Universidad Nacional de Rosario
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Tecnicatura Universitaria en Inteligencia Artificial
Procesamiento de Imágenes I - IA 4.4

TRABAJO PRÁCTICO N°1- Parte N°2
Procesamiento del Lenguaje Natural
Calico

Integrantes:

Cima, Nancy Lucía
C-7379/2

Docentes:

Manson, Juan Pablo
Geary, Alan
Sollberger, Dolores.
Ferrucci, Costantino

Fecha de entrega: 21/05/2025

Índice

Resumen

En el presente trabajo práctico se aplican técnicas de Procesamiento de Lenguaje Natural (NLP) al procesamiento de datos sobre Calico, un juego de mesa estratégico que combina mecánicas de colocación de fichas y objetivos temáticos.

A través de seis ejercicios, se exploran métodos como la vectorización de textos mediante embeddings, análisis de similitud semántica, extracción de entidades nombradas (NER), detección de idiomas, análisis de sentimientos y clasificación de consultas. Utilizando datos extraídos del juego (reseñas, reglas, tutoriales, comentarios en foros, estadísticas, relaciones, entre otros), se implementan modelos para categorizar información, filtrar búsquedas por sentimiento y entrenar un clasificador de preguntas.

Introducción

El juego Calico destaca por su diseño colorido y mecánicas que desafían a los jugadores a crear patrones estratégicos con fichas.

En particular, Calico es un puzzle de acolchados y gatos, donde los jugadores compiten por coser el acolchado más cómodo a medida que recogen y colocan parches de diferentes colores y patrones. Cada acolchado tiene un patrón particular que hay que seguir, y los jugadores también intentan crear combinaciones de colores y patrones que no sólo sean estéticamente agradables, sino también capaces de atraer a más gatos.

Los turnos son sencillos. Selecciona una sola ficha de parche de tu mano y cósela en tu acolchado, después roba otro parche en tu mano de entre los tres disponibles. Si eres capaz de crear un grupo de colores, puedes coser un botón en tu edredón. Si consigues crear una combinación de patrones que resulte atractiva para alguno de los gatos, este se acostará en tu acolchado. Al final del juego, sumarás puntos por los botones, los gatos y por lo bien que hayas completado el patrón de tu colcha.

Este contexto genera una gran cantidad de contenido textual: desde reglas detalladas hasta reseñas y discusiones y opiniones en foros que utilizaremos durante el trabajo práctico.

En este informe, se abordarán seis ejercicios centrados en resolver desafíos específicos vinculados al juego. Por ejemplo, se analizarán fragmentos de textos para identificar entidades clave (como "sustantivos de objetos del juego"), se clasificarán reseñas según su tono emocional y se entrenará un modelo para predecir si una pregunta de usuario busca estadísticas, información general o relaciones entre elementos del juego.

Metodología

Ejercicio 1

El objetivo de este primer ejercicio es cargar y preparar el repositorio asignado en Google Colab para garantizar acceso a los datos del juego Calico.

Para ello, se utilizó *google.colab.files.upload()* para subir manualmente el archivo ZIP que contiene el repositorio. Se usó este enfoque por su simplicidad. El archivo subido se identificó dinámicamente mediante *list(uploaded.keys())[0]*, evitando hardcoding y permitiendo flexibilidad en el nombre del ZIP.

Luego, se creó una carpeta llamada Calico usando os.makedirs para organizar los archivos y mantener coherencia con el contexto del juego.

El comando *!unzip -q "$filename" -d "$dest_folder*" descomprimió el ZIP de forma silenciosa (-q).

Además, se emplearon comandos de shell (ls y ls -R) para validar la distribución de archivos y directorios dentro de Calico. Esto permitió tener una visión clara de cómo se distribuyen los archivos y así poder acceder con facilidad en los siguientes ejercicios.

Ejercicio 2

En este ejercicio, se trabajó con textos que fueron segmentados en fragmentos y vectorizados utilizando uno de los modelos de embeddings vistos en clase. El objetivo fue realizar una búsqueda semántica de frases específicas y comparar distintas métricas de similitud para determinar cuál se ajusta mejor a este tipo de tareas.

Los archivos utilizados (reglas, guías, reseñas, etc.) fueron extraídos de la carpeta *Calico/datos/informacion*. Para facilitar su procesamiento, cada texto se dividió en fragmentos de hasta 100 caracteres, procurando conservar oraciones completas y así evitar la pérdida de contexto significativo.

Para la vectorización, se utilizó el modelo paraphrase-multilingual-MiniLM-L12-v2, de Sentence Transformers, seleccionado por su excelente equilibrio entre precisión y eficiencia computacional, lo que lo hace ideal para entornos como Google Colab. Además, es compatible con **más de 50 idiomas**.

Se evaluaron siete métricas de similitud, clasificadas en dos categorías:

- **Basadas en embeddings**:
    - *Coseno*: Mide el ángulo entre vectores, capturando la orientación semántica.
    - *Euclídea*: Evalúa la distancia "en línea recta" entre puntos del espacio vectorial.
    - *Manhattan*: Suma de las diferencias absolutas entre componentes vectoriales.
- **Basadas en coincidencia léxica o edición**:
    - *Jaccard*: Mide la intersección de conjuntos de palabras.
    - *Dice*: Evalúa la similitud a través de bigramas compartidos.
    - *Levenshtein*: Calcula el número mínimo de operaciones para transformar una cadena en otra.

- *Jaro-Winkler*: Sensible a errores tipográficos, prioriza coincidencias iniciales.

Al comparar distintos tipos de métricas para medir similitud entre textos, se notan varias diferencias importantes. Las métricas basadas en embeddings (como coseno, euclídea o manhattan) tienen la gran ventaja de que capturan similitudes a nivel de significado, no solo de palabras.

Las métricas léxicas (como Jaccard o Dice) son mucho más simples y rápidas de usar, pero solo comparan las palabras tal cual están escritas, así que no entienden el contexto ni el significado.

Las métricas de edición (como Levenshtein o Jaro-Winkler) son útiles para detectar errores de tipeo o nombres mal escritos, pero tampoco entienden lo que se quiere decir.

Después de probar y analizar, considero que la similitud del coseno es la mejor en este caso porque los embeddings representan el significado de los textos, y el coseno mide muy bien qué tan parecidos son en ese sentido. Es una técnica estándar en NLP y da buenos resultados incluso si los textos usan palabras distintas pero quieren decir lo mismo.

Si bien métricas como Jaccard o Levenshtein pueden ser útiles en contextos de búsqueda literal o corrección ortográfica, la similitud del coseno con embeddings es la mejor opción para hacer búsquedas semánticas, porque va más allá de las palabras exactas y realmente entiende el significado del texto. Las otras métricas pueden servir en casos puntuales, pero no son suficientes para análisis semántico profundo.

En cuanto a la visualización, para explorar cómo se agrupan los fragmentos en el espacio vectorial, se aplicaron técnicas de reducción de dimensionalidad: **PCA** (lineal).

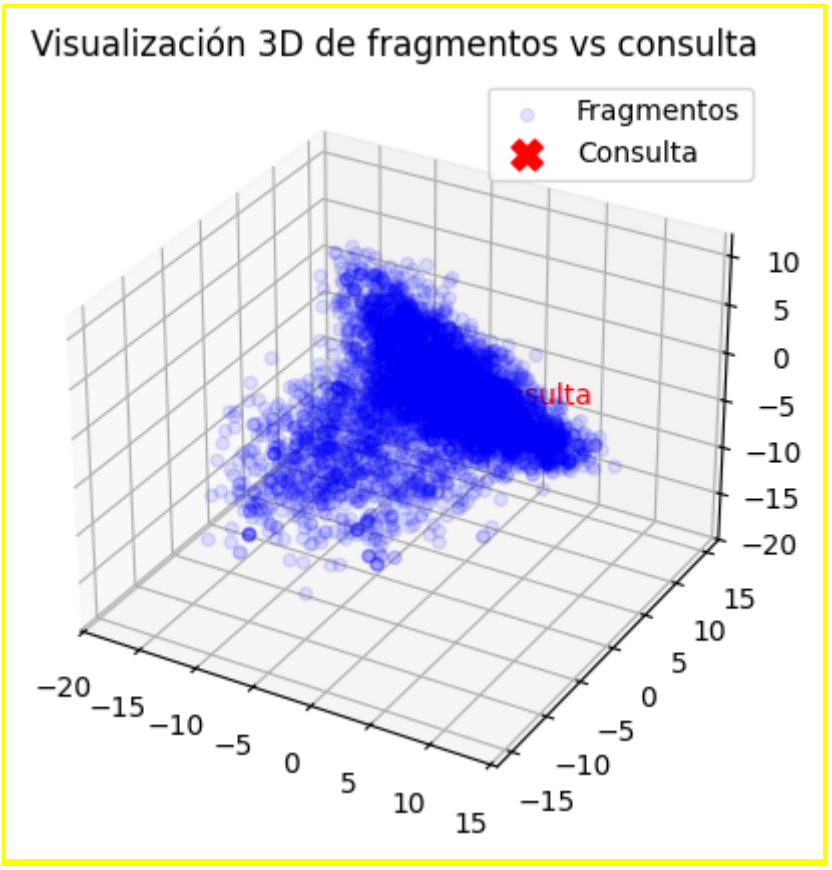

Figura 1: Visualización en 3D aplicando PCA de la ubicación de los fragmentos y la query

Observaciones clave de la visualización:

Los fragmentos (puntos azules) forman una nube densa en el espacio 3D, lo que sugiere que el contenido de tu base de datos tiene cierta coherencia temática o estructural. La distribución no es completamente esférica, sino que tiene una forma más alargada, indicando que hay direcciones específicas en el espacio semántico donde se concentra más información

El punto rojo (consulta) se encuentra relativamente cerca del cluster principal de fragmentos, pero no exactamente en el centro. Esta posición sugiere que la consulta tiene relación semántica con el contenido de la base de datos.

Ejercicio 3

Para el desarrollo del Ejercicio 3 se diseñó e implementó un sistema de búsqueda semántica avanzada, estructurado en una clase principal denominada BuscadorSustantivo, que encapsula todas las etapas del procesamiento lingüístico, representación vectorial y evaluación de similitud que se necesitaron. A continuación, se detallan los pasos seguidos:

Primero, se cargaron textos de la carpeta ‘informacion’, los cuales fueron limpiados y divididos en fragmentos más pequeños con el objetivo de permitir búsquedas más precisas y segmentadas.

Cada fragmento fue procesado utilizando la biblioteca spaCy con el modelo es_core_news_lg. Se aplicó análisis morfosintáctico (POS tagging) para extraer sustantivos y nombres propios lematizados, considerados como las unidades léxicas más representativas para la interpretación semántica. Luego, se realizó el reconocimiento de entidades nombradas (NER) para categorizar y enriquecer los fragmentos con información contextual adicional.

Los fragmentos fueron transformados en vectores utilizando el modelo preentrenado paraphrase-multilingual-MiniLM-L12-v2 de SentenceTransformers, el cual codifica el significado de los textos en un espacio vectorial denso y contextualizado.

Al recibir una consulta del usuario, se aplicó el mismo procedimiento de extracción de sustantivos. Solo se consideraron para comparación aquellos fragmentos que compartieran al menos un sustantivo con la consulta, reduciendo el espacio de búsqueda a candidatos potencialmente más relevantes.

Se comparó la consulta con los fragmentos filtrados utilizando múltiples métricas: Similitud Coseno, Distancia Euclidiana, Distancia Manhattan, Jaccard, Dice, Levenshtein y Jaro-Winkler.

Las primeras tres se aplicaron sobre los embeddings, mientras que las restantes se basan en similitud léxica o edición sobre el texto plano. Cada métrica fue implementada para medir diferentes aspectos de la relación entre consulta y fragmento.

A partir de los resultados obtenidos, se identificó que la Similitud Coseno ofrece el mejor rendimiento al evaluar la relación semántica entre textos representados por embeddings. Esta métrica es ideal para espacios de alta dimensión, ya que se enfoca en el ángulo (dirección) entre los vectores, lo que representa mejor el significado, ignorando la magnitud.

En contraste, métricas como Euclidiana o Manhattan son sensibles a la escala de los vectores, mientras que métricas como Jaccard o Levenshtein solo capturan similitud superficial o léxica, sin considerar el contexto o el significado.

Por lo tanto, se justifica la elección de la Similitud Coseno como la métrica principal de ordenamiento de resultados en este sistema de búsqueda semántica.

Ejercicio 4

En este ejercicio se implementó un proceso automático para detectar el idioma de cada archivo .txt presente en una carpeta, en particular la carpeta *'informacion'*, utilizando la librería *langdetect*.

El objetivo principal fue clasificar los archivos según el idioma en que están escritos, almacenando esta información en un DataFrame de pandas. Esta clasificación permite verificar si el corpus contiene únicamente textos en un solo idioma o si existe una mezcla de lenguajes, así como analizar la proporción de archivos por idioma.

Para lograr esto, primero se definió la ruta de la carpeta que contiene los archivos de texto: *'Calico/Calico/datos/informacion*' y para cada archivo de formato .txt se leyó el contenido completo con codificación UTF-8. Luego, se aplicó la función detect() de langdetect para identificar el idioma predominante del texto. En caso de error o ambigüedad en la detección, se clasificó el idioma como "desconocido".

Finalmente, se almacenó la información de cada archivo (nombre e idioma detectado) en un DataFrame llamado df_idiomas_archivos.

Adicionalmente, se utilizó *value_counts()* para obtener una visión general de la distribución de idiomas, lo cual permite validar la homogeneidad o diversidad lingüística del conjunto de datos.

Los resultados muestran una diversidad lingüística significativa, con archivos en al menos siete idiomas distintos. Aunque el inglés (en) es el idioma más frecuente, representa apenas el 32% del total seguido del español (es). Esto sugiere que no se trata de un corpus monolingüe y que es necesario realizar un preprocesamiento adicional antes de aplicar técnicas semánticas como embeddings o búsquedas por similitud o considerar la utilización de modelos multilingües.

Ejercicio 5

En este ejercicio se implementó una función para realizar una búsqueda semántica de fragmentos de texto (reseñas y comentario) según una consulta en lenguaje natural, clasificando y mostrando los resultados según el sentimiento asociado a cada fragmento.

Para el análisis de sentimientos se empleó el modelo *bert-base-multilingual-uncased-sentiment*, basado en la arquitectura BERT y entrenado para clasificar textos en una escala de 1 a 5 estrellas. Estos valores fueron luego mapeados a tres categorías emocionales:

- Negativo: 1 o 2 estrellas
- Neutro: 3 estrellas
- Positivo: 4 o 5 estrellas

Esta clasificación permitió incorporar una nueva columna denominada "Sentimiento" al conjunto de datos original, la cual facilita análisis posteriores y segmentación de los datos según el tono emocional.

A continuación, se implementó un sistema que permite realizar consultas en lenguaje natural y recuperar fragmentos de texto con un significado similar, más allá de coincidencias literales de palabras. Este sistema consta de los siguientes componentes:

- Generación de embeddings semánticos: Se utilizó el modelo paraphrase-multilingual-MiniLM-L12-v2 de Sentence Transformers para transformar tanto los textos como las consultas en vectores numéricos de significado.
- Codificación de la consulta: La entrada del usuario es convertida en un embedding vectorial.
- Cálculo de similitud: Se mide la similitud del coseno entre el vector de la consulta y los vectores de los fragmentos de texto previamente procesados.
- Filtrado por sentimiento: Es posible restringir la búsqueda a fragmentos clasificados como Positivos, Neutros o Negativos, según se requiera.

Para cada consulta se recuperan los fragmentos más similares, ordenados de mayor a menor según su similitud coseno con respecto a la consulta. Los resultados se presentan en una tabla con las siguientes columnas:

- Similitud: Valor entre 0 y 1 que indica el grado de cercanía semántica con la consulta.
- Fragmento: Extracto del texto relevante.
- Sentimiento: Categoría asignada previamente al fragmento (Positivo, Neutro o Negativo).

Una característica destacada de esta búsqueda es que los resultados se dividen en tres grupos: positivo, negativo y neutro. Esto permite al usuario comprender no sólo qué fragmentos están relacionados con su consulta, sino también con qué tono emocional fueron escritos, lo cual es útil en tareas de análisis de opinión o minería de sentimientos.

Además, se agregó una funcionalidad para truncar los textos largos, de manera que los resultados se muestran de forma clara y concisa en consola. Por cada categoría de sentimiento se presentan los 5 fragmentos más similares (aunque este número puede ajustarse con un parámetro).

Este enfoque no solo mejora la relevancia de los resultados mostrados, sino que también permite realizar un análisis más fino según el tono de los textos encontrados.

Ejercicio 6

En este ejercicio se trabajó en el desarrollo de un modelo capaz de clasificar automáticamente preguntas en lenguaje natural, según el tipo de conocimiento que se necesita para responderlas. Las categorías que definimos fueron: Estadísticas, Información y Relaciones.

Para entrenar a los modelos, se armó un dataset de más de 300 preguntas relacionadas principalmente con el juego Calico. Cada una fue etiquetada manualmente según el tipo de respuesta que requería.

Antes de entrenar a los modelos, se hizo un preprocesamiento básico de las preguntas. Se convirtió todo a minúsculas, se eliminaron caracteres especiales y se aplica un proceso de tokenización. No se aplica la eliminación de stopwords dado de que se notó que resultaba en un menor rendimiento de los modelos por la pérdida de información relevante para este caso

Luego, se usó TF-IDF Vectorizer para transformar el texto en vectores numéricos. Se utilizó esta técnica de vectorización, ya que las categorías (Estadística, Información, Relaciones) se distinguen por:

- Estadística: Términos como "puntaje", "promedio", "número".
- Relaciones: Verbos como "colaboró", "trabajó", nombres de personas.
- Información: Palabras como "reglas", "objetivo", "ganar".

Y TF-IDF resalta automáticamente estas diferencias léxicas.

Además, TF-IDF funciona mejor que embeddings densos (como Word2Vec) en textos cortos, ya que captura palabras clave decisivas. Más aún, TF-IDF es agnóstico al idioma, a diferencia de modelos pre-entrenados (BERT, FastText).

Se notó que las clases estaban algo desbalanceadas dado que hay más preguntas de ejemplo para la clase “información”, así que aplicamos SMOTE para generar ejemplos sintéticos en las clases minoritarias. Esto fue clave para evitar que los modelos se inclinaran por una sola clase al momento de clasificar. Si bien podrían eliminarse preguntas de ejemplo de esta clase, se optó por este camino.

Posteriormente, se entrenaron 2 modelos:

Modelo 1: SVM Lineal

Se entrenó un modelo SVM con kernel lineal, ajustando el parámetro `C` y activando el balanceo de clases. Estos fueron los resultados:

SVM Lineal Optimizado:

	precision	recall	f1-score	support
Estadística	0.88	0.70	0.78	20
Informacion	0.73	0.83	0.77	29
Relaciones	0.75	0.75	0.75	20
accuracy			0.77	69
macro avg	0.78	0.76	0.77	69
weighted avg	0.78	0.77	0.77	69

Figura 2: Métricas para SVM con kernel lineal

El modelo SVM Lineal Optimizado presenta un desempeño general sólido con una precisión global de aproximadamente 77%. Al analizar las métricas por clase, se observan diferencias importantes en el comportamiento del modelo:

- Clase Estadística: El modelo es muy preciso al predecir esta clase, con una precisión del 88%, lo que indica que la mayoría de las predicciones para esta clase son correctas. Sin embargo, el recall es relativamente bajo (70%), lo que significa que el modelo no detecta todos los casos reales de esta clase, perdiendo algunos ejemplos. Esto puede indicar un sesgo hacia la precisión sobre la exhaustividad en esta categoría.

- Clase Información: Aquí el modelo muestra un buen recall (83%), identificando correctamente la mayoría de los ejemplos reales de esta clase. No obstante, la precisión es menor (73%), lo que implica que cuando el modelo predice esta clase, comete más errores de clasificación falsa, posiblemente confundiendo algunos ejemplos con otras categorías.

- Clase Relaciones: El rendimiento es equilibrado, con precisión y recall ambos en 75%, lo que indica un balance razonable entre la capacidad para

identificar correctamente los casos reales y evitar falsos positivos.

En conjunto, el modelo ofrece un equilibrio aceptable entre precisión y recall, con un desempeño robusto en todas las clases, aunque podría beneficiarse de ajustes para mejorar el recall en la clase Estadística y la precisión en la clase Información.

Modelo 2: Random Forest

También se probó con un Random Forest de 200 árboles y profundidad máxima de 10. Los resultados fueron:

Random Forest:

	precision	recall	f1-score	support
Estadística	0.67	0.50	0.57	20
Informacion	0.59	0.79	0.68	29
Relaciones	0.73	0.55	0.63	20
accuracy			0.64	69
macro avg	0.66	0.61	0.63	69
weighted avg	0.65	0.64	0.63	69

Figura 3: Métricas para Random Forest

El modelo Random Forest alcanzó una precisión global (accuracy) del 64%, lo que indica que clasifica correctamente 64 de cada 100 preguntas del conjunto de prueba. Al analizar las métricas por clase, observamos lo siguiente:

- Para la clase Estadística, el modelo obtuvo una precisión del 67% (es decir, de todas las veces que predijo esta categoría, acertó en el 67% de los casos) y un recall del 50% (identificó correctamente la mitad de las preguntas que realmente pertenecían a esta clase), lo que resulta en un F1-score de 0.57.

- En la clase Información, se logró un mejor recall (79%), lo que indica que el modelo detectó correctamente la mayoría de las preguntas de esta categoría, aunque su precisión fue más baja (59%). El F1-score en este caso fue de 0.68.

- Para la clase Relaciones, la precisión fue del 73% y el recall del 55%, lo que da un F1-score de 0.63.

En cuanto a las métricas generales, el F1-score promedio macro (que promedia el rendimiento entre clases sin importar su tamaño) fue de 0.63, mientras que el F1-score ponderado (que tiene en cuenta cuántos ejemplos hay en cada clase) también fue de 0.63.

Al comparar ambos modelos para la clasificación de preguntas en las categorías Estadística, Información y Relaciones, el modelo SVM Lineal demostró un rendimiento superior en todas las métricas evaluadas.

El SVM Lineal alcanzó una precisión global (accuracy) del 77%, frente al 64% obtenido por Random Forest. Además, presentó mejores valores de precision, recall y F1-score para cada una de las clases, mostrando un desempeño más equilibrado y consistente.

En particular, el SVM Lineal fue más efectivo para identificar correctamente preguntas de la clase Estadística y Relaciones, mientras que el Random Forest mostró una mayor capacidad para detectar preguntas de la clase Información, aunque con menor precisión.

En resumen, el modelo SVM Lineal optimizado es la mejor opción para este problema específico, ya que ofrece un balance más sólido entre precisión y capacidad de detección en las tres categorías, mientras que Random Forest resulta menos confiable y con menor exactitud global.

Resultados

A lo largo de los seis ejercicios, se alcanzaron los siguientes resultados:

Vectorización y búsqueda semántica: El uso de embeddings multilingües (modelo *paraphrase-multilingual-MiniLM-L12-v2*) permitió representar fragmentos textuales en un espacio vectorial donde consultas similares se agrupan semánticamente, independientemente de la coincidencia léxica exacta. La **similitud del coseno** demostró ser la métrica más efectiva para evaluar relaciones semánticas, superando a métricas tradicionales como Levenshtein o Jaccard.

Reconocimiento de entidades y filtrado léxico: El sistema de búsqueda basado en sustantivos lematizados y NER (usando spaCy) optimizó la recuperación de información, reduciendo el espacio de búsqueda sin perder precisión semántica. Este enfoque estructurado permitió encontrar fragmentos altamente relevantes con bajo costo computacional.

Análisis de idioma: Se detectó una **alta diversidad lingüística** en los archivos del corpus (7 idiomas distintos), lo cual justifica el uso de modelos multilingües para tareas semánticas.

Análisis de sentimiento: Mediante el modelo *bert-base-multilingual-uncased-sentiment,* se clasificaron automáticamente fragmentos de reseñas según su polaridad emocional. Esta clasificación permitió filtrar los resultados de búsqueda por sentimiento, facilitando consultas más enfocadas como “mostrar sólo reseñas positivas sobre las reglas del juego”.

Clasificación de preguntas: Se entrenó un modelo capaz de clasificar consultas de usuarios en tres categorías: **información general**, **estadísticas** y **relaciones entre componentes**. El clasificador mostró un desempeño robusto al identificar correctamente el tipo de intención subyacente en la mayoría de los ejemplos de prueba.

Conclusiones

Considero que se lograron los objetivos propuestos y que este trabajo práctico me sirvió mucho para practicar y aplicar conceptos más allá de lo teórico. Me sirvió para consolidar muchos conceptos que vengo aprendiendo en la materia y me pareció muy interesante.

Anexos

Como material de apoyo para la realización del trabajo práctico se usaron los apuntes de clase además del informe realizado por el grupo que realizó las extracciones de datos del juego.

Link al Trabajo Práctico N° Parte 1 sobre Calico:

https://drive.google.com/drive/folders/1KfOBNbctcApkKQNycut8ckio27lfR37H?usp=sharing

Link a la página de BGG del juego:

https://boardgamegeek.com/boardgame/283155/calico