres inesperados o problemas de codificación (confirmado por el uso previo de chardet). Las intenciones (intent) son etiquetas de texto que corresponden a las 150 categorías predefinidas. Además, la columna derivada text\_length, que representa la longitud de cada consulta, está correctamente tipificada como int64 (entero).

- Los datos tienen diferentes unidades de medida?

Las columnas text e intent contienen datos cualitativos que no se expresan en unidades de medida físicas o estandarizadas (como metros, segundos, etc.) mientras que la columna derivada text\_length es una variable cuantitativa que sí tiene una unidad implícita: caracteres (o longitud de la cadena de texto). Los valores de esta columna varían desde un mínimo de 2 caracteres hasta un máximo de 136 caracteres, con una longitud promedio de aproximadamente 39.9 caracteres. Esta es la única "unidad" de medida numérica directa presente en el dataset CLINC150.

- Cuáles son los datos categóricos, ¿hay necesidad de convertirlos en numéricos?

No, porque la principal columna de datos categóricos en el dataset CLINC150 es intent y se utiliza principalmente para validar la calidad de los clústeres formados.

- ¿Qué representa un registro?

Cada registro representa una única consulta de usuario (text) junto con una etiqueta de intención específica (intent). Consta de 150 clases distintas, cada una correspondiente a una acción o necesidad expresada por el usuario, como traducir una palabra o configurar una alarma. Estas etiquetas permiten evaluar modelos de procesamiento de lenguaje natural en su capacidad para identificar correctamente la intención detrás de una frase dada, incluso cuando existen ambigüedades semánticas entre diferentes clases.

En cuanto a la granularidad, el dataset opera en un nivel fino y específico: cada fila es una unidad completa de expresión del usuario, etiquetada con una sola intención. No se manejan niveles más bajos como palabras individuales ni niveles más altos como conversaciones completas. Tampoco incluye información temporal ni geográfica, lo que refuerza su carácter atemporal y universal.

- ¿Están todas las filas completas o tenemos campos con valores nulos?

El análisis exploratorio de los subconjuntos train, val y test del dataset CLINC150 confirmó que todas las filas están completas y no contienen valores nulos. La verificación con df.info() y df.isnull().sum() mostró que tanto las columnas originales (text e intent) como las derivadas (text\_length) tienen valores no nulos en todos los registros, lo cual refleja una estructura de datos limpia y bien mantenida.

Dado que no hay valores faltantes, no es necesario aplicar estrategias de imputación, eliminación o combinación de datos, ni verificar comportamientos consistentes al agregar nuevas fuentes.

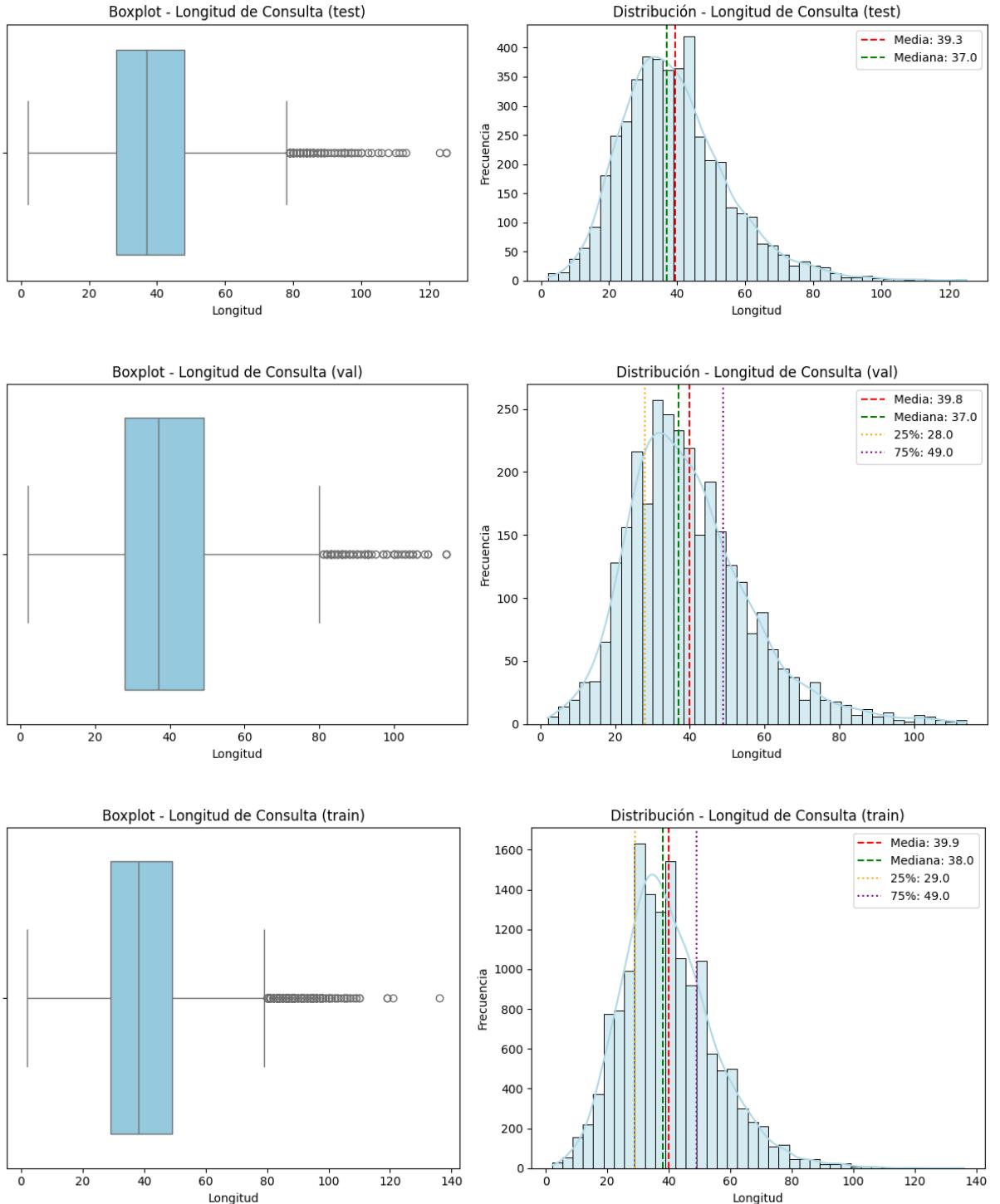
- ¿Siguen alguna distribución?

Usa describe() y analiza los valores.

En los subconjuntos de train, val y test se observa una distribución relativamente simétrica. La media y la mediana son muy cercanas (aproximadamente 39 caracteres), y los cuartiles reflejan esta centralidad. Aunque hay un rango considerable de longitudes (desde 2 hasta

136 caracteres), la distribución no presenta un sesgo extremo, sugiriendo una forma similar a una campana, pero sin ser una distribución normal perfecta.

Para la columna intent (intención), que es categórica, la distribución es altamente uniforme. Cada una de las 150 intenciones aparece un número consistentemente igual de veces en los subconjuntos de val y test (20 veces cada una), y de manera similar en train (aproximadamente 100 veces cada una). Esta uniformidad asegura una representación equitativa de todas las clases, lo cual es óptimo para tareas de clasificación y clustering.



- Usa medidas estadísticas:
- Medidas de tendencia central: media aritmética, geométrica, armónica, mediana, moda, desviación estándar.
- Correlación y covarianza: permite entender la relación entre dos variables aleatorias.

Se trabajó con la variable numérica `text_length`, que mide la longitud en caracteres de cada consulta. Sobre los subconjuntos `train`, `val` y `test` se calcularon medidas de tendencia central y dispersión, destacando en `train`: media ~57.3, mediana 56, moda 61, media geométrica ~55.9, media armónica ~54.1 y desviación estándar ~15.2. Estos valores reflejan una longitud típica con moderada variabilidad y sin distribución perfectamente simétrica.

Respecto a la relación entre longitud e intención, se codificó la variable categórica `intent` numéricamente para calcular correlación y covarianza con `text_length`. Ambas medidas resultaron cercanas a cero, indicando ausencia de relación lineal fuerte entre la intención y la longitud de la consulta. Esto sugiere que la semántica del mensaje no depende significativamente de su extensión.

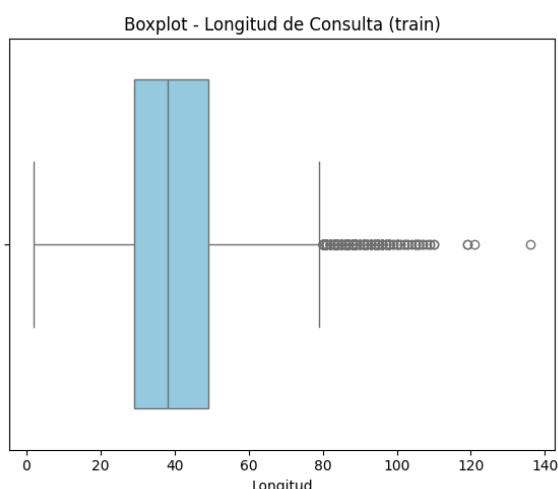
- ¿Hay correlación entre features (características)?

En el dataset CLINC150, las características principales son el texto de la consulta (`text`) y su intención (`intent`), siendo ambas variables categóricas o de texto. Al transformar el texto en su longitud (`text_length`) y codificar la intención numéricamente, se puede calcular correlación entre estas variables numéricas derivadas.

Sin embargo, los análisis muestran que la correlación entre la longitud de la consulta y la intención codificada es cercana a cero, lo que indica que no existe una relación lineal significativa entre estas características. Por lo tanto, en términos de correlación entre las features disponibles y derivadas, no se observa una dependencia relevante.

## Paso 2. Análisis de outliers

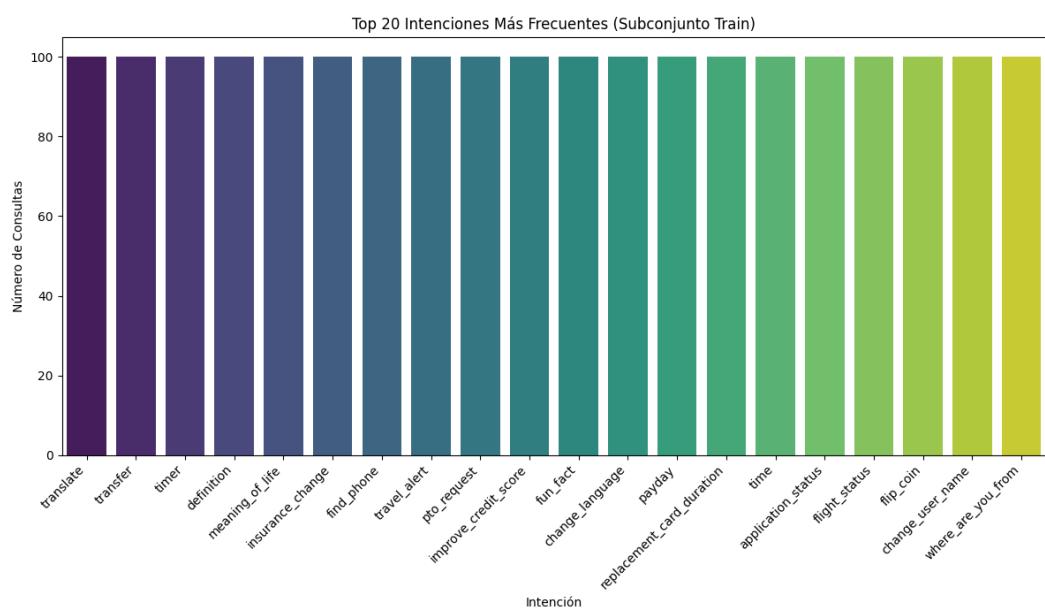
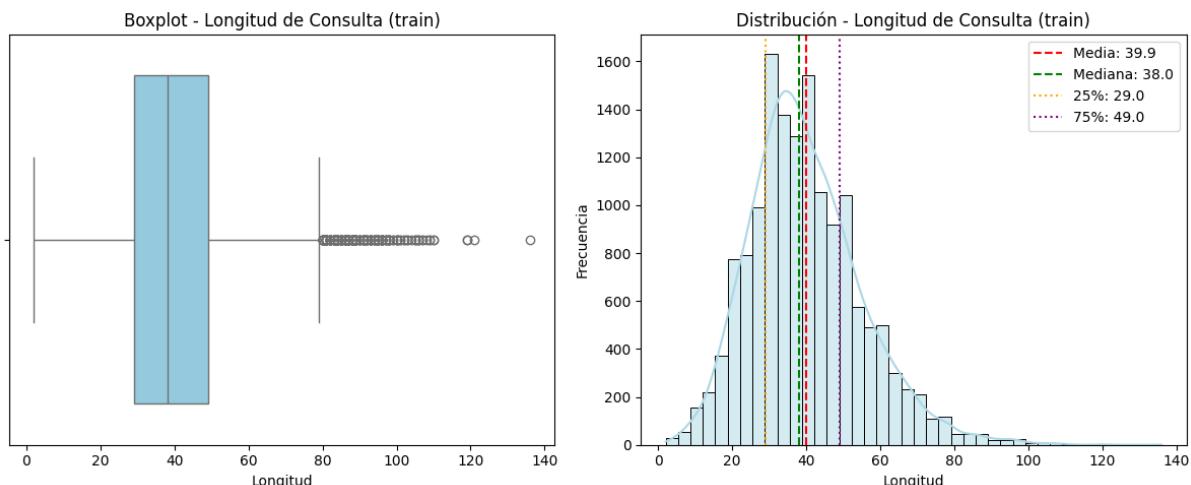
- ¿Cuáles son los Outliers? (unos pocos datos aislados que difieren drásticamente del resto y “contaminan” ó desvían las distribuciones) ¿Podemos eliminarlos? ¿Es importante conservarlos? son errores de carga o son reales?

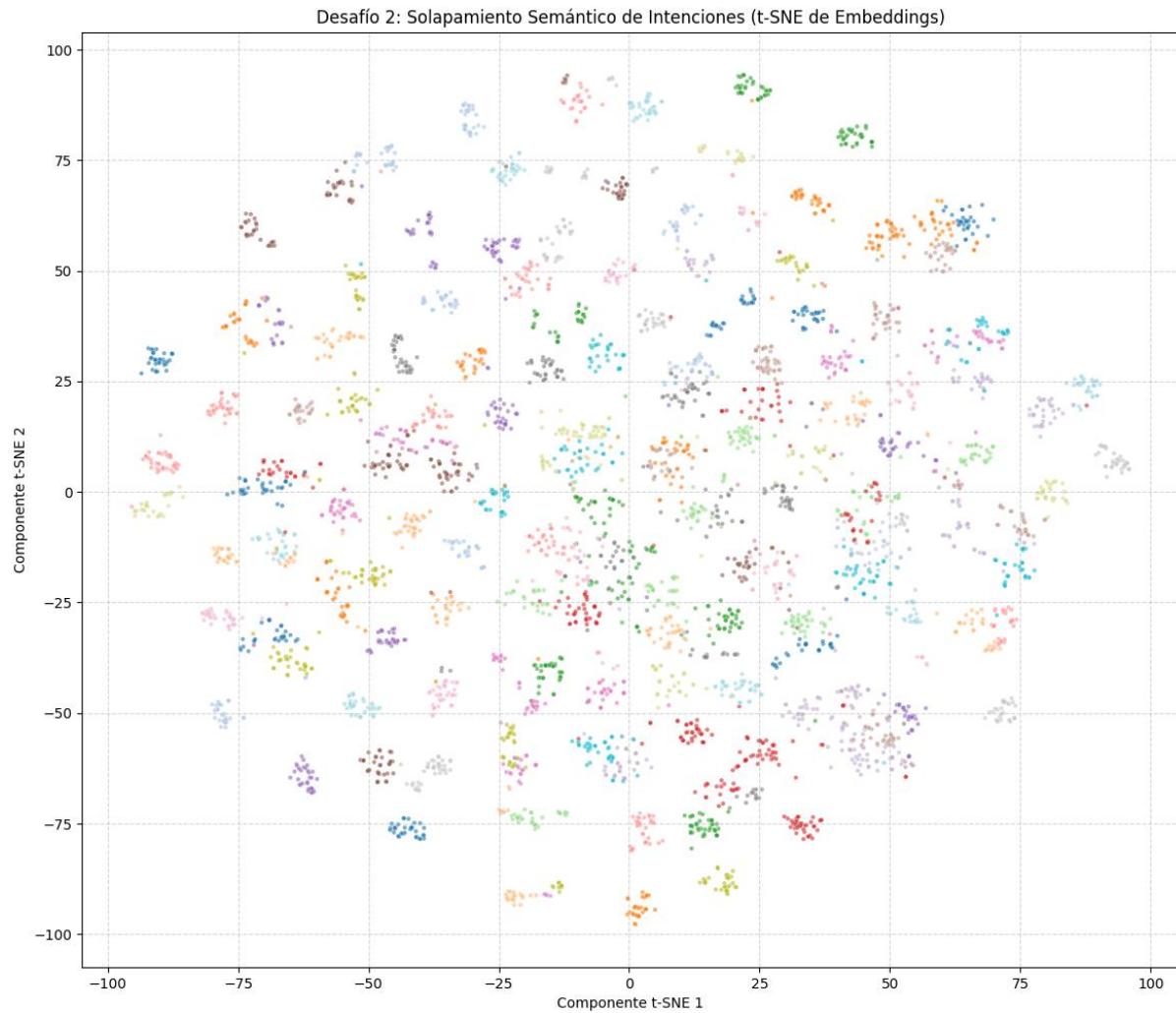


En el análisis del dataset CLINC150, se identificaron outliers en la columna `text_length`, correspondiente a la longitud de caracteres de las consultas. Utilizando el método del Rango Intercuartílico (IQR), se determinó que las consultas con más de 79 caracteres son consideradas outliers, ya que superan el límite superior definido por  $Q3 + 1.5 \times IQR$  (con  $Q1 = 29$ ,  $Q3 = 49$ ,  $IQR = 20$ ). El boxplot confirmó visualmente la existencia de estos valores extremos, con un máximo de hasta 136 caracteres en el conjunto de entrenamiento.

Aunque estos puntos pueden parecer “datos que contaminan” la distribución, no deben ser eliminados. A diferencia de errores de carga o registros corruptos, estos outliers son consultas reales y representativas del comportamiento natural de los usuarios. De hecho, las frases largas pueden reflejar intenciones más complejas o detalladas, como “Can you please tell me the exact amount of money I have remaining in my primary checking account after deducting all pending transactions?”, que transmite más contexto que una consulta breve como “check balance”.

### Paso 3: Visualización





#### **Paso 4. Encuentra un problema potencial en tus datos.**

Se identifico dos problemas significativos en el dataset CLINC150 que representan desafíos considerables para el agrupamiento no supervisado de intenciones de usuario:

#### **Alta Granularidad y Solapamiento Semántico.**

El dataset CLINC150 se distingue por su alta granularidad de intenciones, al contar con 150 categorías distintas. Un problema central surge de que muchas de estas categorías son semánticamente muy próximas entre sí. Si bien los modelos de lenguaje grandes (LLMs) son capaces de generar embeddings robustos que codifican el significado contextual del texto, estas representaciones a menudo no logran discriminar de manera óptima entre intenciones tan sutilmente diferenciadas. Como resultado, las visualizaciones de embeddings revelan un solapamiento considerable entre clases relacionadas. Este solapamiento dificulta que un algoritmo de clustering identifique límites claros entre las intenciones, llevando a agrupamientos confusos o inexactos.

#### **Ambigüedad Semántica.**

También se observa casos donde el dataset presenta ambigüedad en el significado de algunas frases. Hemos encontrado consultas idénticas o muy similares que están etiquetadas con

intenciones distintas. Este fenómeno es una característica inherente del lenguaje humano, donde una misma frase puede tener diferentes significados dependiendo del contexto. Para un modelo automático, esta ambigüedad representa un reto considerable, ya que sin información contextual adicional, distinguir entre estas intenciones se vuelve extremadamente difícil. Esta inconsistencia en el etiquetado para frases similares podría llevar a un aprendizaje erróneo o a agrupamientos incorrectos.

- Si es un problema de tipo supervisado: ¿Cuál es la columna de “salida”? ¿binaria, multiclase? ¿Está balanceado el conjunto salida?

Sí, el problema abordado con el dataset CLINC150 es de tipo supervisado. Esto se debe a que, para cada instancia de texto (consulta del usuario), se dispone de una etiqueta de intención (intent) predefinida. El objetivo es entrenar un modelo que aprenda a mapear la entrada de texto a su correspondiente intención.

Columna de “Salida”: La columna de salida es intent.

Tipo de Problema de Clasificación: Es un problema de clasificación multiclase. El dataset contiene 150 categorías de intención distintas, lo que requiere que un modelo sea capaz de predecir una de estas múltiples clases como salida. El conjunto de salida (intent) está altamente balanceado. Como se verificó en el análisis exploratorio de datos (EDA), cada una de las 150 intenciones tiene una representación equitativa en los subconjuntos de entrenamiento, validación y prueba (aproximadamente 100 ejemplos por intención en el conjunto de entrenamiento). Este balance es crucial para evitar sesgos del modelo hacia clases mayoritarias y para una evaluación justa del rendimiento en todas las intenciones.

- ¿Cuáles parecen ser features importantes? ¿Cuáles podemos descartar? ¿Estamos ante un problema dependiente del tiempo? Es decir, un TimeSeries.

#### **Features Importantes:**

- ✓ text (Consultas de Texto): Es la característica de entrada fundamental. El texto crudo es la base a partir de la cual los Large Language Models (LLMs) extraerán el significado semántico.
- ✓ embeddings (Representaciones Vectoriales del Texto): Son las características derivadas más importantes. Generados por LLMs (como all-MiniLM-L6-v2), estos vectores de alta dimensión capturan el significado semántico profundo de cada consulta y son la base numérica para el clustering y las visualizaciones.
- ✓ intent (Etiquetas de Intención Originales): Aunque es la variable objetivo, su información es vital para la evaluación cualitativa del clustering (comparar los clusters generados con las intenciones reales) y para guiar las funcionalidades de depuración y feedback humano.

#### **Features a Descartar:**

- ✓ text\_length (Longitud de la Consulta): Si bien es útil para el análisis exploratorio del dataset, se ha demostrado que tiene una correlación insignificante (-0.0507) con la intent codificada numéricamente. Esto implica que la longitud de una consulta no es una característica predictiva directa o relevante para determinar su intención semántica. Por lo tanto, no se utilizaría como feature de entrada para el clustering o para el LLM en esta tarea.

- Si fuera un problema de Visión Artificial: ¿Tenemos suficientes muestras de cada clase y variedad, para poder hacer generalizar un modelo de Machine Learning?

Trasladando esta pregunta al dominio de Procesamiento de Lenguaje Natural (PLN), sí, el dataset CLINC150 está diseñado con un número de muestras (15,000 en train, 3,000 en val, 4,500 en test) que se considera suficiente para entrenar y evaluar modelos de PLN para 150 clases. La presencia de 100 ejemplos por intención en el conjunto de entrenamiento es un número razonable para que un LLM aprenda a diferenciar las características semánticas de cada intención, lo que permite la generalización. La variedad de formulaciones dentro de las consultas (incluso si son para la misma intención) también contribuye a la capacidad de generalización.

- La distribución, tendencia de las variables varía en el tiempo?

Como se mencionó, este dataset no es temporal. La distribución y las tendencias observadas para variables como la longitud de las consultas (`text_length`) o el balance de clases (`intent`) son estáticas dentro del dataset. No hay un eje temporal para observar variaciones en estas distribuciones.

- ¿Hay algún problema notable con la calidad de los datos?

No hay problemas "notables" de baja calidad de datos en el sentido tradicional (ej. nulos, formatos inconsistentes, errores de carga masivos). CLINC150 es un dataset de investigación bien curado. Sin embargo, se han identificado desafíos inherentes a la complejidad del lenguaje natural y la granularidad de la tarea de comprensión de intenciones:

- Ambigüedad Semántica: La presencia de consultas de texto idénticas con diferentes etiquetas de intención. Este es un "problema" en el sentido de que introduce una ambigüedad que los sistemas automatizados deben aprender a manejar.
- Granularidad Fina de Intenciones: Las 150 intenciones, muchas de las cuales son semánticamente muy cercanas, generan solapamiento en el espacio de embeddings. Esto no es un problema de calidad del dato, sino una característica del dominio que dificulta la separación limpia de clases por algoritmos ciegos.

- ¿Existe alguna relación sorprendente entre las variables?

La ausencia de una correlación lineal significativa entre la longitud de la consulta (`text_length`) y la intención codificada numéricamente (-0.0507). Esto indica que la forma superficial del texto (su longitud) no es un indicador de su significado o intención subyacente. Esta "falta de relación" es, en sí misma, una relación importante que subraya la necesidad de enfoques basados en la semántica (como los LLMs) para comprender la intención. Otra relación clave, evidente en los gráficos t-SNE, es la fuerte agrupación semántica de consultas de la misma intención en el espacio de embeddings, lo que valida la capacidad de los LLMs para capturar el significado. Al mismo tiempo, la superposición visual entre intenciones semánticamente cercanas es una relación compleja y desafiante que justifica la necesidad de una herramienta de analítica visual como TextLens para su interpretación y refinamiento.

# Conclusión

¿Qué podemos aprender de este análisis?

## 1. Fiabilidad del Dataset como Base de Trabajo

El dataset CLINC150 ha demostrado ser una base de datos de alta calidad. No presenta valores nulos, mantiene un formato consistente en todos los ejemplos y, lo más importante, tiene un número equilibrado de ejemplos por cada una de las 150 clases de intención. Esto hace que podamos confiar en los datos para entrenar y evaluar modelos sin preocuparnos por errores básicos o sesgos de distribución. Gracias a esto, podemos concentrarnos directamente en los retos más complejos como la ambigüedad del lenguaje y la separación entre intenciones.

## 2. La Complejidad del Lenguaje Natural como Principal Desafío

- Ambigüedad en las consultas: Durante el análisis encontramos casos donde dos frases iguales tenían intenciones distintas. Esto muestra que el lenguaje humano es muy flexible y que entender la verdadera intención del usuario no siempre es fácil, ni siquiera para un modelo avanzado.
- Muchas clases con significados parecidos: Tener 150 intenciones tan específicas hace que algunas se parezcan mucho entre sí. Por ejemplo, intenciones como "check\_balance" y "account\_balance" pueden confundirse fácilmente. Al visualizar los embeddings con técnicas como t-SNE, vimos que hay bastante superposición entre clases. Esto hace que los clusters no estén claramente separados, lo que complica la tarea de clasificarlos automáticamente.

## 3. Las características simples no explican bien el significado

- También analizamos si la longitud del texto tenía relación con la intención, y descubrimos que no. Es decir, que una consulta más larga o más corta no predice con precisión su intención. Esto nos confirma que no podemos usar características superficiales como esa para resolver el problema, y que necesitamos herramientas más potentes, como los embeddings de modelos de lenguaje (LLMs), que entienden mejor el significado profundo del texto.

Este análisis exploratorio también nos permitió ver que nuestras hipótesis iniciales tienen mucho sentido.

- Hipótesis 1: Para entender mejor esta complejidad, necesitamos herramientas de visualización que nos ayuden a ver cómo se agrupan (o no) las intenciones.
- Hipótesis 2: Cuando vimos los solapamientos en los embeddings, notamos que TextLens podría ser muy útil para detectar esos casos difíciles y ayudarnos a analizarlos mejor.
- Hipótesis 3: Como hay muchas clases que se parecen, es esperable que los modelos no puedan separarlas perfectamente. Esto lo vimos en los gráficos t-SNE.

En resumen, aunque CLINC150 es un dataset bien hecho y balanceado, los verdaderos retos están en el lenguaje mismo: cómo las personas expresan sus intenciones y lo parecidas que pueden ser entre sí.