UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Diseño de sistema de profiling para ecosistema de ciencia ciudadana

AUTORES: Elías Biagioni

DIRECTOR/A: Diego Torres

CODIRECTOR/A: -

ASESOR/A PROFESIONAL: -

CARRERA: Licenciatura en Sistemas

Resumen

El proyecto de ciencia ciudadana "Ágora", cuenta con proyectos, en los cuales hay variedad de temáticas y de usuarios que participan, con diferentes orientaciones e intereses. Debido a la gran diversidad de proyectos y usuarios, se hace difícil recomendar proyectos a personas, así como personas a proyectos. La idea de esta tesina es mejorar el proyecto Ágora, desarrollando un sistema de recomendación, capaz de extraer las diferentes características de sus usuarios y proyectos, de manera tal que se puedan recomendar proyectos a usuarios para participar.

Palabras Clave

Ciencia ciudadana – Cientopolis – Ágora – Intereses de participantes – Intereses de lideres – PPSR-Core – Profiling – Sistema de recomendación – Sistema de recomendación personalizado – Sistema de recomendación basado en contenido – Content Analyzer – Profile Learner – Filtering Component – Java – Neo4j – Similitud Jaccard – Normalizacion minimo-maximo – Escalar por vector – Producto de matrices – KNN – Precision – Recall – F1-Score

Conclusiones

Esta investigación se distingue en el ámbito de los sistemas de recomendación, es su enfoque integral en la ciencia ciudadana. Se ha profundizado en cómo las recomendaciones pueden tener un impacto significativo en la participación y compromiso ciudadano en proyectos de su interés, enriqueciendo así la contribución colectiva al conocimiento y la investigación.

Trabajos Realizados

Se ha llevado a cabo el análisis, diseño e implementación de un sistema de recomendación basado en contenido, con el propósito de mejorar y potenciar la interacción y compromiso de los usuarios en proyectos de ciencia ciudadana. En lugar de optar por un modelo basado en Machine Learning, se optó por Neo4j por su facilidad de adaptación. Se desarrolló el algoritmo utilizando consultas en Cypher. Se evaluó el rendimiento del sistema mediante las métricas de Precision, Recall y F1-Score

Trabajos Futuros

Optimización – Inclusión de la Ubicación Geográfica – Recomendación de Usuarios a Proyectos – Sistema de Recomendación Colaborativo – Integración con Ágora

Fecha de la presentación: **Febrero 2024**

UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Diseño de sistema de profiling para ecosistema de ciencia ciudadana

AUTORES: Elías Biagioni

DIRECTOR/A: Diego Torres

CODIRECTOR/A: -

ASESOR/A PROFESIONAL: -

CARRERA: Licenciatura en Sistemas

Resumen

El proyecto de ciencia ciudadana "Ágora", cuenta con proyectos, en los cuales hay variedad de temáticas y de usuarios que participan, con diferentes orientaciones e intereses. Debido a la gran diversidad de proyectos y usuarios, se hace difícil recomendar proyectos a personas, así como personas a proyectos. La idea de esta tesina es mejorar el proyecto Ágora, desarrollando un sistema de recomendación, capaz de extraer las diferentes características de sus usuarios y proyectos, de manera tal que se puedan recomendar proyectos a usuarios para participar.

Palabras Clave

Ciencia ciudadana – Cientopolis – Ágora – Intereses de participantes – Intereses de lideres – PPSR-Core – Profiling – Sistema de recomendación – Sistema de recomendación personalizado – Sistema de recomendación basado en contenido – Content Analyzer – Profile Learner – Filtering Component – Java – Neo4j – Similitud Jaccard – Normalizacion minimo-maximo – Escalar por vector – Producto de matrices – KNN – Precision – Recall – F1-Score

Conclusiones

Esta investigación se distingue en el ámbito de los sistemas de recomendación, es su enfoque integral en la ciencia ciudadana. Se ha profundizado en cómo las recomendaciones pueden tener un impacto significativo en la participación y compromiso ciudadano en proyectos de su interés, enriqueciendo así la contribución colectiva al conocimiento y la investigación.

Trabajos Realizados

Se ha llevado a cabo el análisis, diseño e implementación de un sistema de recomendación basado en contenido, con el propósito de mejorar y potenciar la interacción y compromiso de los usuarios en proyectos de ciencia ciudadana. En lugar de optar por un modelo basado en Machine Learning, se optó por Neo4j por su facilidad de adaptación. Se desarrolló el algoritmo utilizando consultas en Cypher. Se evaluó el rendimiento del sistema mediante las métricas de Precision, Recall y F1-Score

Trabajos Futuros

Optimización – Inclusión de la Ubicación Geográfica – Recomendación de Usuarios a Proyectos – Sistema de Recomendación Colaborativo – Integración con Ágora

Fecha de la presentación: **Febrero 2024**

Diseño de sistema de profiling para ecosistema de ciencia ciudadana

Elías Biagioni

Febrero 2024

Agradecimientos

A mi familia, por su apoyo incondicional en cada paso de este camino, por ser mi fuente de motivación y fuerza. A mis compañeros y amigos, quienes han sido un apoyo constante y una fuente de alegría. A mi director de tesina, Diego Torres, cuya guía, paciencia y conocimientos han sido fundamentales en cada etapa de este proyecto. A Javier Bazzocco, quien fue el enlace con Diego Torres y una pieza clave para dar inicio a este proyecto. Finalmente, agradezco a la Universidad Nacional de La Plata y Facultad de Informática por darme la oportunidad de formarme en un ambiente académico de excelencia, sentando las bases para mi futuro profesional. Todos han sido fundamentales en mi crecimiento profesional y personal.

Resumen

La ciencia ciudadana es un tipo de colaboración en la cual las actividades científicas son llevadas a cabo por una comunidad de colaboradores. En este ambiente, existe una amplia diversidad de proyectos de investigación, como así también, un amplio marco de ciudadanos interesados que participan en ellos. El proyecto de ciencia ciudadana “Ágora”, cuenta con proyectos, en los cuales hay variedad de temáticas y de usuarios que participan, con diferentes orientaciones e intereses. Debido a la gran diversidad de proyectos y usuarios, se hace difícil recomendar proyectos a personas, así como personas a proyectos. Esto es debido, en parte, a que los intereses de los usuarios van cambiando a lo largo del tiempo sobre la gran variedad de categorías de proyectos. Una manera de solucionar este problema, es utilizando técnicas de perfilamiento (profiling) tanto a nivel de usuarios cómo a nivel de proyectos. La idea de esta tesina es mejorar el proyecto Ágora, desarrollando un sistema de recomendación, capaz de realizar la extracción de las diferentes cualidades y características de sus usuarios y proyectos, contextualizando en ecosistemas de proyectos de ciencia ciudadana, de manera tal que se puedan recomendar usuarios a proyectos y viceversa.

Índice general

Capítulo 1

Introducción

1.1. Motivación

Según el libro *The Science of Citizen Science* [32], la ciencia ciudadana es una forma de colaboración abierta en la que ciudadanos participan en el proceso científico para abordar problemas del mundo real. Esto incluye actividades como la identificación de preguntas de investigación, la recolección y análisis de datos, la interpretación de resultados, la realización de nuevos descubrimientos, el desarrollo de tecnologías y aplicaciones, y la solución de problemas complejos. En este proceso, los ciudadanos se involucran directamente en proyectos científicos, contribuyendo con calidad y aprendiendo en el proceso, mientras que las instituciones científicas obtienen ayuda y divulgan su trabajo a la comunidad.

El centro de investigación LIFIA (Laboratorio de Investigación y Formación en Informática Avanzada) desarrolló un proyecto de ciencia ciudadana denominado ”Ágora”. Este proyecto busca conectar diversas temáticas y atraer a usuarios con distintas orientaciones e intereses. Está diseñado para permitir la creación de múltiples proyectos con una amplia variedad de temáticas, aunque actualmente no está en línea y, por lo tanto, aún no cuenta con proyectos existentes. En Ágora, cada proyecto creado tiene un líder, co-creadores (ayudan en la definición del proyecto), un objetivo específico y un protocolo, que es un formulario de pasos que deben completar aquellos ciudadanos, o participantes, que tienen interés en el tema del proyecto. Los participantes de los proyectos presentan características e intereses diversos, que van desde las temáticas específicas hasta la empatía con los organizadores.

Sin embargo, debido a la gran diversidad de proyectos y usuarios, resulta difícil recomendar proyectos a personas, así como personas a proyectos. Esta dificultad surge, en parte, porque los intereses de los usuarios varían con el tiempo en

respuesta a la variedad de categorías de proyectos disponibles. Para abordar este problema, se propone la utilización de técnicas de perfilamiento (Profiling) tanto para usuarios como para proyectos. Según Dong et al., 2020 [36], el perfilamiento de usuarios busca comprender las características y requisitos específicos de los individuos en diferentes escenarios. Esta técnica tiene como objetivo principal obtener información sobre los intereses de los usuarios y el intervalo de tiempo en el cual han mostrado dichos intereses. El propósito es mejorar la calidad del acceso a la información del usuario y determinar su intención. Para ello, se utilizan algoritmos de aprendizaje automático (Machine Learning) que son entrenados para tareas de recolección y preprocesamiento de datos, así como la extracción automatizada de características relevantes. Estos algoritmos aprovechan los contenidos generados por los usuarios, los cuales reflejan directamente los pensamientos e intenciones de los individuos, siendo una fuente valiosa para el perfilamiento. El perfilado permitirá obtener una gran variedad de información sobre los proyectos, los usuarios y su interés a través del tiempo.

El propósito de esta tesina es mejorar el proyecto Ágora mediante el desarrollo de un módulo capaz de realizar la extracción de distintas cualidades y características de sus usuarios y proyectos, en el contexto de ecosistemas de ciencia ciudadana. Dichos ecosistemas engloban proyectos con diversas temáticas, que se llevan a cabo en distintos lugares, implican una amplia variedad de tareas y son organizados y participados por diferentes grupos de personas.

1.1.1. Objetivos

El objetivo principal es diseñar un sistema para modelar perfiles en ecosistemas de ciencia ciudadana, basándose en información dinámica y a largo plazo, tanto de usuarios como de proyectos.

Los objetivos específicos son:

- Diseñar y modelar perfiles para proyectos de ciencia ciudadana.
- Diseñar y modelar perfiles para usuarios de proyectos de ciencia ciudadana.
- Diseñar un modelo de recomendación para sugerir usuarios a proyectos y viceversa.
- Desarrollar una herramienta que genere y muestre información sobre el perfil obtenido para cada usuario y proyecto.

1.2. Enfoque

El enfoque de esta tesina es el desarrollo e implementación de un sistema de recomendación para la aplicación Ágora. Se busca ofrecer recomendaciones precisas de proyectos a usuarios basadas en sus intereses a lo largo del tiempo. Para ello, se implementó un sistema de recomendación estructurado en varios componentes esenciales que evalúan, categorizan y recomiendan proyectos a los usuarios según sus interacciones. A nivel técnico, se integró una base de datos orientada a grafos para el desarrollo del algoritmo de recomendación, en combinación con la base de datos orientada a documentos ya existente en Ágora.

Además, se lleva a cabo una evaluación cuantitativa y cualitativa del sistema propuesto, considerando las métricas Precision, Recall y F1-Score.

Finalmente, el trabajo contempla una discusión sobre posibles mejoras y extensiones para el sistema, como la incorporación de geolocalización y la integración de un sistema de recomendación colaborativo.

1.3. Organización

La presente tesina está organizada en varios capítulos, cada uno con un enfoque y objetivo específico, para facilitar la comprensión del problema y su solución.

El primer capítulo, la **Introducción**, tiene como objetivo situar al lector en el contexto general del proyecto, delineando los problemas que se abordan, y los objetivos y metas específicos que guían el trabajo de investigación.

El segundo capítulo, **Relacionado**, ofrece una revisión literaria de los conceptos y tecnologías que son fundamentales para el entendimiento de los proyectos de ciencia ciudadana, los distintos beneficios para los líderes y participantes, los objetivos de la aplicación Ágora y la motivación para el desarrollo de un algoritmo de recomendación

El tercer capítulo, **Estrategia General**, proporciona un plano estratégico para el desarrollo del algoritmo de recomendación. Se presenta el rol y armado de los componentes clave del sistema de recomendación, explicando las razones detrás de las decisiones de diseño y cómo contribuyen a los objetivos del sistema.

En el cuarto capítulo, **Diseño e Implementación**, se profundizan los aspectos técnicos de la implementación del sistema.

El quinto capítulo, **Evaluación**, introduce un enfoque riguroso para evaluar la eficacia del algoritmo de recomendación, utilizando las métricas Precisión, Recall y F1-Score. Este es elemental para validar la calidad del sistema de recomendación,

ofreciendo un análisis detallado que respalda la efectividad de la solución propuesta.

El capítulo final, **Conclusiones y Trabajo Futuro**, proporciona una evaluación general y discute las implicancias para futuras investigaciones y desarrollos, para mejoras y/o agregados en el sistema de recomendación.

Capítulo 2

Participación y Liderazgo en Ciencia Ciudadana

En este capítulo, se desarrollará la motivación del desarrollo de la tesina con mayor profundidad, con el objetivo de comprender cuáles son los intereses y/o motivaciones de los usuarios al participar en un proyecto de ciencia ciudadana, así como los intereses u objetivos de los creadores de proyectos. La información detallada en la siguiente sección, fue extraída del libro *The Science of Citizen Science* [32].

2.1. Introducción

Como vimos en la sección 1.1 y según el libro *The Science of Citizen Science*, la ciencia ciudadana es una forma de colaboración abierta en la que ciudadanos participan en el proceso científico para abordar problemas del mundo real. En esta, se incluyen científicos, académicos y ciudadanos comunes sin ningún tipo de preparación científica previa. La característica que más distingue a la ciencia ciudadana de otras formas de ciencia, es que científicos no profesionales, participan en el proceso científico. Son los denominados ”ciudadanos” de la ciencia ciudadana. Estos pueden colaborar con los científicos en todas las etapas y aspectos del proceso científico, pero, en la mayoría de los proyectos, contribuyen en la recolección y análisis de datos. En este lazo, los ciudadanos se involucran directamente en proyectos científicos aportando su trabajo con la calidad necesaria para realizar ciencia, aprendiendo de los mismos y, en alguna medida, son formados en el método científico. El campo de la investigación sobre las experiencias de los participantes en la ciencia ciudadana es nuevo y toma prestado de muchos otros campos. Por ejemplo, aunque el uso del

término ”voluntario” para referirse a los participantes en la ciencia ciudadana puede ser problemático, gran parte de la investigación sobre los participantes se inspira en el campo de la investigación sobre el voluntariado en las ciencias sociales y la salud. Las motivaciones del voluntariado en general son similares a las de la participación en la ciencia ciudadana. La investigación y la teoría de otros campos como la educación, la psicología y las ciencias sociales también aplican al estudio de la participación en la ciencia ciudadana.

2.2. Motivaciones de los Participantes

Tomar en serio los conocimientos, habilidades y experiencia de los ciudadanos como colaboradores en la investigación científica se inscribe en una tendencia más amplia hacia la participación pública en varios ámbitos. En el campo de la ciencia ciudadana, se ha incrementado el escrutinio del rol y el punto de vista de los participantes porque, además de desarrollar y proporcionar procedimientos y protocolos claros para garantizar la calidad de los datos, asegurar que se satisfacen las expectativas y necesidades de los participantes también influye en la calidad de los resultados científicos. Dado que la mayoría de los proyectos de ciencia ciudadana tienen objetivos relacionados con los beneficios para los participantes, resulta necesario comprender sus experiencias para poder medir los resultados. Además, se considera esencial conocer mejor las experiencias de los participantes para poder hacer afirmaciones sobre los beneficios generales de la ciencia ciudadana para los participantes y la sociedad.

En general, se permite que los ciudadanos participen en diferentes niveles del proceso científico: desarrollo de preguntas e hipótesis de investigación, recopilación de datos, análisis de datos, elaboración de conclusiones y difusión de datos. La mayoría de los proyectos de ciencia ciudadana son proyectos de inteligencia contributiva o distribuida. En estos proyectos, se recluta a los participantes para que contribuyan a una determinada causa científica, y luego se inscriben en un proyecto que se ajusta a sus motivaciones e intereses y comienzan a contribuir según un protocolo fijo. Aunque muchos participantes pueden estar satisfechos con un nivel mínimo de compromiso en un proyecto de ciencia ciudadana, en general se considera beneficioso ofrecer oportunidades para que los participantes se involucren más en un proyecto de ciencia ciudadana, si así lo desean.

Según estudios realizados por Clary et al. (1998), Raddick et al. (2013), Wright et al. (2015), y Sullivan et al. (2009), las motivaciones para participar en proyectos de ciencia ciudadana varían según los individuos. Para atraer a los participantes y

mantenerlos comprometidos con un proyecto, se considera importante entender qué les impulsa a participar y por qué se quedan con un proyecto o lo abandonan. Los estudios sobre la motivación en la ciencia ciudadana suelen utilizar métodos de investigación en ciencias sociales, como encuestas y entrevistas. En las encuestas, los participantes suelen indicar hasta qué punto están de acuerdo con una lista de afirmaciones sobre su motivación o indican las motivaciones más importantes para ellos. A menudo, varias de estas preguntas se combinan dentro de categorías de motivación como la contribución, la motivación intrínseca o la motivación extrínseca. En muchos de estos estudios, los participantes están motivados por el hecho de que están contribuyendo a la ”ciencia real” o al objetivo general del proyecto. Otra motivación importante suele ser el interés intrínseco por el tema concreto del proyecto, como las aves, las galaxias, las plantas, el lenguaje, etc. Otras motivaciones comunes están relacionadas con el disfrute, el ocio y la interacción social; los participantes suelen buscar actividades agradables o una forma de formar parte de una comunidad de personas con ideas afines. Otros estudios han demostrado que las interacciones cara a cara con los científicos más destacados pueden influir positivamente en el nivel de participación en un proyecto. Otros proyectos ofrecen canales de comunicación en su sitio web, como un foro, u organizan eventos independientes en los que los participantes pueden reunirse entre sí y con los líderes del proyecto fuera de línea y compartir sus experiencias.

Según estudios realizados por Blaney et al. (2016), Haywood (2014), King et al. (2016), Moore et al. (2006), Townsend (2006), y Phillips et al. (2018), se identifican una serie de beneficios y resultados para los participantes en la ciencia ciudadana. Desde el punto de vista del participante, estos beneficios están relacionados, por ejemplo, con la alfabetización científica, los beneficios para la salud, la oportunidad de socializar y el empoderamiento. Además del impacto científico de la ciencia ciudadana, los proyectos pueden y deben tratar de garantizar los beneficios para los participantes que, a su vez, pueden ser impulsores de los productos y resultados, como el nivel de inversión personal y la voluntad de defender el programa. Los beneficios suelen estar relacionados con las motivaciones que tienen los participantes al contribuir a un proyecto. Por ejemplo, cuando alguien participa en un proyecto para aprender más sobre las mariposas, un beneficio esperado sería un mayor conocimiento y comprensión. Además, se considera que los líderes de proyectos pueden tener ciertos objetivos en mente con respecto a los resultados para los participantes, a menudo en términos de resultados de aprendizaje, mayor concientización sobre un tema y cambio de comportamiento. En una encuesta de seguimiento en línea de líderes de proyectos, se descubrió que alrededor de la mitad de los encuestados midieron los resultados de su proyecto. Los resultados más reportados fueron el interés

o el compromiso con la ciencia (46 %), el conocimiento (43 %), el cambio de comportamiento (36 %), el cambio de actitud (33 %) y las habilidades de investigación (28 %). Es interesante observar que existe una discrepancia entre los objetivos del proyecto más comúnmente declarados y los resultados medidos. Estudios realizados por Raddick et al. (2013) demuestran que muchos científicos ciudadanos comienzan a contribuir a un proyecto porque tienen un interés preexistente en el tema del proyecto. También pueden interesarse aún más por el tema una vez que se involucran más en él. La autoeficacia, la confianza en la capacidad para una determinada tarea o comportamiento, no suele medirse como un resultado de la ciencia ciudadana. Sin embargo, la participación en un proyecto de ciencia ciudadana puede mostrar a los participantes que son capaces de realizar ciencia, incluso si no se consideraban previamente como científicos. La motivación puede verse como un factor que influye en la decisión de una persona de participar en un proyecto, así como un resultado del mismo. Ya se ha descrito la motivación como insumo para el compromiso en un proyecto. La motivación como resultado incluye la motivación para seguir participando en un proyecto, para ser más activo y para participar en otras actividades relacionadas.

Además de conocer mejor las motivaciones para participar en proyectos de ciencia ciudadana, se considera importante entender cómo cambian las motivaciones a lo largo de un proyecto. Sin embargo, no es lo mismo comparar a los participantes nuevos con los de larga duración que medir el cambio de motivación a lo largo del tiempo dentro del mismo grupo.

2.3. Motivaciones de los Líderes de Proyectos

A menudo, el resultado más obvio al que se aspira desde la gestión de los proyectos es el aumento de los conocimientos y la comprensión de los participantes sobre un tema específico o la alfabetización científica en general. Aunque esta no suele ser la motivación más importante para los participantes, se puede conseguir un aumento de los conocimientos (ya sea medido o autodeclarado). Según estudios realizados por Agnello (2014), cumplir con los rasgos de personalidad, los valores, las emociones y los intereses individuales de los participantes en la ciencia ciudadana resulta una tarea compleja. Estos aspectos determinan las motivaciones, expectativas y barreras del público objetivo, que pueden variar en cada fase del proyecto y de la experiencia de los participantes. La forma en que se comunican los objetivos del proyecto, las tareas y los mensajes de captación representa un factor clave. Inicialmente, la disposición a participar en la ciencia ciudadana puede verse afectada por el

grado en que la estrategia de comunicación sea inclusiva y se ajuste a la motivación de los participantes. Por ejemplo, la expresión ”nivel de conocimientos” puede no ser la adecuada durante la captación, ya que la distinción entre participantes ”expertos” e ”inexpertos” puede desalentar la participación. Se considera que los posibles participantes sin las competencias necesarias en términos de capacidad o sin experiencia previa, pero cuyo entusiasmo y conocimientos pueden construirse con el tiempo, podrían sentirse excluidos. Según Bruyere y Rappe (2007), Measham y Barnett (2008), Wright et al. (2015), y Agnello et al. (2020), para construir una comunidad inclusiva y efectivamente comprometida, se sugiere que el equipo del proyecto debe garantizar un entendimiento común de la calidad de los datos entre los participantes y proporcionar una retroalimentación constructiva, teniendo cuidado de corregir a los participantes sin desmotivarlos.

2.4. Conclusión

Como vemos, los intereses y/o motivaciones de los participantes y de los líderes de los proyectos, no suelen ser los mismos. Para los participantes, el interés más común pasa por el hecho de que están contribuyendo a la ”ciencia real” o al objetivo general del proyecto, el tema concreto del mismo, el disfrute, el ocio, la interacción social o por formar parte de una comunidad de personas con ideas afines. Es importante entender y tener en cuenta, que las motivaciones de los participantes pueden ir cambiando a lo largo del tiempo y de los proyectos.

Por el contrario, para los líderes de proyectos, el interés principal suele centrarse en el aumento de los conocimientos y la comprensión de los participantes sobre un tema específico o la alfabetización científica en general. Es por esto que es esencial contar con participantes que no solamente estén interesados en la temática, sino que realicen aportes ricos en información.

Por lo tanto, es importante tener presente y diferenciar los intereses y motivaciones, tanto para participantes como para líderes de proyectos a la hora de pensar en recomendaciones. Al pensar en recomendaciones de proyectos a usuarios para su participación, se sugiere considerar aquellos proyectos que más se asemejen a los proyectos en los que participaron, evaluando el nivel de interés a lo largo del tiempo. Por otro lado, al considerar recomendaciones de usuarios a proyectos, es decir, a los líderes de los proyectos, se recomienda considerar aquellos usuarios que mostraron un buen desempeño en proyectos similares, asumiendo que podrían realizar aportes científicos más valiosos al proyecto en cuestión.

En el contexto del proyecto Ágora, resultaría relevante facilitar el acercamiento

entre proyectos y usuarios, teniendo en cuenta los intereses de cada parte y cómo estos pueden evolucionar con el tiempo. Este enfoque podría ayudar a mejorar y aumentar el nivel de conocimientos científicos en los temas propuestos por los líderes, contribuyendo así al avance general de la ciencia ciudadana.

Capítulo 3

Cientopolis y Ágora

En este capítulo, se presenta a Cientopolis como plataforma de ciencia abierta y ciencia ciudadana, Ágora como sistema web para el desarrollo de proyectos de ciencia ciudadana y PPSR-Core como estándar de modelo de datos de ciencia ciudadana.

3.1. Cientópolis

Cientópolis, una plataforma de ciencia abierta y ciencia ciudadana, fue desarrollada por la Facultad de Informática de la Universidad Nacional de La Plata, específicamente en el Laboratorio de Investigación y Formación en Informática Avanzada (LIFIA). Esta plataforma impulsa una participación activa de la ciudadanía en investigaciones científicas, incorporando elementos de ludificación para hacer que las actividades de ciencia ciudadana sean más atractivas y agradables.

La ludificación se refiere al proceso de integrar mecánicas de juego en actividades que originalmente no fueron concebidas como juegos, para así aumentar el compromiso y motivación. En el caso de Cientópolis, se busca que los voluntarios se involucren en las actividades de ciencia ciudadana de una manera lúdica y divertida.

Según su sitio web [9], los objetivos principales de Cientópolis son:

- Formar una red de ciudadanos dispuestos a colaborar en proyectos propuestos por científicos.
- Proporcionar a los científicos una plataforma para implementar y gestionar sus proyectos de ciencia ciudadana.

- Estrechar los vínculos entre las instituciones científicas y la comunidad que las rodea.
- Fomentar y motivar la participación ciudadana en la ciencia.

3.2. Ágora

Avanzando en la línea del trabajo de la plataforma Cientópolis, el centro LIFIA desarrolló el proyecto de ciencia ciudadana denominado ”Ágora”. Este proyecto se caracteriza por la variedad de temáticas que aborda y por la diversidad de usuarios que participan, cada uno con sus propias orientaciones e intereses. En el sistema de Ágora, el proceso comienza cuando un usuario, denominado líder del proyecto, decide iniciar una nueva iniciativa de investigación o estudio. Este líder, junto con la colaboración de co-creadores, tiene la responsabilidad de definir claramente los objetivos y alcances del proyecto. Los co-creadores son usuarios que, aunque no inician el proyecto, se suman en su creación y definición, aportando su experiencia, conocimientos y perspectivas para enriquecer y dar forma al proyecto. Una vez que el proyecto ha sido creado y estructurado, se le asignan categorías que permiten a otros usuarios identificar y comprender la temática y el propósito del mismo de manera rápida y efectiva. Posteriormente, se diseña y crea un protocolo específico para el proyecto. Este protocolo es un conjunto estructurado de instrucciones o formulario de pasos, diseñado para guiar a los participantes en la recolección de informacion y registro de datos de manera coherente y uniforme. La definición de este protocolo garantiza que los datos recopilados sean consistentes, comparables y de alta calidad, independientemente del participante que los haya recolectado. Cuando otro usuario se interesa en un proyecto y decide colaborar, se postula como participante en el mismo. Este sigue el protocolo establecido para contribuir con datos. Al completar todos los pasos y criterios del protocolo, se dice que el participante ha realizado una ”muestra”. Cada participante puede realizar muchas muestras en un proyecto. Los participantes en los proyectos varían ampliamente en términos de sus intereses y características, que pueden abarcar desde las distintas temáticas, hasta la empatía con los organizadores.

Debido a la gran diversidad de proyectos y usuarios, se hace difícil recomendar proyectos a personas, así como personas a proyectos. Esta dificultad se debe, en parte, a que los intereses de los usuarios van cambiando a lo largo del tiempo sobre la gran variedad de categorías de proyectos.

En la actualidad, Ágora no posee la capacidad de realizar recomendaciones de proyectos a personas y viceversa, para participar. Esto es una buena estrategia que

ayudaría, desde el punto de vista de los usuarios, a acceder a proyectos que son de su interes y, desde el punto de vista de los líderes de proyectos, a acceder a usuarios que puedan estar interesados en realizar aportes de calidad, ayudando en el aumento de conocimiento de la temática propuesta.

3.2.1. Desarrollo de Ágora

Ágora se comenzó a desarrollar con vista en el uso de tecnologías y paradigmas actuales. El paradigma elegido para este desarrollo, fue el de micro servicios. La arquitectura de microservicios estructura a la aplicación como un conjunto de servicios débilmente acoplados, permitiendo entregar aplicaciones grandes y complejas de manera rápida, frecuente y segura.

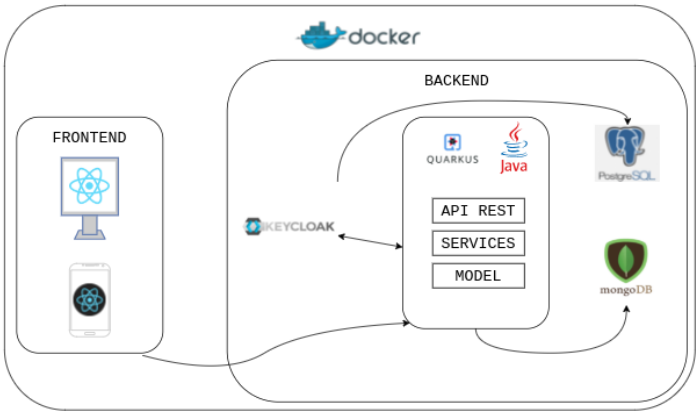

Figura 3.1: Tecnologías utilizadas en Ágora

Ágora está compuesto por un frontend, desarrollado en el lenguaje de programación ReactJS para versión web, Ionic React, para versión móvil; y un backend, desarrollado en el lenguaje de programación Java.

Frontend

- ReactJS [30]: es una biblioteca de código abierto de Javascript, que se desarrolló para la construcción de componentes de interfaces de usuario web y nativas. Esta puede ser utilizada para el desarrollo de single-page applications o aplicaciones móviles, a partir de componentes individuales, que luego pueden ser combinados en pantallas. Estos componentes son funciones de

JavaScript, y React proporciona la capacidad de añadir interactividad a estas interfaces, respondiendo a interacciones del usuario y actualizando la pantalla acorde a los nuevos datos recibidos.

- Ionic React [6]: es un proyecto de código abierto que ofrece componentes de interfaz de usuario (UI) multiplataforma y funcionalidades nativas para construir aplicaciones para iOS, Android, Electron y Aplicaciones Web Progresivas (PWA) utilizando React y tecnología web estándar. Es la versión oficial de React del popular framework Ionic, ampliamente utilizado por desarrolladores y empresas para crear aplicaciones críticas para su operación.

Backend

- Java & Quarkus: Java es un lenguaje de programación orientado a objetos, utilizado para el desarrollo de aplicaciones multiplataforma. Quarkus [28], por su parte, es un framework completo y nativo de Kubernetes hecho para máquinas virtuales Java (JVM) y compilación nativa, que optimiza Java para su utilización en el desarrollo y despliegue en contenedores, permitiéndole convertirse en una plataforma eficaz para entornos serverless, en la nube y en Kubernetes. Una de las características más importantes de Quarkus, es el bajo consumo de memoria y la rapidez de arranque, lo cual permite mantener bajos costos a la hora del alojamiento de una aplicación en un hosting.

- Keycloak [8]: es un producto de código abierto que proporciona una solución para la administración de usuarios en una aplicación. Keycloak facilita la protección de aplicaciones y servicios con poco o ningún código, lo que permite a los desarrolladores centrarse en la funcionalidad empresarial sin tener que preocuparse por los aspectos de seguridad de la autenticación.

- OpenAPI [19]: es una especificación que permite la descripción de una API remota, a la cual se puede acceder mediante el protocolo HTTP. A través de la especificación OpenAPI, el backend puede comunicar su funcionalidad al frontend de manera estandarizada, permitiendo que la interfaz de usuario se actualice dinámicamente en función de los cambios en el backend.

Base de Datos

Para base de datos, se utiliza MongoDB [11]. Esta es un sistema de base de datos NoSQL, distribuida y basada en documentos. Esta no maneja la noción de

esquemas bajo una estructura fija, guardando a la información en formato BSON (Binary JSON) y permitiendo trabajar con estructuras dinámicas y flexibles. El uso de MongoDB como sistema de base de datos en Ágora permite el manejo de datos estructurados de una manera más flexible que con las bases de datos relacionales. Gracias a su naturaleza basada en documentos, MongoDB ofrece la posibilidad de trabajar con estructuras de datos que pueden cambiar con el tiempo, lo que se alinea perfectamente con la naturaleza dinámica de los proyectos de ciencia ciudadana en Ágora. Además, MongoDB permite escalar horizontalmente de manera eficiente, es decir, se pueden añadir más servidores para incrementar la capacidad de procesamiento de datos.

Despliegue de la Aplicación

Con el objetivo de garantizar la versatilidad y escalabilidad del sistema Ágora, el proceso de despliegue se diseñó con una arquitectura flexible en mente. Esta arquitectura es capaz de ser desplegada en una amplia variedad de servidores, desde máquinas locales hasta clústeres en servicios de la nube como Amazon Web Services o Heroku, minimizando la necesidad de realizar cambios significativos entre despliegues.

El componente clave de esta estrategia es Docker [5], un proyecto de código abierto que facilita el desarrollo y despliegue de aplicaciones dentro de contenedores de Linux. Estos contenedores proporcionan un entorno aislado para cada aplicación, separándola del sistema operativo subyacente. Esto permite que las aplicaciones sean fácilmente portables y desplegables, tanto en entornos locales como en la nube.

Además, Docker contribuye a la eficiencia en el uso de recursos. Los contenedores solo requieren los recursos necesarios para ejecutar la aplicación que contienen, reduciendo la sobrecarga asociada con la ejecución de máquinas virtuales completas. Este enfoque de contenedores permite un uso más eficiente de los recursos de hardware y puede contribuir a reducir los costos operativos.

3.3. Modelo de Datos PPSR-Core

En esta sección, se presenta el modelo de datos PPSR-Core, una especificación clave para la estandarización de la información en proyectos de ciencia ciudadana. Se describe la estructura integral del modelo, subrayando su importancia y utilidad en el ámbito de la ciencia ciudadana. Se detallan las entidades y relaciones fundamentales que conforman el modelo. Asimismo, se analizan las propiedades específicas

de cada entidad. Finalmente, se examinan los metadatos vinculados al proyecto, ofreciendo una perspectiva holística de cómo el PPSR-Core facilita la organización y estandarización de los datos en este tipo de proyectos.

3.3.1. Asociación de Ciencia Ciudadana (CSA)

La *Citizen Science Association (CSA)* [3] es una organización que conecta a personas con una amplia gama de experiencias en torno a un propósito compartido: avanzar en el conocimiento de la investigación y el monitoreo realizado por, para y con miembros del público.

La ciencia ciudadana está ampliando el alcance, la relevancia y el impacto de la ciencia en casi todas las áreas de investigación, a través de esfuerzos locales y globales. Con una mayor atención a la ciencia ciudadana, la CSA trabaja en profundizar la comprensión de la misma, tanto en términos de participación pública como investigación, iluminando la integridad y complejidad de la práctica.

La CSA llega a una audiencia de más de 10.000 personas involucradas en el diseño, liderazgo, gestión y estudio de la ciencia ciudadana para intercambiar conocimientos, identificar preocupaciones compartidas y avanzar en el trabajo innovador. La CSA se asocia con organizaciones en otros continentes para participar en una conversación global, pero centra sus esfuerzos de convocatoria y comunicación hacia una audiencia norteamericana. Además, trabajan para construir alianzas estratégicas con otras sociedades (de distintas disciplinas y prácticas) para mantener altas las expectativas hacia la ciencia ciudadana.

3.3.2. Estándar de Modelo de Datos de Ciencia Ciudadana PPSR-Core

En el ámbito de la ciencia ciudadana, existe un modelo de datos que define un estándar para datos y metadatos globales y transdisciplinarios, para utilizar en proyectos de *Public Participation in Scientific Research (PPSR)*. Estos estándares están unidos, apoyados y respaldados por un framework común, que ilustra como se estructura la información en el ámbito de la ciencia ciudadana. Esto, permite que los datos se utilicen en diversas plataformas y proyectos de forma uniforme, contribuyendo a los objetivos de investigación de la comunidad científica. El *Public Participation in Scientific Research Core (PPSR-Core)* es mantenido por la CSA [25].

El modelo (figura 3.2) de datos está compuesto por tres grandes esquemas:

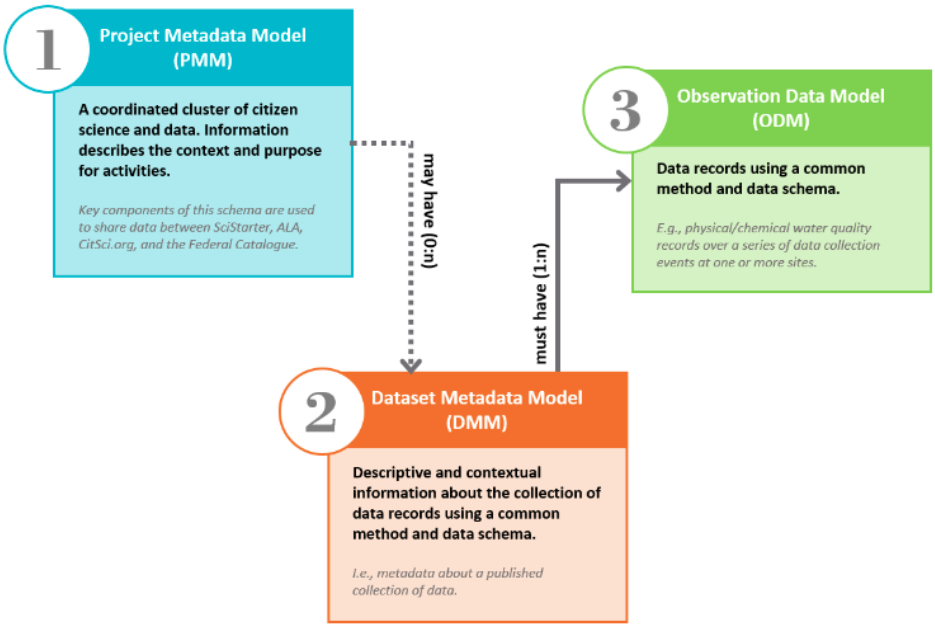

Figura 3.2: PPSR-Core - Esquema general [24]

- *Project Metadata Model (PMM)*: es un modelo de metadatos que describe los proyectos. Los metadatos a nivel proyecto proporcionan el marco de organización y contexto para la recolección de datos y tareas asociadas, permitiendo que los proyectos sean descubiertos y accedidos en diferentes ubicaciones, periodos de tiempo, temas, entre otros. A su vez, permite a los líderes del proyecto explicar el objetivo a lograr y alentar a las personas a participar en él. Incluye metadatos que describen el contexto y el propósito de las actividades. [27]
- *Dataset Metadata Model (DMM)*: es un modelo de metadatos que describe una colección de observaciones. Proporciona el contexto para la colección de registros de observación y expresa información asociada y común a todos los registros dentro de un dataset. [4]
- *Observation Data Model (ODM)*: describe un número de perfiles para dominios particulares (o áreas de la ciencia) y conjuntos de atributos básicos que deben incluirse para tipos particulares de encuestas. [17]

Estos estándares están unidos, respaldados y subrayados por un marco común, denominado *Common Data Model (CDM)*. Este es un modelo de datos que une pro-

yectos, datasets y observaciones. Permite que los proyectos se agreguen en un concepto de nivel superior de un programa o campaña, proporcionando contexto para grupos de proyectos que se están llevando a cabo dentro de un marco organizativo común. Como objeto central, el CDM es necesario para la implementación completa de esta norma.

El PPSR-Core puede ser utilizado como un esquema de componentes individuales, como combinación de esquemas individuales o en su totalidad como conjunto de esquemas unidos por el esquema CDM. La configuración a utilizar en nuestra aplicación será determinada por el caso de uso acorde a nuestra situación. Por ejemplo, si estamos desarrollando una aplicación web y/o móvil de un solo proyecto para recoger datos, es necesario utilizar la combinación del PMM, DMM y el perfil de ODM específico del dominio, unidos de acuerdo con el esquema del CDM. Esta flexibilidad permite a los usuarios adaptar la implementación del PPSR-Core a las necesidades y requisitos específicos de su proyecto.

3.4. Conclusión

En este capítulo, se exploraron tres componentes clave para la evolución de la ciencia ciudadana: la plataforma Cientópolis, el proyecto Ágora y el estándar de modelo de datos PPSR-Core. Cientópolis actúa como un puente entre la comunidad científica y los ciudadanos, usando la ludificación para mejorar el compromiso. Ágora, una extensión de Cientópolis, se destaca por su diversidad temática y de usuarios, aunque enfrenta desafíos en la recomendación de proyectos. PPSR-Core estandariza la recopilación de datos, facilitando la colaboración y el intercambio de información.

Capítulo 4

Sistemas de Recomendación

En este capítulo, se desarrollarán los conceptos de sistemas de recomendación, sistemas de recomendación basados en el contenido y dos tecnologías útiles para su desarrollo, Machine Learning y Neo4j.

4.1. Definición

El término **sistema de recomendación (RS, en inglés, *recommender system*)** se refiere a todas las herramientas y técnicas que, utilizando el conocimiento que pueden recolectar acerca de los usuarios e ítems en cuestión, puedan proveer sugerencias sobre ítems que pueden resultarle de interés a un usuario particular. En este contexto, un ítem es un término general utilizado para identificar lo que el sistema le recomienda al usuario. Un sistema de recomendación normalmente se enfoca en un tipo específico o clase de ítems, como qué libros comprar, qué artículos leer o qué hoteles reservar. Estos proveen una experiencia personalizada, ayudando a las personas a encontrar lo que están buscando o lo que pueda llegar a interesar, de forma más rápida. Como resultado final, la satisfacción de los usuarios será mayor, ya que obtendrán resultados relevantes en un lapso corto de tiempo. [1]

4.2. Tipos de Sistemas de Recomendación

Existen dos grandes grupos en los que podemos diferenciar a los sistemas de recomendación. Estos son los *personalizados* y los *no personalizados*.

4.2.1. No Personalizados

Según Poriya, Patel, Bhagat y Sharma [16], los sistemas de recomendación no personalizados representan la forma más simple de sistemas de recomendación. Este tipo de sistemas no toma en cuenta las preferencias personales de los usuarios. Las recomendaciones generadas por estos sistemas son idénticas para cada usuario. Estos sistemas recomiendan ítems a los usuarios basándose en lo que otros usuarios han comentado sobre los ítems o en cómo los han calificado en promedio. Debido a su naturaleza, las recomendaciones generadas por este tipo de sistemas son idénticas para todos los usuarios, ya que se basan en estadísticas generales en lugar de en preferencias individuales.

4.2.2. Personalizados

Según Jannach et al [12], en los sistemas de recomendación personalizados, todos los usuarios reciben una lista de recomendaciones distintas dependiendo de sus gustos, los cuales son inferidos de las interacciones previas o información obtenida utilizando diferentes técnicas. En este tipo de recomendación, se requiere que el sistema conozca algunas o muchas cosas acerca de cada usuario y cada ítem. Para hacer esto, el sistema de recomendación necesita desarrollar y mantener un modelo o perfil de cada usuario, que puede incluir información sobre las preferencias del mismo. También se requiere un modelo o perfil de cada ítem que incluye información relevante sobre este. La construcción del modelo de usuarios y de ítems es central en todos los sistemas de recomendación. Sin embargo, la forma en que esta información es obtenida, modelada y explotada depende de la técnica de recomendación particular y del algoritmo de filtrado de información relacionado. Existen tres grandes tipos de filtrados de información: *Filtrado basado en el contenido (content-based filtering)*, *Filtrado colaborativo (collaborative filtering)* y *Filtrado híbrido (hybrid filtering)*.

- Filtrado basado en el contenido (content-based filtering)[12]: esta técnica aprovecha la disponibilidad de las descripciones de los ítems (creada manualmente o extraída de forma automática) y de los perfiles de los usuarios, que le asigna importancia a diferentes características. La esencia de este enfoque radica en asignar importancia a diferentes características y aprender a identificar ítems cuyo contenido sea similar al de aquellos que el usuario ha preferido en el pasado.
- Filtrado colaborativo (collaborative filtering)[12]: la idea básica detrás de las

recomendaciones colaborativas, es que si usuarios compartieron los mismos intereses en el pasado (por ejemplo, si compraron libros similares o miraron películas similares), es probable que tengan comportamientos similares en el futuro. Esta premisa permite al sistema de recomendación sugerir ítems basándose en las preferencias de otros usuarios con perfiles similares.

- Filtrado híbrido (hybrid filtering)[12]: esta técnica combina los enfoques del filtrado basado en el contenido y el filtrado colaborativo, superando las limitaciones de ambos enfoques al considerar tanto los metadatos (colaborativos) como los datos transaccionales (basados en el contenido). En un motor de recomendación híbrido, se pueden generar etiquetas de procesamiento del lenguaje natural para cada producto o artículo (película, canción, etc), y utilizar ecuaciones vectoriales para calcular la similitud de los productos. A continuación, se puede utilizar una matriz de filtrado colaborativo para recomendar artículos a los usuarios en función de sus comportamientos, actividades y preferencias. Netflix es un ejemplo de motor de recomendación híbrido. Tiene en cuenta tanto los intereses del usuario (colaborativo) como las descripciones o características de la película o el programa (basado en el contenido).

4.3. Funcionamiento de un Sistema de Recomendación

Un sistema de recomendación opera mediante una combinación de datos y tecnología de aprendizaje de máquina. Los datos son esenciales en el desarrollo de estos sistemas; a mayor cantidad de datos, mayor eficiencia y efectividad para proporcionar sugerencias relevantes. El funcionamiento de un sistema de recomendación se puede dividir en cuatro etapas:

1. **Recolección de datos**: el primer y más importante paso para la creación de un sistema de recomendación es recolectar datos. Hay dos tipos principales de datos que deben recopilarse:

    - Datos implícitos: incluye información recolectada de actividades como historial de búsqueda web, clicks, eventos del carro de compras, log de búsqueda, historial de pedidos, etc.
    - Datos explícitos: esta información es recolectada de las entradas del usuario, como puede ser reseñas y puntajes, me gustas y no me gustas, comentarios, etc.

Los motores de recomendación también utilizan atributos del usuario, como pueden ser datos demográficos (edad, género) y psicográficos (intereses, valores), para identificar usuarios similares, así como datos de características (género, tipo de ítem) para identificar la similitud entre ítems.

2. **Almacenamiento de datos**: una vez obtenidos los datos, hay que almacenarlos. Con el tiempo, la cantidad de datos será enorme. Esto significa que hay que disponer de un almacenamiento amplio y escalable. Dependiendo del tipo de datos que se recopilen, existen diferentes tipos de almacenamiento.

3. **Análisis de datos**: para poder utilizarlos, hay que profundizar en los datos y analizarlos. Hay varias formas de analizar los datos. Entre ellas:

   - Análisis en tiempo real: los datos se procesan a medida que se crean.
   - Análisis por lotes: los datos se procesan de manera periódica.
   - Análisis casi en tiempo real: los datos se procesan en minutos en lugar de en segundos cuando no se necesitan inmediatamente.

4. **Filtrado de datos**: la etapa final consiste en el filtrado. En este punto, se aplican diversas matrices, reglas y fórmulas matemáticas a los datos, dependiendo de si se utiliza un enfoque de filtrado de recomendaciones colaborativo, basado en contenido o híbrido. El resultado de este filtrado son las recomendaciones que se presentan a los usuarios [34].

En el marco de la presente tesina, y particularmente en el contexto del proyecto Ágora, se aborda el tema de las recomendaciones basadas en el contenido. El funcionamiento específico de este tipo de sistema de recomendación se detalla en la sección siguiente.

4.4. Sistemas de Recomendación Basados en el Contenido

En esta sección, se describirá en profundidad el funcionamiento general de un sistema de recomendación basado en el contenido. Se abordarán aspectos como su arquitectura, el modelado de perfiles, así como sus ventajas y desventajas. La información detallada en la siguiente sección y sus imágenes, fueron extraídas del artículo *Content-based Recommender Systems: State of the Art and Trends* [20].

4.4.1. Funcionamiento General

Los sistemas que utilizan un enfoque de recomendación basado en el contenido se centran en el análisis de un conjunto de documentos y/o descripciones de ítems previamente calificados por un usuario. Basándose en las características de los ítems valorados, estos sistemas construyen un modelo o perfil que refleja los intereses del usuario. Este perfil, una representación estructurada de las preferencias del usuario, es utilizado para recomendar nuevos ítems que podrían ser de interés.

En el proceso de recomendación, los atributos del perfil del usuario se comparan con los atributos de un ítem. Esto resulta en un juicio de relevancia que representa el nivel de interés del usuario hacia ese ítem. Si el perfil logra capturar con precisión las preferencias del usuario, proporciona una ventaja significativa para la eficacia del proceso de acceso a la información. Por ejemplo, este perfil podría utilizarse para filtrar los resultados de búsqueda, permitiendo determinar si un usuario está interesado o no en una página web específica y, en caso de que no lo esté, prevenir su visualización.

4.4.2. Arquitectura

Los sistemas de recomendación basados en contenido requieren técnicas adecuadas para representar los ítems y construir el perfil del usuario, así como estrategias para comparar este perfil con la representación de los elementos. El proceso de recomendación se lleva a cabo en tres pasos, gestionados por componentes independientes (figura 4.1):

- Content analyzer: cuando la información no tiene estructura (por ejemplo, un texto), se necesita algún tipo de preprocesamiento para extraer la información relevante estructurada. La principal responsabilidad de este componente es representar el contenido de los ítems (por ejemplo, documentos, páginas web, noticias, descripciones de productos, etc.) procedentes de las fuentes de información en una forma adecuada para los siguientes pasos de procesamiento. Los ítems se analizan mediante técnicas de extracción de características para trasladar la representación de los ítems del espacio de información original al espacio de destino (por ejemplo, páginas web representadas como vectores de palabras clave).

- Profile learner: este módulo recoge datos representativos de las preferencias del usuario y trata de generalizar estos datos, para construir el perfil del usuario. Normalmente, la estrategia de generalización se realiza mediante técnicas de

aprendizaje automático, que son capaces de inferir un modelo de intereses del usuario a partir de los elementos que le han gustado o no en el pasado.

- Filtering component: este módulo aprovecha el perfil del usuario para sugerir ítems relevantes comparando la representación del perfil con la de los elementos que se van a recomendar. El resultado es un juicio de relevancia binario o continuo (calculado mediante algunas métricas de similitud). En este último caso, se obtiene una lista clasificada de ítems potencialmente interesantes.

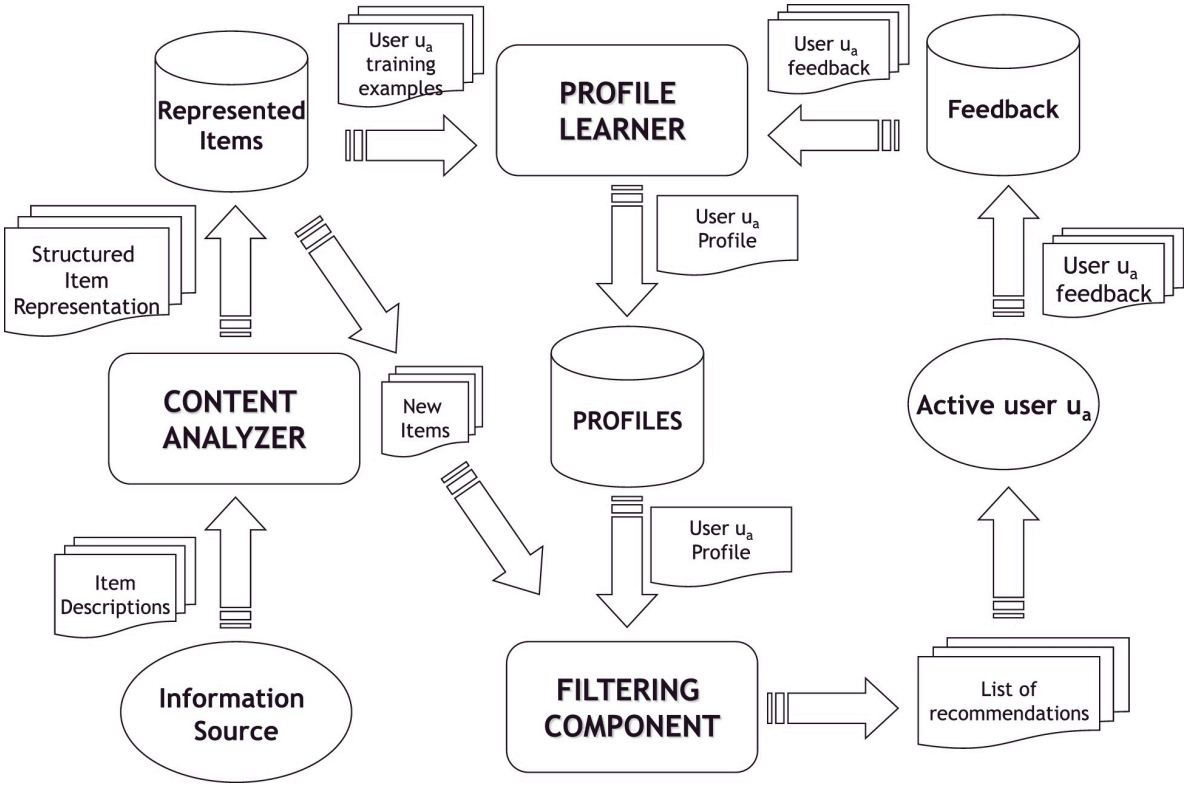

Figura 4.1: Arquitectura de un sistema de recomendación basado en contenido (de *Content-based Recommender Systems: State of the Art and Trends* [20])

El primer paso del proceso de recomendación lo realiza el *Content Analyzer*. Las descripciones de los ítems procedentes de *Information Source* son procesadas por el *Content Analyzer* que extrae características (palabras clave, n-gramas, conceptos, etc.) del texto no estructurado para producir una representación estructurada del ítem, la cual se almacena en el repositorio de *Represented Items*.

Para construir y actualizar el perfil del usuario u_a (al que se proporcionarán las recomendaciones), se recogen de alguna manera sus reacciones a los ítems y se registran en el repositorio *Feedback*. Estas reacciones, denominadas *annotations* o *feedback*, junto con las descripciones de los ítems relacionados, se explotan durante el proceso de aprendizaje de un modelo útil, para predecir la relevancia real de los nuevos ítems presentados. Los usuarios también pueden definir explícitamente sus áreas de interés como perfil inicial. Normalmente, es posible distinguir entre dos tipos de información de relevancia: la información positiva (que infiere características que le gustan al usuario) y la información negativa (es decir, que infiere características que no son de interés para al usuario).

Se pueden adoptar dos técnicas diferentes para registrar los comentarios de los usuarios. Cuando un sistema requiere que el usuario evalúe explícitamente los ítems, esta técnica suele denominarse ”feedback explícito”; la otra técnica, denominada ”feedback implícito”, no requiere ninguna participación activa del usuario, en el sentido de que el feedback se deriva de la supervisión y el análisis de las actividades del usuario.

Las evaluaciones explícitas indican lo relevante o interesante que es un ítem para el usuario. Existen tres enfoques principales para obtener información explícita sobre la relevancia:

- Me gusta/no me gusta: los elementos se clasifican como ”relevantes” o ”no relevantes” adoptando una escala de valoración binaria simple.
- Calificaciones: se suele adoptar una escala numérica discreta para evaluar los elementos.
- Comentarios: se recogen los comentarios sobre un solo ítem y se presentan a los usuarios como medio para facilitar el proceso de toma de decisiones.

El feedback explícito tiene la ventaja de la simplicidad, aunque la adopción de escalas numéricas/simbólicas aumenta la carga cognitiva del usuario, y puede no ser adecuada para captar el sentimiento real del usuario sobre los ítems. Los métodos de feedback implícito se basan en la asignación de una puntuación de relevancia a acciones específicas del usuario sobre un ítem, como guardar, descartar, imprimir, marcar, etc. La principal ventaja es que no requieren la participación directa del usuario, aunque es probable que se produzcan sesgos, por ejemplo, la interrupción de una llamada telefónicas durante la lectura.

Para construir el perfil del usuario activo u_a hay que definir el conjunto de entrenamiento TR_a para u_a. TR_a es un conjunto de pares (I_k, r_k), donde r_k es la calificación proporcionada por u_a en la representación del ítem I_k. Dado un conjunto de

representaciones de ítems etiquetados con valoraciones, el *Profile Learner* aplica algoritmos de aprendizaje supervisado para generar un modelo predictivo. El perfil del usuario se almacena en un repositorio de perfiles para su uso posterior por parte del *Filtering Component*. Dada la representación de un nuevo ítem, el *Filtering Component* predice si es probable que sea de interés para el usuario activo, comparando las características de la representación del ítem con las de la representación de las preferencias del usuario (almacenadas en el perfil del usuario). Normalmente, el *Filtering Component* implementa algunas estrategias para clasificar los ítems potencialmente interesantes según la relevancia con respecto al perfil del usuario. Los elementos mejor clasificados se incluyen en una lista de recomendaciones L_a, que se presenta a u_a. Los gustos del usuario suelen cambiar con el tiempo, por lo que es necesario mantener la información actualizada y proporcionársela al *Profile Learner* para actualizar automáticamente el perfil del usuario. Sobre las recomendaciones generadas se recoge el *feedback*, dejando que los usuarios expresen su satisfacción o no satisfacción con los ítems de L_a. Tras recoger estos comentarios, el proceso de aprendizaje se realiza de nuevo con el nuevo conjunto de entrenamiento, y el perfil resultante se adapta a los intereses actualizados del usuario. La iteración del ciclo de retroalimentación-aprendizaje a lo largo del tiempo permite al sistema tener en cuenta la naturaleza dinámica de las preferencias de los usuarios.

4.4.3. Ventajas y Desventajas

El paradigma de recomendación basado en contenido presenta varias ventajas en comparación con el enfoque colaborativo:

- Independencia del usuario: los sistemas de recomendación basados en el contenido explotan únicamente las valoraciones proporcionadas por el usuario activo para construir su perfil. En cambio, los métodos de filtrado colaborativo necesitan las valoraciones de otros usuarios para identificar a los ”vecinos más cercanos” del usuario activo, es decir, aquellos usuarios con gustos similares, ya que han valorado los mismos ítems de forma parecida.

- Transparencia: los sistemas basados en contenido pueden proporcionar explicaciones sobre sus recomendaciones, enumerando explícitamente las características o descripciones del contenido que llevaron a que un ítem aparezca en la lista de recomendaciones. Estos detalles pueden ser útiles para que los usuarios decidan si confiar en una recomendación. Por el contrario, los sistemas colaborativos son cajas negras, ya que la única explicación de la reco-

mendación de un ítem es que ese ítem ha gustado a usuarios desconocidos con gustos similares.

- Nuevo ítem: los sistemas de recomendación basados en el contenido son capaces de recomendar ítems que aún no han sido valorados por ningún usuario. Por lo tanto, no sufren el problema del ”primer evaluador”, que afecta a los sistemas de recomendación colaborativos, que se basan únicamente en las preferencias de los usuarios para hacer recomendaciones. En estos últimos, hasta que un nuevo ítem no sea valorado por un número suficiente de usuarios, el sistema no podrá recomendarlo.

Sin embargo, los sistemas basados en el contenido también tienen varias limitaciones:

- Análisis de contenido limitado: las técnicas basadas en contenido encuentran un límite natural en el número y el tipo de características que se asocian, de forma automática o manual, a los ítems que se recomiendan. A menudo, se requiere conocimiento del dominio para generar recomendaciones efectivas. Por ejemplo, para las recomendaciones de películas, el sistema necesita conocer a los actores y directores. Ningún sistema de recomendación basado en el contenido puede proporcionar sugerencias adecuadas si el contenido analizado no contiene suficiente información para distinguir los ítems que le gustan al usuario de los que no le gustan. Algunas representaciones solo captan ciertos aspectos del contenido, pero hay muchos otros que pueden influir en la experiencia del usuario. Por ejemplo, en el caso de las páginas web, las técnicas de extracción de características a partir del texto ignoran por completo las cualidades estéticas y la información multimedia adicional. En resumen, tanto la asignación automática como la manual de características a los ítems podría no ser suficientes para definir los aspectos distintivos de los ítems que resultan necesarios para la obtención de los intereses de los usuarios.

- Superespecialización: los sistemas de recomendación basados en contenido carecen de un método inherente para encontrar algo inesperado. El sistema sugiere ítems cuyas puntuaciones son altas cuando coincide con el perfil del usuario, llevando a recomendaciones de ítems similares a los ya valorados. Este inconveniente también se denomina ”problema de serendipia”, aludiendo a la tendencia de los sistemas basados en contenido a producir recomendaciones con un grado limitado de novedad. Por ejemplo, si un usuario solo ha valorado películas dirigidas por Stanley Kubrick, se le recomendará solo ese

tipo de películas. Una técnica perfecta basada en el contenido rara vez encontraría algo novedoso, lo que limitaría la gama de aplicaciones para las que sería útil.

- Nuevo usuario o *cold start* : es necesario recopilar suficientes valoraciones para que un sistema de recomendación basado en contenido logre entender realmente las preferencias del usuario y ofrecer recomendaciones precisas. Cuando hay pocas valoraciones disponibles, como en el caso de un nuevo usuario, el sistema puede no ser capaz de proporcionar recomendaciones precisas.

4.5. Tecnologías para Sistemas de Recomendación

En esta sección, se realiza una descripción de Machine Learning, sus tipos y su utilización en sistemas de recomendación.

4.5.1. Machine Learning

Según Arthur Samuel [10], pionero en el campo de los juegos informáticos y la inteligencia artificial, Machine Learning (ML) se define como el campo de estudio que otorga a las computadoras la capacidad de aprender sin ser programadas explícitamente. ML se utiliza para enseñar a las máquinas a manejar datos de manera más eficiente y aprender de ellos. El aprendizaje automático se basa en diferentes algoritmos, y el tipo de algoritmo empleado varía según el problema, el número de variables y el modelo adecuado. No existe un único algoritmo que sea el mejor para todos los problemas. Los algoritmos de aprendizaje automático pueden ser entrenados para realizar una variedad de tareas, como la clasificación, la regresión y la agrupación.

4.5.2. Tipos de Machine Learning

Existen diferentes tipos de ML, cada uno con sus propias características, métodos y aplicaciones. El tipo de ML a utilizar depende del tipo de problema que se está tratando de resolver y de los datos disponibles. Algunos tipos de ML son:

- *Supervised Learning*: busca aprender una función que mapea una entrada a una salida basada en pares de ejemplos de entrada-salida. Se infiere una fun-

ción a partir de datos de entrenamiento etiquetados que consisten en un conjunto de ejemplos de entrenamiento. Se considera que los algoritmos de aprendizaje automático supervisado son aquellos que requieren asistencia externa. Los algoritmos más utilizados son la regresión lineal, la regresión logística y las máquinas de vectores de soporte (SVM). Este tipo de aprendizaje es útil cuando el resultado que se desea predecir es conocido.

- *Unsupervised Learning*: en este tipo de entrenamiento, a diferencia del aprendizaje supervisado, no hay respuestas correctas y no se cuenta con un maestro. En estos métodos, los algoritmos se encargan de descubrir y presentar la estructura interesante en los datos. Se aprenden algunas características de los datos. Los algoritmos más utilizados son los de agrupamiento, como K-means y DBSCAN.

- *Semi-Supervised Learning*: es una combinación de métodos de aprendizaje automático supervisados y no supervisados. Este enfoque resulta útil en áreas del aprendizaje automático y la minería de datos, donde ya se cuenta con datos no etiquetados y obtener datos etiquetados representa un proceso tedioso. Con los métodos más comunes de aprendizaje automático supervisado, se entrena un algoritmo de aprendizaje automático utilizando un conjunto de datos etiquetado, en el cual cada registro incluye la información del resultado.

- *Reinforcement Learning*: requiere el aprendizaje a través de la interacción con el entorno. Estos algoritmos aprenden a tomar decisiones óptimas a través de la prueba y el error, recompensando las acciones que conducen a resultados positivos y penalizando las acciones que conducen a resultados negativos. Se preocupa por cómo los agentes de software deben tomar acciones en un entorno para maximizar alguna noción de recompensa acumulativa. Se clasifica como uno de los tres paradigmas básicos de aprendizaje automático, junto con el aprendizaje supervisado y el aprendizaje no supervisado.

- *Multitask Learning*: implica el entrenamiento de un modelo para realizar múltiples tareas. Se utiliza cuando varias tareas comparten estructuras o patrones similares y se puede beneficiar de compartir información entre ellas durante el proceso de aprendizaje, aprovechando las similitudes entre las diferentes tareas. Se considera que puede mejorar la eficiencia del aprendizaje y también actuar como un regularizador.

4.5.3. Uso de Aprendizaje Automático en Sistemas de Recomendación

Según Portugal, Alencar y Cowan [22], los sistemas de recomendación (RS) emplean conceptos de inteligencia artificial (IA) para ofrecer recomendaciones de ítems a los usuarios. Por ejemplo, una librería en línea podría utilizar un algoritmo de machine learning (ML) para clasificar libros por género y recomendar otros libros a un usuario interesado en realizar una compra. La introducción de los RS se remonta a 1992 con la aparición de Tapestry, el primer sistema de recomendación. Sus autores acuñaron el término ”filtrado colaborativo” para referirse a la actividad de recomendación. Estos sistemas desempeñan un papel crucial en la toma de decisiones, asistiendo a los usuarios para maximizar beneficios o minimizar riesgos. En la actualidad, los RS se utilizan en numerosas aplicaciones informáticas, y su relevancia es alta, convirtiendo las recomendaciones conscientes del contexto en un campo de investigación desafiante.

El aprendizaje automático es un campo de la IA que se ocupa de utilizar computadoras para simular el aprendizaje humano. Esto permite que las computadoras identifiquen y adquieran conocimiento del mundo real y mejoren su rendimiento en determinadas tareas basadas en dicho conocimiento.

Los sistemas de recomendación utilizan algoritmos que permiten a las computadoras aprender en función de la información del usuario y personalizar aún más las recomendaciones. El aprendizaje automático es un campo de investigación de la IA que abarca algoritmos cuyo objetivo es predecir el resultado del procesamiento de datos. A pesar de los avances significativos de ML en campos como el reconocimiento de imágenes, motores de búsqueda y seguridad, este campo cuenta con varios algoritmos descritos en la literatura, con características que los hacen adecuados o no para una aplicación en particular. La literatura carece de una clasificación adecuada de los algoritmos que muestre el entorno en el que son adecuados para ser utilizados. Por lo tanto, elegir un algoritmo de ML para ser utilizado en RS es una tarea compleja.

Ventajas del uso de Aprendizaje Automático

El aprendizaje automático tiene una serie de ventajas sobre los métodos de aprendizaje tradicionales. Las ventajas del aprendizaje automático incluyen:

- Capacidad de aprender de grandes cantidades de datos.
- Capacidad de hacer predicciones precisas.

- Capacidad de adaptarse a nuevos datos.

Desventajas del uso de Aprendizaje Automático

El aprendizaje automático también tiene algunas desventajas. Las desventajas del aprendizaje automático incluyen:

- Necesidad de grandes cantidades de datos.
- Necesidad de expertos en aprendizaje automático.
- Posibilidad de sesgos.
- Necesidad de reentrenamiento periódico de los modelos para evitar el sobre ajuste y el olvido, garantizando que reflejen tanto las preferencias a largo plazo como las recientes del usuario [37].

4.5.4. Neo4j

En esta sección se realiza una descripción de Neo4j, detallando su funcionalidad y sus ventajas a la hora de realizar recomendaciones.

¿Qué es Neo4j?

Neo4j es una base de datos orientada a grafos nativa, NoSQL y de código abierto, que proporciona un backend transaccional compatible con ACID para sus aplicaciones. Almacenar nodos relacionados, aceleran las consultas.

Ventajas y Mejoras

- Neo4j ofrece el rendimiento de lectura y escritura que se necesita, al tiempo que protege la integridad de sus datos. Es la única base de datos orientada a grafos que combina el almacenamiento nativo de grafos, la arquitectura escalable optimizada para la velocidad y el cumplimiento de ACID para garantizar la previsibilidad de las consultas basadas en relaciones.
- La adyacencia libre de índices reduce el tiempo de lectura y mejora aún más a medida que aumenta la complejidad de los datos. Se obtienen transacciones rápidas y fiables con un rendimiento ultraalto en paralelo, incluso cuando sus datos crezcan.

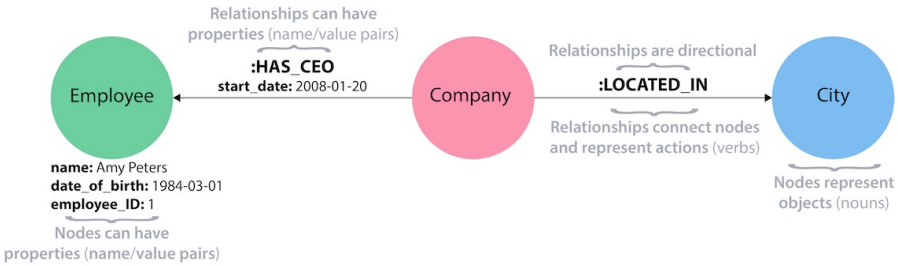

Figura 4.2: Modelo simple en grafos (de sitio oficial Neo4j [13])

- Los algoritmos de grafos proporcionan métodos de aprendizaje automático no supervisados y heurísticos, que aprenden y describen la topología de su grafo. Con la integración de grafos y los modelos entrenados dentro del espacio de trabajo de análisis, puede hacer predicciones sobre su grafo desde Neo4j.
- La tecnología orientada a grafos, permite combinar el comportamiento e historial de navegación y los datos demográficos de un usuario para analizar al instante sus elecciones actuales y ofrecerle de inmediato las recomendaciones pertinentes.
- La elaboración de recomendaciones eficaces en tiempo real depende de una plataforma de datos que comprenda las relaciones entre entidades, así como la calidad y la fuerza de esas conexiones. Solo la tecnología orientada a grafos rastrea eficazmente estas relaciones en función del historial de interacciones y las reseñas de los usuarios para proporcionar la información más significativa sobre las necesidades de los clientes y las tendencias de los productos.
- Los motores de recomendación basados en grafos suelen adoptar dos enfoques principales: identificar los recursos de interés para los individuos o identificar a los individuos que probablemente estén interesados en un recurso determinado. En ambos casos, la tecnología de bases de datos orientadas a grafos establece las correlaciones y conexiones necesarias para ofrecer los resultados más relevantes para el individuo o el recurso en cuestión.[35] [23]

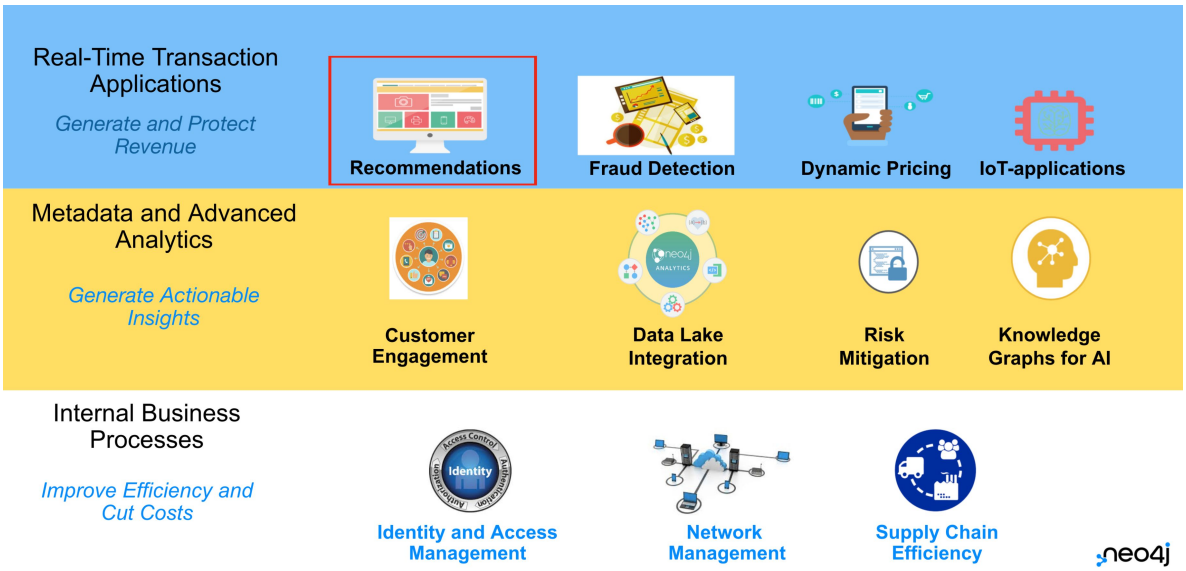

Figura 4.3: Neo4j - Casos de usos (de *What is a Graph Database?* [33])

Recomendaciones en Neo4j

Pensando en un modelado típico, donde la información es almacenada en bases de datos relacionales como MySql o bases de datos orientadas a documentos como MongoDB, podemos tener mucha información distribuida entre diferente bases de datos. Obteniendo la información de estas bases de datos, nos lleva a formar un lago de datos (data lake). Estos datos se pueden procesar para realizar recomendaciones utilizando técnicas de MapReduce, como por ejemplo Hadoop, pero el problema es que nos lleva a generar una sobrecarga que impacta en la velocidad para generar una sola recomendación para un único usuario. El procesamiento de la información para generar una recomendación se debería hacer de manera diaria o por hora, para que esta sea presentada al usuario en un tiempo considerable. Pero necesitamos que las recomendaciones se puedan realizar en tiempo real, en el momento en qué el usuario está interactuando con nuestra web.

Utilizando un enfoque orientado a grafos, como es Neo4j, donde tenemos procesamiento nativo de grafos y adyacencia libre de índices, podremos realizar consultas con características de rendimiento en tiempo constante en conjuntos de datos muy grandes, lo que nos permitirá generar nuestras recomendaciones en tiempo real.

Framework Inteligente de Recomendaciones en Neo4j

Neo4j es una base de datos nativa construida y diseñada para datos conectados. Teniendo en cuenta la creciente prevalencia de las recomendaciones, muchos han encontrado el modelo de grafos, útil para apoyar las recomendaciones ricas en contexto que se entienden intuitivamente y se desarrollan de manera rápida.

Hay varias razones por las cuales los grafos se prestan tan bien para las recomendaciones.

- Explicabilidad: Los modelos de grafos son fácil de entender y visualizar para usuarios no técnicos. Neo4j es una base de datos de grafos nativa, es decir, el grafo no es una simple abstracción o capa por encima de una base de datos relacional. Los datos son almacenados como grafos. El modelo de datos de grafos es amigable. Cuando se diseña un modelo de datos, es bastante intuitivo dibujar los datos en forma de entidades conectadas. En el mundo relacional, el modelo se reformatea para ajustarse a las tablas de un modelo relacional. Con Neo4j, sin embargo, no es necesario reestructurar porque los datos se almacenan en forma de nodos etiquetados y relaciones. Neo4j ha desarrollado el lenguaje de consulta Cypher, el cual es de código abierto. Cypher es un lenguaje declarativo, lo que significa que usted especifica los resultados que desea obtener, no lo que quiere que haga. Cuando se utiliza Cypher, el usuario especifica el patrón del grafo que desea recuperar de la base de datos y Neo4j se encarga del resto.

- Desarrollo rápido: Los requerimientos cambian de forma rápida, y los modelos necesitan adaptarse a ellos. Neo4j no posee esquema definido, dando la posibilidad de reestructurar el grafo para que considere nuevos datos o diferentes patrones de acceso. Muchos sistemas de recomendación se basan en gran medida en enfoques estadísticos. Para adaptarse a los cambios en los requisitos, es posible que haya que ajustar y reentrenar los modelos, lo que puede requerir mucho tiempo. En cambio, los patrones y algoritmos de grafos de Cypher pueden editarse y adaptarse rápidamente sin necesidad de reentrenamiento. Además, las bases de datos orientadas a grafos no tienen esquema y, por tanto, son más flexibles que las bases de datos relacionales. Los modelos de grafos pueden crecer y cambiar en función de las necesidades, a diferencia de los modelos relacionales, en los que hay que eliminar y reconstruir las tablas.

- Personalización y contextualización: Neo4j suele ser utilizado como una capa de persistencia que conecta datos de grandes almacenamientos. Teniendo

un acceso rápido a todo lo que se conoce acerca de un usuario o producto, permite realizar recomendaciones más relevantes. El modelo de grafos admite intrínsecamente una información rica y contextualizada. Si un grafo está bien modelado, cada nodo debería tener vecinos que proporcionen un contexto valioso sobre ese nodo. El valor del contexto personalizado a la hora de hacer recomendaciones es muy alto. Es probable que los usuarios tengan gustos diferentes, y el sistema de recomendación debe tenerlo en cuenta para maximizar su éxito.

- Performance: Los sistemas de recomendación necesitan analizar una gran cantidad de datos para producir resultados significativos. Las recomendaciones suelen ser útiles solo en tiempo real. El motor de grafos nativo de Neo4j lo convierte en una herramienta excepcional para producir recomendaciones en tiempo real. Neo4j utiliza la adyacencia libre de índices para recorrer el grafo en tiempo constante, evitando los joins computacionalmente costosos de una base de datos relacional. Esto nos permite anclar en un solo nodo del grafo y luego recorrerlo hacia afuera, analizando los patrones de relación en tiempo constante para generar recomendaciones.

Neo4j Graph Data Science Library

La biblioteca Neo4j Graph Data Science (GDS) proporciona algoritmos de grafos comunes eficientemente implementados y expuestos como procedimientos Cypher. Además, GDS incluye machine learning pipelines para entrenar modelos predictivos supervisados para resolver problemas de grafos.

Los algoritmos de grafos se utilizan para calcular métricas, nodos o relaciones. Pueden proporcionar información sobre entidades relevantes en el grafo (centralities, ranking), o estructuras inherentes como comunidades (community-detection, graph-partitioning, clustering). Muchos algoritmos de grafos utilizan enfoques iterativos que suelen recorrer el grafo para el cálculo utilizando caminos aleatorios, búsquedas de amplitud o profundidad, o coincidencia de patrones. Debido al crecimiento exponencial de los posibles caminos con el aumento de la distancia, muchos de los enfoques también tienen una alta complejidad algorítmica. Afortunadamente, existen algoritmos optimizados que utilizan ciertas estructuras del grafo, memorizan partes ya exploradas y paralelizan las operaciones.

Para ejecutar los algoritmos de la forma más eficiente posible, GDS utiliza un formato de grafo especializado para representar los datos de los grafos. Para esto, es necesario cargar los datos de los grafos desde la base de datos Neo4j en un catálogo

de grafos en memoria. La cantidad de datos cargados puede controlarse mediante las llamadas proyecciones de grafos, que también permiten, por ejemplo, filtrar nodos por etiquetas y tipos de relación, entre otras opciones. En general, existen cuatro modos de ejecución de algoritmos:

- stream: retorna el resultado del algoritmo como un stream de registros.
- stats: retorna un único registro de estadísticas resumidas, pero no escribe en la base de datos Neo4j.
- mutate: escribe los resultados del algoritmo en el grafo proyectado y retorna un único registro de estadísticas resumidas.
- write: escribe los resultados del algoritmo en la base de datos Neo4j y retorna un único registro de estadísticas resumidas.

4.6. Conclusión

Hemos visto que existen diferentes técnicas para generar recomendaciones. Machine Learning (ML) ha revolucionado la forma en que las computadoras interpretan y actúan sobre los datos, permitiendo aprendizajes sin una codificación explícita. A través de algoritmos variados, adaptables a la esencia del problema, ML se ha transformado en un mecanismo esencial en la predicción de intereses de usuario, utilizable en sistemas de recomendación. No obstante, presenta algunos desafíos, como la dependencia de grandes conjuntos de datos y necesidad de reentrenamiento para mantener los modelos actualizados, los cuales necesitan ser abordados de forma efectiva. Neo4j emerge como una posible solución. Esta base de datos, con su enfoque nativo en grafos y estructura NoSQL, optimiza la manipulación de datos interconectados, y junto con su biblioteca Neo4j Graph Data Science (GDS), une algoritmos de grafos con pipelines de ML. Esto amplía las capacidades de análisis y recomendación, proporcionando una plataforma robusta para enfrentar y reducir algunos desafíos propios de ML.

Capítulo 5

Estrategia General

En este capítulo, se detalla la estrategia general adoptada para el diseño e implementación del algoritmo de recomendación basado en contenido para el sistema Ágora. Dicho algoritmo tiene como objetivo recomendar proyectos a los usuarios en función de sus intereses individuales. Para facilitar la comprensión del enfoque propuesto, se describe el desarrollo de cada uno de los componentes que conforman un sistema de recomendación basado en contenido. Estos componentes son *Content Analyzer*, *Profile Learner*, y *Filtering Component*.

5.1. Matemática y Algoritmos Aplicados

Previo a explicar la solución del algoritmo de recomendación desarrollado, es importante tener en mente conceptos sobre algunos algoritmos y funciones matemáticas aplicados. A continuación se explica la similitud Jaccard, normalización mínimo-máximo, producto de un escalar por un vector, producto de matrices y el algoritmo KNN.

5.1.1. Similitud Jaccard

La similitud o coeficiente Jaccard es una estadística utilizada para conocer la similitud entre conjuntos de datos. La medida hace hincapié en la similitud entre conjuntos finitos, y se define formalmente como el tamaño de la intersección dividido por el tamaño de la unión de los conjuntos. El resultado de la división es un valor comprendido entre 0 y 1, donde 0 indica que los conjuntos son totalmente diferentes y 1 indica que los conjuntos son idénticos.

La fórmula es la siguiente:

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

Ejemplo - Similitud 0

Se consideran dos vectores A y B distintos, es decir, su intersección es igual a 0.

$$A = [1, 2], B = [3, 4]$$

$$Jaccard(A, B) = \frac{|[1, 2] \cap [3, 4]|}{|[1, 2] \cup [3, 4]|} = \frac{0}{4} = 0$$

Ejemplo - Similitud 1

Se consideran dos vectores A y B idénticos.

$$A = [1, 2], B = [1, 2]$$

$$Jaccard(A, B) = \frac{|[1, 2] \cap [1, 2]|}{|[1, 2] \cup [1, 2]|} = \frac{2}{2} = 1$$

Ejemplo - Similitud entre 0 y 1

Se consideran dos vectores A y B que comparten algunas características.

$$A = [1, 2, 3, 5], B = [2, 3, 6]$$

$$Jaccard(A, B) = \frac{|[1, 2, 3, 5] \cap [2, 3, 6]|}{|[1, 2, 3, 5] \cup [2, 3, 6]|} = \frac{2}{5} = 0{,}4$$

5.1.2. Normalización Mínimo-Máximo

La normalización mínimo-máximo realiza una transformación lineal sobre los datos originales. Esta técnica obtiene todos los datos escalados en el rango [0, 1]. Es una técnica utilizada en el modelado y procesamiento de información, en el que los

datos que utilicemos tengan escalas o precisión variables. Por ejemplo, considerando un conjunto de datos que tiene dos características (salario y años trabajados). Ambas características tendrán grandes diferencias entre sus valores. Por ejemplo, el salario puede ser mil veces mayor que los años trabajados. La normalización asegura que todos los datos estén en el mismo rango.

La fórmula que se emplea para normalizar los datos es la siguiente:

$$x' = \frac{x - min(x)}{max(x) - min(x)}$$

Ejemplo

Dado un conjunto de datos como el que se muestra a continuación:

Salario	**Años trabajados**
10000	2
20000	4
50000	5

Cuadro 5.1: Ejemplo de normalización mínimo-máximo

Se observa una gran diferencia entre los valores de las características descritas. Al normalizar cada columna, se obtiene la siguiente tabla:

Salario	**Años trabajados**
0	0
0,25	0,67
1	1

Cuadro 5.2: Tabla normalizada

En el ejemplo presentado, se destaca la importancia de la normalización en el procesamiento de datos. Esta técnica permite transformar valores de diferentes magnitudes a una escala común, facilitando así el análisis y comparación entre ellos. La normalización no solo mejora la interpretación de los datos, sino que también puede optimizar el rendimiento de muchos algoritmos de aprendizaje automático.

5.1.3. Producto de un Escalar por un Vector

La multiplicación de un escalar k por un vector u, resulta en un vector con las siguientes características:

- el vector resultante posee igual dirección que el vector u.
- si k es positivo, el vector resultante posee el mismo sentido que el vector u.
- si k es negativo, el vector resultante posee el sentido contrario al vector u.

Las componentes del vector resultante se obtienen multiplicando el escalar k, por cada una de las componentes del vector u.

$$u = (x, y)$$
$$k \cdot u = k \cdot (x, y) = (k \cdot x, k \cdot y)$$

Ejemplo

Supongamos los siguientes datos:

$$u = (4, 5)$$
$$k = 8$$

El producto del escalar k por el vector u es:

$$k \cdot u = 8 \cdot (4, 5) = (8 \cdot 4, 8 \cdot 5) = (32, 40)$$

5.1.4. Producto de Matrices

Definición

Dadas dos matrices A y B, tales que $A \in \mathbb{R}^{mxn}$ y $B \in \mathbb{R}^{nxp}$, la multiplicación de A por B es una matriz con m filas y p columnas, cuya (i, j)-ésima entrada es: $\sum_{r=1}^{n} a_{ir} \cdot b_{rj}$. Como resultado, obtenemos una matriz C^{mxp}.

Ejemplo

Se consideran dos matrices:

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

$$B = \begin{pmatrix} 5 & 2 \\ 7 & 1 \\ 3 & 4 \end{pmatrix}$$

El producto $A \cdot B$ da como resultado la siguiente matriz:

$$C = \begin{pmatrix} 28, 16 \\ 73, 37 \end{pmatrix}$$

5.1.5. Algoritmo KNN

La siguiente información sobre el algoritmo K-Nearest Neighbors (KNN) y sus aplicaciones ha sido extraída de la documentación proporcionada por IBM [7].

El algoritmo K-Nearest Neighbors (KNN) es un algoritmo de machine learning supervisado, no paramétrico, que utiliza el cálculo de distancias para realizar clasificaciones o predicciones sobre la agrupación de un punto de datos individual. Si bien se puede utilizar para problemas de regresión o clasificación, normalmente es utilizado como algoritmo de clasificación partiendo de la base de que se pueden encontrar puntos similares cerca unos de otros. Para los problemas de clasificación, se asigna una etiqueta de clase sobre la base de un voto mayoritario. Los problemas de regresión utilizan un concepto similar al de los problemas de clasificación, pero en este caso se toma el promedio de los K vecinos más cercanos para hacer una predicción sobre una clasificación. La principal distinción en este caso es que la clasificación se utiliza para valores discretos, mientras que la regresión se utiliza con valores continuos. Sin embargo, antes de poder realizar una clasificación, hay que definir la distancia.

Definición del Valor de K

El valor K en el algoritmo KNN define cuántos vecinos se comprobarán para determinar la clasificación de un punto de consulta específico. Por ejemplo, si K = 1, a la instancia se le asignará la misma clase que su único vecino más cercano. Definir el valor de K puede ser un acto de equilibrio, ya que diferentes valores pueden llevar

a un ajuste excesivo o insuficiente. Los valores más bajos de K pueden tener una varianza alta, pero un sesgo bajo, y los valores más grandes de K pueden llevar a un sesgo alto y una varianza más baja. La elección del valor de K dependerá de los datos de entrada, ya que los datos con más valores atípicos probablemente funcionarán mejor con valores más altos de K. En general, se recomienda tener un número impar para K a fin de evitar empates en la clasificación, y las técnicas de validación cruzada pueden ayudarle a elegir la K óptima para su conjunto de datos.

Aplicaciones de KNN

El algoritmo KNN se ha utilizado en una variedad de aplicaciones, principalmente en la clasificación. Algunos ejemplos son:

- Preprocesamiento de datos: los conjuntos de datos suelen tener valores faltantes, pero el algoritmo KNN puede estimar esos valores en un proceso conocido como imputación de datos perdidos.
- Motores de recomendación: el algoritmo KNN se ha utilizado para ofrecer recomendaciones automáticas a los usuarios a partir de los datos del flujo de clicks de los sitios web.
- Reconocimiento de patrones: KNN ha ayudado a identificar patrones, como en la clasificación de textos y dígito, especialmente útil para identificar números escritos a mano que se pueden encontrar en formularios o sobres de correo.

Ventajas

- Fácil de implementar: la simplicidad y precisión del algoritmo, hace que sea uno de los primeros clasificadores que aprenderá un nuevo científico de datos.
- Se adapta fácilmente: a medida que se añaden nuevas muestras de entrenamiento, el algoritmo se ajusta para tener en cuenta los nuevos datos, ya que todos los datos de entrenamiento se almacenan en la memoria.
- Pocos parámetros: KNN solo requiere un valor para K y una métrica de distancia.

Desventajas

- Poca escalabilidad: KNN es un algoritmo lazy, por lo que ocupa más memoria y almacenamiento de datos en comparación con otros clasificadores. Esto puede ser costoso tanto desde el punto de vista del tiempo como de memoria.
- Maldición de dimensionalidad: KNN no funciona bien con datos de entrada de alta dimensión. Esto se conoce a veces como el fenómeno del pico, en el que después de que el algoritmo alcanza el número óptimo de características, las características adicionales aumentan la cantidad de errores de clasificación, especialmente cuando el tamaño de la muestra es menor.
- Propenso al sobre ajuste: debido a la ”maldición de dimensionalidad”, el KNN también es más propenso al sobre ajuste. Aunque las técnicas de selección de características y de reducción de la dimensionalidad se aprovechan para evitar que esto ocurra, el valor de K también puede afectar al comportamiento del modelo. Los valores más bajos de K pueden sobre ajustar los datos, mientras que los valores más altos de K tienden a suavizar los valores de predicción, ya que se promedian los valores sobre un área mayor, o vecindad. Sin embargo, si el valor de K es demasiado alto, puede no ajustarse a los datos.

5.2. Sistema de Recomendación de Proyectos a Usuarios para Ágora

En esta sección se muestra cómo se construye y qué información contiene cada una de las partes del sistema de recomendación de proyectos a usuarios. Para la construcción del sistema de recomendación se utiliza el interés del usuario sobre los proyectos en los que participó, con el objetivo de predecir el interés en proyectos similares. Una vez predicho este interés, se recomiendan aquellos proyectos que presenten un nivel de interés mayor.

5.2.1. Content Analyzer

En este tipo de recomendación, los ítems a recomendar se representan por los proyectos de ciencia ciudadana. Para construir el perfil del proyecto, se utilizan las categorías de ciencia ciudadana a las que pertenece y los métodos de recolección de información del protocolo (explicado en la sección 3.2). Siendo N el número total

de categorías de ciencia ciudadana y M el número total de métodos de recolección de información, el perfil de cada proyecto se define en un vector de módulo $N + M$. Este vector contiene, para cada característica, un valor que indica si el proyecto está relacionado o no. Para esto, se emplea la codificación *One Hot Encoding*[18], cuya estrategia crea una columna para cada valor distinto que exista en la característica que se está codificando (categoría de ciencia ciudadana o método de recolección) y, para cada registro (proyecto), se marca con un 1 la columna a la que pertenezca dicho registro y se dejan las demás con 0. Esto resulta en un vector de 0 y 1, donde 0 indica que la categoría o método de recolección no aplica al proyecto, y 1 indica que la categoría o método de recolección aplica al proyecto. Si se tienen múltiples proyectos, la información del perfil de estos se representa por una matriz del orden NxM, donde N es el número de proyectos y M resulta de la suma del número de categorías de ciencia ciudadana y del número de métodos de recolección en el sistema. Dicha matriz la podemos representar como se describe en la figura 5.3.

Proyecto	**Categoría 1**	**Categoría 2**	**Categoría N**	**Método 1**	**Método 2**	**Método N**
Proyecto 1	0	1	0	1	0	1
Proyecto 2	0	1	0	1	0	1
Proyecto N	0	1	0	1	0	1

Cuadro 5.3: Perfil de los proyectos

Las categorías de ciencia ciudadana y los métodos de recolección se alojan, en conjunto, en un mismo vector para cada proyecto, dado que se considera que tienen el mismo nivel de importancia al momento de determinar el perfil del mismo. Esto se debe a la necesidad de determinar si las categorías y los métodos de recolección que aplican actualmente resultarán interesantes para un usuario según los proyectos en los que ha participado. Esta estructura de perfil para los proyectos también se emplea para determinar el grado de similitud entre proyectos.

Cálculo de Similitud entre los Proyectos

La similitud entre dos proyectos se calcula basándose en las categorías de ciencia ciudadana y métodos de recolección de información del protocolo activo que poseen actualmente. Esta similitud se determina utilizando el algoritmo KNN. La ejecución de este algoritmo proporciona, para cada proyecto P en el sistema, los K proyectos más similares, utilizando una medida de similitud específica. Las entradas

para calcular la medida de similitud entre dos proyectos son propiedades, de tipo array, que contienen cada una de las categorías y métodos de recolección de los mismos. La medida de similitud empleada es la similitud Jaccard, ya que determina cuán similares son los conjuntos de características que describen a los proyectos en cuestión. El resultado de la similitud es un valor entre 0 y 1, donde 0 indica que son totalmente distintos y 1 indica que son totalmente iguales. Lo ideal es tener las similitudes pre calculadas, relacionando cada proyecto con sus K proyectos más similares, para evitar realizar el cálculo en cada recomendación. El algoritmo KNN debe ser ejecutado ante ciertos eventos cómo:

- Ingreso de nuevo proyecto: se obtienen los K ítems similares para el nuevo proyecto y se actualizan los K ítems similares para los demás proyectos, considerando el nuevo proyecto.
- Actualización de categorías de un proyecto: se recalculan los K ítems más similares debido a la modificación de una característica que influye en la similitud entre proyectos.
- Actualización de métodos de recolección o protocolo de un proyecto: se recalcula los K ítems más similares debido a la modificación de una característica que influye en la similitud entre proyectos.

Para determinar qué valor de K utilizar, se realizarán pruebas que comparen distintas recomendaciones utilizando un dataset con las similitudes pre calculadas para todos los proyectos del grafo y ejecutando el algoritmo KNN con diferentes valores de K. Las comparaciones se realizarán en términos de “precision”, “recall” y “F1-score”. Esto será explicado con mayor detalle en el capítulo 7.

5.2.2. Profile Learner

En este tipo de recomendación, el perfil del usuario se representa por cada uno de los usuarios del sistema y su interés en los proyectos. En este contexto, a los usuarios no se les brinda la opción de indicar explícitamente el nivel de interés sobre un proyecto, ya sea con “me gusta”, “no me gusta” o puntajes. Para determinar el interés de un usuario X en un proyecto Y, se utiliza la información de las muestras realizadas por el usuario en el protocolo activo del proyecto. Por lo tanto, el interés se obtiene de manera implícita.

Como se observa en la figura 5.4, el perfil del usuario se define por el número de muestras realizadas en el protocolo activo de cada proyecto en el que se registró

como participante. Esto resulta en un vector de N elementos, siendo N el número de proyectos donde participó el usuario.

	Proyecto 1	**Proyecto 2**	**Proyecto 3**	**Proyecto 4**
Nº muestras	200	10	25	100

Cuadro 5.4: Ejemplo del número de muestras de los usuarios

El vector de la figura 5.4 representa el número de muestras realizadas por un usuario en el protocolo activo de un proyecto. Sin embargo, no refleja si el interés es reciente o si las muestras se llevaron a cabo hace varios años. Podría darse que el usuario haya realizado 200 muestras para el proyecto P1 a lo largo de los últimos 5 años y haya realizado 100 muestras para el proyecto P4 en el último mes. Es apropiado considerar la fecha del último aporte a un proyecto como un factor influyente al definir el perfil del usuario. Por lo tanto, al perfil del usuario se le incorpora una nueva dimensión que indica la fecha de la última muestra, tal como se representa en la figura 5.5.

	N° muestras	**Fecha última muestra**
P1	200	04/03/2018
P2	10	15/05/2022
P3	25	10/04/2021
P4	100	27/09/2022

Cuadro 5.5: Ejemplo del número de muestras y fecha de la última muestra de un usuario en proyectos

En la figura 5.5, se observa el número de muestras realizadas por el usuario en un proyecto y la fecha del último aporte. Sin embargo, no se considera la tasa de actividad del usuario en dicho proyecto. Para esto, resulta apropiado definir una tasa de actividad del usuario en el proyecto, teniendo en cuenta el número total de muestras realizadas entre la fecha del primer y último aporte. La tasa de actividad se obtiene mediante la siguiente fórmula:

$$TA(U, P) = \frac{N}{M}$$

Donde N es el total de muestras realizadas por el usuario U en el proyecto P, M es la diferencia en días entre la fecha del primer y último aporte, y $TA(U, P)$ es el

valor de la tasa de actividad calculado. El resultado obtenido en la tasa de actividad se interpreta cómo ”el número de muestras, por día, aportadas por un participante en un proyecto”. Así, el perfil completo del usuario se representa como se muestra en la figura 5.6:

	N° muestras	**Fecha última muestra**	**Tasa de actividad**
P1	200	04/03/2018	20
P2	10	15/05/2022	2
P3	25	10/04/2021	0.26
P4	100	27/09/2022	50

Cuadro 5.6: Número de muestras, fecha de la última muestra y tasa de actividad de un usuario en proyectos

El perfil de un usuario U_x para un proyecto X_y en el que participó, se construye considerando tres características: cantidad de muestras, fecha del último aporte y tasa de actividad. Estas se representan mediante valores numéricos (cantidad de muestras y tasa de actividad) y fechas (fecha del último aporte). Por lo tanto, y como primer paso, es esencial convertir toda la información a valores numéricos. Para llevar las fechas a valores numéricos, es necesario definir una escala para representar que fecha es más actual y cuál es más lejana en el tiempo. Así, se propone una escala temporal en un rango de 1 a 3, donde 3 representa la fecha más reciente:

- fecha menor a un año comparada con la fecha actual → 3
- fecha entre 1 y 3 años comparada con la fecha actual → 2
- fecha mayor a 3 años comparada con la fecha actual → 1

Al aplicar dicha escala en la tabla definida en la descripción del perfil del usuario de la sección **Profile Learner**, y considerando como fecha actual 28/09/2022, el perfil se visualiza como en la figura 5.7.

	N° muestras	**Fecha última muestra**	**Tasa de actividad**
P1	200	1	20
P2	10	3	2
P3	25	2	0.26
P4	100	3	50

Cuadro 5.7: Fecha de la última muestra utilizando escala numérica

En la tabla 5.7, se observa que el perfil de un usuario U_x para un proyecto P_y, posee más de una caracteristica. Por lo tanto, el nivel de interés de U_x en P_y, está definido por varios criterios influyentes. Para determinar el interés utilizando estos múltiples criterios, es esencial establecer el peso o importancia de cada uno en la decisión. El peso de criterios como el número de muestras, la escala de tiempos y la tasa de actividad se determinará mediante el método AHP (Analytic Hierarchy Process) o PJA (Proceso Analítico Jerárquico). Según Saaty [31], AHP es un método que selecciona alternativas en función de una serie de criterios o variables, normalmente jerarquizados, los cuales suelen entrar en conflicto. En esta estructura jerárquica, el objetivo final se encuentra en el nivel más elevado, y los criterios y sub criterios en los niveles inferiores. Para que el método sea eficaz, es fundamental elegir bien los criterios y sub criterios, los cuales deben estar muy bien definidos, ser relevantes y mutuamente excluyentes (independencia entre ellos).

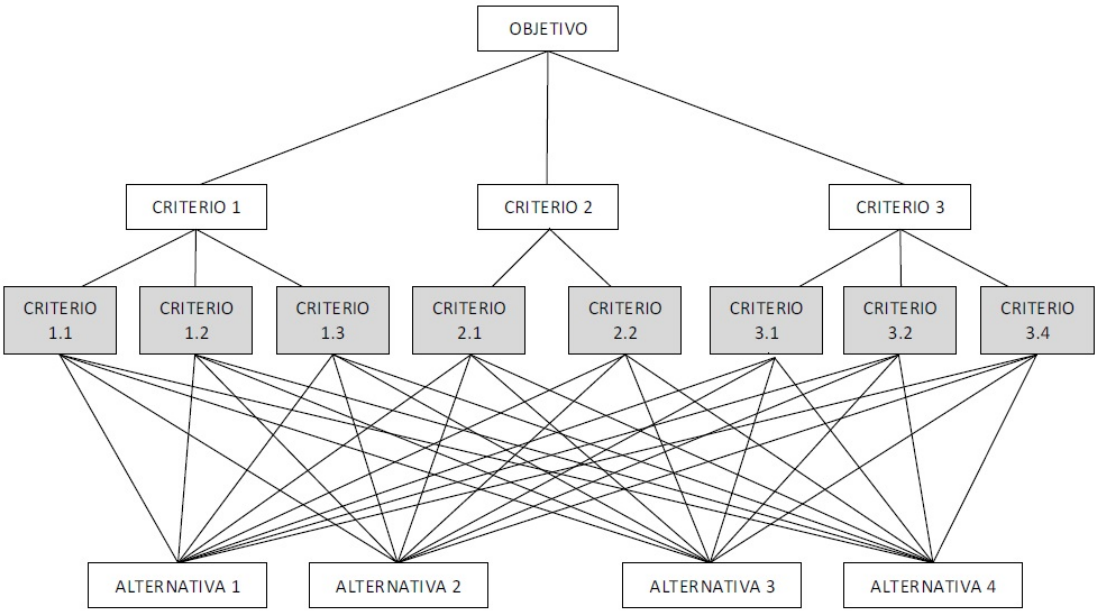

Figura 5.1: Estructura AHP (de [21])

En AHP se utiliza una escala fundamental del 1 al 9, la cual ha demostrado ser satisfactoria en pruebas empíricas realizadas en situaciones reales muy diversas (escala fundamental de comparación por pares). Esta tabla de valores, se utiliza para indicar el nivel de preferencia comparando cada elemento uno contra uno para todas

las combinaciones posibles. Según Saaty [31], la forma más efectiva de concentrar un juicio sobre algo es tomar solo dos elementos y compararlos entre sí respecto a una sola propiedad y dejando de lado todas las demás. La escala de valores, la define con la tabla de la figura 5.8.

Valor	**Definición**
1	Igual importancia
3	Importancia moderada
5	Importancia grande
7	Importancia muy grande
9	Importancia extrema
2, 4, 6 y 8	Valores intermedios entre los anteriores
1/2, 1/3, 1/4, 1/5, 1/6, 1/7, 1/8, 1/9	Valores para la comparación inversa

Cuadro 5.8: Escala fundamental definida por Saaty

Al aplicar AHP a la estructura del perfil del usuario previamente definida, se puede desarrollar la siguiente tabla de pesos (figura 5.9):

	N° muestras	**Fecha última muestra**	**Tasa de actividad**
N° muestras		Fecha última muestra, 5	Tasa de actividad, 3
Fecha última muestra			Tasa de actividad, 3
Tasa de actividad			

Cuadro 5.9: Tabla de comparación por pares

Como resultado, se obtiene una matriz de comparación por pares (MCP) o Pairwise Comparison Matrix (PCM). Estas se utilizan para calcular las prioridades relativas de los criterios o las alternativas, y son componentes integrales de herramientas de toma de decisiones ampliamente aplicadas en, por ej. AHP. En cada celda de intersección, se define cuál es la opción preferida y un punteo que va desde 1 a 9, según

definida por Saaty en la escala fundamental. A la tabla resultante, se la interpreta de la siguiente manera:

- Entre el número de muestras y la fecha de la última muestra, esta última posee una importancia grande sobre la primera.
- Entre el número de muestras y la tasa de actividad, esta última posee una importancia moderada sobre la primera.
- Entre la fecha de la última muestra y la tasa de actividad, esta última posee una importancia moderada sobre la primera.

El objetivo es obtener el peso que posee cada una de las características evaluadas en un rango de valores definido, en este caso entre 0 y 1. La matriz se tabula para totalizar los puntos obtenidos cada vez que una de las opciones fue elegida. Los resultados se presentan en forma de porcentaje (figura 5.10):

Opción	**Punteo**	**Porcentaje**
N° muestras	0	0.0 %
Fecha última muestra	5	45.45 %
Tasa de actividad	6	54.55 %
Total	**11**	**100 %**

Cuadro 5.10: Tabla de comparación por pares tabulada

Finalmente, se puede decir que la tasa de actividad del usuario en un proyecto posee mayor importancia que el tiempo transcurrido desde la actualidad hasta la fecha de la última muestra, y la fecha de la última muestra tiene mayor importancia sobre el número de muestras realizadas en total.

El perfil de un usuario es único para cada proyecto en el que participa. Así, este tendrá tantos perfiles como proyectos en los que haya participado. Es importante que el perfil de un usuario U en un proyecto P se actualice cada vez que U realiza un nuevo aporte en el protocolo activo de P.

5.2.3. Filtering Component

Una vez definidos los perfiles de los proyectos (a través del Content Analyzer 5.2.1) y de los usuarios (a través del Profile Learner 5.2.2), es necesario realizar el filtrado de la información para conocer el estado actual de ambos perfiles y, así, efectuar una recomendación. El filtrado de la información se realiza siguiendo los pasos que se detallan a continuación.

Paso 1: Obtener Proyectos Similares

Para recomendar proyectos a un usuario U, es necesario diferenciar entre los proyectos en los que ya ha participado, denominados X, y aquellos en los que aún no ha tenido participación. En este paso, a partir de los proyectos en X, se obtienen aquellos similares en los que aún no ha participado, denominados Y. Y es el conjunto de proyectos a los cuales se les predecirá el nivel de interés probable que el usuario podría tener, basándose en el interés actual calculado para los proyectos en X (obtenido en el paso 3).

Paso 2: Convertir las Fechas del Último Aporte a Valor Numérico

En este paso, se reemplazan las fechas del último aporte para cada proyecto en X por los valores de la escala definida en la figura 5.7. De esta forma, cada perfil del usuario U en X, queda representado con valores numéricos.

Paso 3: Obtener el Interés Implícito del Usuario en Cada Proyecto en X

Para determinar el porcentaje de interés implícito del usuario U en cada proyecto en X, se sigue una serie de cálculos hasta llegar al valor final. Estos pasos son los siguientes:

1. Normalización del perfil del usuario: es necesario normalizar el perfil del usuario U en cada proyecto en X, dado que se definen tres características numéricas para cada proyecto en el que ha participado (número de muestras, escala de tiempo y tasa de actividad), con rangos de valores distintos. Se utiliza la normalización de mínimo-máximo entre 0 y 1 para normalizar cada columna de valores, correspondiente a cada característica mencionada. La fórmula empleada es la siguiente:

$$n_{ij} = \frac{r_{ij} - r_{min}}{r_{max} - r_{min}}$$

   Donde r_{min} representa el valor mínimo de la columna a normalizar, r_{max} indica el valor máximo de dicha columna, r_{ij} es el valor que se busca normalizar y n_{ij} es el resultado de la normalización. En este contexto, cada columna a normalizar corresponde a cada una de las características que describen el perfil del usuario (N.º Muestras, Fecha última muestra y Tasa de actividad). Puede darse el caso de que, para una determinada característica, todos los valores sean iguales. Por ejemplo, que un usuario haya registrado 1 observación en

todos los proyectos. Al aplicar la fórmula mencionada anteriormente, la normalización daría un error debido a una división por 0. Para evitar esto, se define una función partida, con un valor de normalización intermedio que contempla dicho caso:

$$n_{ij} = \begin{cases} \frac{r_{ij} - r_{min}}{r_{max} - r_{min}} & si \quad r_{min} \neq r_{max} \\ \\ 0,5 & si \quad r_{min} = r_{max} \end{cases}$$

2. Producto de vector por escalar entre cada columna normalizada y el peso calculado para la misma (porcentaje obtenido en la tabla de comparación por pares 5.10): este proceso tiene como objetivo ajustar el peso (o importancia) de la característica con el valor normalizado. El producto de un escalar por un vector da por resultado otro vector, con la misma dirección que el primero:

$$V = (x, y)$$

$$k \cdot V = k \cdot (x, y) = (x \cdot k, y \cdot k)$$

Donde V representa el vector, (x, y) son los valores del vector y k es el escalar. En este contexto, el vector V corresponde a cada columna que describe al perfil del usuario y el escalar k es el peso de cada una (ver figura 5.10).

3. Cálculo de la suma total de cada fila correspondiente a cada proyecto para obtener el interés general del usuario en cada proyecto en X (valores entre 0 y 1): este proceso resulta en un vector de longitud M, donde M representa el número de proyectos en los que el usuario ha participado. La fórmula utilizada para este cálculo es:

$$Int(Pro_x) = \sum_{i}^{n} Car(i)$$

Donde Pro_x representa el proyecto al que se le determinará el interés del usuario, n indica el número de características que describen el perfil del usuario para Pro_x, $Car(i)$ es el valor de la característica i para el proyecto Pro_x, y $Int(Pro_x)$ corresponde al porcentaje de interés determinado para el proyecto Pro_x.

Paso 4: Obtener el Interés General a Nivel Característica

En este paso, se busca determinar el porcentaje de interés del usuario U en cada una de las características que describen a los proyectos a nivel global, es decir, utilizando el conjunto completo de categorías de ciencia ciudadana y de métodos

de recolección de información disponibles en la base de datos. Para determinar el porcentaje de interés de cada característica, se siguen los siguientes pasos:

1. Producto de vector por escalar entre el perfil de booleanos de cada proyecto en X y el porcentaje de interés obtenido, para X, en el punto anterior: la fórmula es:

$$IntCar(P_x) = Per(P_x) \cdot I(P_x)$$

   Donde $Per(P_x)$ representa el vector de booleanos del proyecto x, $I(P_x)$ indica el porcentaje de interés para el proyecto x calculado en el punto anterior, y $IntCar(P_x)$ es el vector resultante que contiene el interés por característica en el proyecto P_x.

2. Cálculo del promedio de interés general para cada característica, utilizando el número total de proyectos: la fórmula para el cálculo es:

$$Prom(Car_x) = \frac{\sum_i^n Car(x)_i}{n}$$

   Donde n representa el total de proyectos en los que ha participado el usuario, $Car(x)_i$ indica el valor de interés de la característica x para el proyecto i, y $Prom(Car_x)$ es el promedio de interés calculado para la característica Car_x.

Paso 5: Obtener el Interés Estimado Para Cada Proyecto Obtenido en el Punto 1

Una vez obtenido el promedio de interés del usuario U en cada una de las características que describen a los proyectos, este paso tiene como objetivo predecir el interes probable de U en los proyectos en Y. Para determinar el porcentaje de interés probable para cada proyecto en Y, se realizan los siguientes pasos:

1. Se realiza el producto de vector por escalar utilizando el porcentaje de interés de cada característica, obtenido en el punto 4, y la columna correspondiente a dicha característica en los proyectos identificados en el punto 1. La fórmula empleada es:

$$CarPerRec(x) = Per(Car_x) \cdot I(Car_x)$$

   Donde $Per(Car_x)$ representa un vector de longitud N (donde N es el total de proyectos obtenidos en el punto 1) y cada celda de este vector corresponde a la codificación obtenida por One Hot Encoding para cada característica x en cada proyecto, $I(Car_x)$ es el interés para la característica x calculado en el punto 4, y $CarPerRec(x)$ es el vector resultado para la característica x.

2. Se calcula el promedio de interés estimado para cada proyecto. La fórmula empleada es:

$$I(P_x) = \frac{\sum_i^n I(C_i)}{n}$$

Donde P_x representa el proyecto al cual se le calcula el interés, n es el número total de características que aplican al perfil del proyecto (es decir, aquellas donde el booleano es un 1). Por su parte, $I(C_i)$ refleja el interés determinado para la característica i en P_x, y $I(P_x)$ representa el porcentaje de interés estimado obtenido para el proyecto x.

Paso 6: Ordenamiento de los Resultados

Como último paso, los resultados se ordenan por el porcentaje de interés en orden descendente, permitiendo presentar los proyectos en Y desde el de mayor a menor interés predicho.

5.2.4. Ejemplo Aplicado

En esta sección, se presenta un ejemplo de recomendación de proyectos a usuarios. Se utilizan 10 proyectos de muestra, los cuales se supone que poseen los siguientes perfiles (tabla 5.11), tras haber aplicado la técnica One Hot Encoding (*Cat x* representa a una categoría de ciencia ciudadana y *Met x* representa a un metodo de recolección de información):

	Cat 1	**Cat 2**	**Cat 3**	**Met 1**	**Met 2**	**Met 3**
P1	1	0	1	0	1	0
P2	1	1	1	0	1	0
P3	0	1	1	1	1	0
P4	1	1	0	1	0	1
P5	0	0	1	0	1	1
P6	1	0	0	0	0	1
P7	0	1	0	1	0	0
P8	1	1	0	1	0	1
P9	0	0	0	0	0	0
P10	0	1	0	0	0	0

Cuadro 5.11: Perfiles de proyectos

De los proyectos descritos, se supone que el usuario ha participado en los proyectos P1 a P4, y se calculará el porcentaje de interés para los proyectos P5 a P10. La fecha de referencia para la recomendación es el 28/09/2022. A continuación, se presenta el perfil actual del usuario para cada proyecto en el que ha participado (tabla 5.12):

Proyecto	**Nro. muestras**	**Fecha última muestra**	**Tasa de actividad**
P1	200	04/03/2018	20
P2	10	15/05/2022	2
P3	25	10/04/2021	0.26
P4	100	27/09/2022	50

Cuadro 5.12: Perfil actual del usuario

Paso 1

En este paso, se aplica el algoritmo KNN para encontrar los proyectos más similares, utilizando la similitud de Jaccard, a aquellos en los que ha participado el usuario. Se supone que tras aplicar KNN-Jaccard utilizando los proyectos en los que ha participado (P1 a P4) sobre el resto de los proyectos restantes (P5 a P10), se obtiene la siguiente matriz de perfiles de proyectos similares (tabla 5.13):

	Cat 1	**Cat 2**	**Cat 3**	**Met 1**	**Met 2**	**Met 3**
P5	0	0	1	0	1	1
P6	1	0	0	0	0	1
P7	0	1	0	1	0	0
P8	1	1	0	1	0	1

Cuadro 5.13: Perfiles de proyectos similares

Estos proyectos serían, en principio, los recomendados para el usuario. A partir de este punto, se realizarán los cálculos necesarios para determinar el interés probable y ordenarlos de mayor a menor interés.

Paso 2

En este paso se realiza el reemplazo, en el perfil del usuario, de cada una de las fechas de la última muestra, con el valor definido en la escala de rango 1 a 3, de la sección Profile Learner (ver 5.2.2). Al aplicar dicha escala, el perfil del usuario queda de la siguiente forma (tabla 5.14):

Proyecto	**Nro. muestras**	**Fecha última muestra**	**Tasa de actividad**
P1	200	1	20
P2	10	3	2
P3	25	2	0.26
P4	100	3	50

Cuadro 5.14: Perfil actual del usuario con escala aplicada

Paso 3

Este paso se compone de una serie de operaciones matemáticas que resultarán en el interés actual implícito del usuario en los proyectos en los que ha participado.

- **Normalización mínimo-máximo sobre el perfil del usuario**

  La técnica de normalización se aplica a cada columna que describe una característica del perfil del usuario. En esta etapa, se define una función partida en la que se utiliza la fórmula de normalización para columnas con valores diferentes y un valor fijo, teniendo en cuenta el caso en el que todos los valores de una misma columna sean iguales. Para llevar a cabo esta técnica, se deben identificar el valor máximo y mínimo de cada columna y aplicar la fórmula a cada celda:

$$n_{ij} = \begin{cases} \frac{r_{ij} - r_{min}}{r_{max} - r_{min}} & si \quad r_{min} \neq r_{max} \\ 0,5 & si \quad r_{min} = r_{max} \end{cases}$$

  Una vez identificados los valores mínimos y máximos de cada columna, se obtiene que:

  - El valor mínimo del número de muestras es 10 y el valor máximo es 200.

- El valor mínimo de la fecha de la última muestra es 1 y el valor máximo es 3.
- El valor mínimo de la tasa de actividad es 0.26 y el valor máximo es 50.

Como se puede observar, los valores mínimos y máximos varían en todas las columnas. Por lo tanto, se aplica la función de normalización a cada columna, y los resultados se presentan a continuación (consulte la tabla 5.15):

Proyecto	**Nro. muestras**	**Fecha última muestra**	**Tasa de actividad**
P1	$\frac{200-10}{200-10}$	$\frac{1-1}{3-1}$	$\frac{20-0,26}{50-0,26}$
P2	$\frac{10-10}{200-10}$	$\frac{3-1}{3-1}$	$\frac{2-0,26}{50-0,26}$
P3	$\frac{25-10}{200-10}$	$\frac{2-1}{3-1}$	$\frac{0,26-0,26}{50-0,26}$
P4	$\frac{100-10}{200-10}$	$\frac{3-1}{3-1}$	$\frac{50-0,26}{50-0,26}$

Cuadro 5.15: Perfil actual del usuario con fórmula de normalización Max-Min aplicada

Finalmente, la tabla normalizada queda de la siguiente forma (tabla 5.16):

Proyecto	**Nro. muestras**	**Fecha última muestra**	**Tasa de actividad**
P1	1	0	0,4
P2	0	1	0,03
P3	0,08	0,5	0
P4	0,47	1	1

Cuadro 5.16: Resultado de la fórmula de normalización

- **Producto de vector por escalar entre cada columna normalizada y el peso calculado**

  En este punto, se realiza un ajuste entre el peso (o porcentaje de importancia) de la característica obtenido en luego de aplicar AHP y el valor normalizado. Recordemos que los porcentajes de importancia obtenidos para cada característica son los siguientes:

  - Para el número de muestras es 0
  - Para la fecha de la última muestra es 0,45

- Para la tasa de actividad es 0,55

Se aplican los pesos al perfil del usuario, lo que resulta en la siguiente tabla (tabla 5.17 y 5.18):

Proyecto	**Nro. muestras**	**Fecha última muestra**	**Tasa de actividad**
P1	$1 * 0$	$0 * 0,45$	$0,4 * 0,55$
P2	$0 * 0$	$1 * 0,45$	$0,03 * 0,55$
P3	$0,08 * 0$	$0,5 * 0,45$	$0 * 0,55$
P4	$0,47 * 0$	$1 * 0,45$	$1 * 0,55$

Cuadro 5.17: Producto entre pesos y perfil del usuario

Proyecto	**Nro. muestras**	**Fecha última muestra**	**Tasa de actividad**
P1	0	0	0,22
P2	0	0,45	0,02
P3	0	0,23	0
P4	0	0,45	0,55

Cuadro 5.18: Resultado del producto entre pesos y perfil del usuario

- **Suma total de cada fila**

  Finalmente, se realiza la suma total de los valores obtenidos para cada proyecto con el fin de obtener el interés general actual del usuario en cada uno.

  - Interés en el proyecto 1: $0 + 0 + 0,22 = 0,22$
  - Interés en el proyecto 2: $0 + 0,45 + 0,02 = 0,47$
  - Interés en el proyecto 3: $0 + 0,23 + 0 = 0,23$
  - Interés en el proyecto 4: $0 + 0,45 + 0,55 = 1$

Paso 4

En este paso, se busca obtener el porcentaje de interés general a nivel características que describen a los proyectos. Para obtenerlo, se parte de dos tablas (tabla 5.19 y 5.20):

Proyecto	Interes
P1	0,22
P2	0,47
P3	0,23
P4	1

Cuadro 5.19: Interés actual del usuario en los proyectos que participó

	Cat 1	Cat 2	Cat 3	Met 1	Met 2	Met 3
P1	1	0	1	0	1	0
P2	1	1	1	0	1	0
P3	0	1	1	1	1	0
P4	1	1	0	1	0	1

Cuadro 5.20: Perfiles de los proyectos en los que participó

Para calcular el interés a nivel característica, se deben seguir dos pasos, detallados a continuación:

- **Producto del perfil de cada proyecto, por el porcentaje de interés obtenido para cada uno**

	Cat 1	Cat 2	Cat 3	Met 1	Met 2	Met 3
P1	$1 * 0,22$	$0 * 0,22$	$1 * 0,22$	$0 * 0,22$	$1 * 0,22$	$0 * 0,22$
P2	$1 * 0,47$	$1 * 0,47$	$1 * 0,47$	$0 * 0,47$	$1 * 0,47$	$0 * 0,47$
P3	$0 * 0,23$	$1 * 0,23$	$1 * 0,23$	$1 * 0,23$	$1 * 0,23$	$0 * 0,23$
P4	$1 * 1$	$1 * 1$	$0 * 1$	$1 * 1$	$0 * 1$	$1 * 1$

Cuadro 5.21: Producto entre interés de cada proyecto y características

	Cat 1	Cat 2	Cat 3	Met 1	Met 2	Met 3
P1	0,22	0	0,22	0	0,22	0
P2	0,47	0,47	0,47	0	0,47	0
P3	0	0,23	0,23	0,23	0,23	0
P4	1	1	0	1	0	1

Cuadro 5.22: Resultado del producto entre interes de cada proyecto y características

- **Promedio de interés por característica**

  Se calcula el promedio de cada columna para determinar el interés general en cada característica.

  - Cat 1: $\frac{0,22+0,47+0+1}{4} = 0,42$
  - Cat 2: $\frac{0+0,47+0,23+1}{4} = 0,43$
  - Cat 3: $\frac{0,22+0,47+0,23+0}{4} = 0,23$
  - Met 1: $\frac{0+0+0,23+1}{4} = 0,31$
  - Met 2: $\frac{0,22+0,47+0,23+0}{4} = 0,23$
  - Met 3: $\frac{0+0+0+1}{4} = 0,25$

Paso 5

En este paso, se determina el promedio de interés estimado para cada proyecto similar a aquellos en los que se ha participado. Para lograrlo, se siguen dos pasos detallados a continuación. Este cálculo se basa en dos tablas:

Caracteristica	Interés
Cat 1	0,42
Cat 2	0,43
Cat 3	0,23
Met 1	0,31
Met 2	0,23
Met 3	0,25

Cuadro 5.23: Interés a nivel características

	Cat 1	Cat 2	Cat 3	Met 1	Met 2	Met 3
P5	0	0	1	0	1	1
P6	1	0	0	0	0	1
P7	0	1	0	1	0	0
P8	1	1	0	1	0	1

Cuadro 5.24: Perfiles de proyectos similares

Los pasos para determinar el interés estimado se describen a continuación:

- **Producto entre el interés y las características de cada proyecto**

	Cat 1	**Cat 2**	**Cat 3**	**Met 1**	**Met 2**	**Met 3**
P5	$0 * 0,42$	$0 * 0,43$	$1 * 0,23$	$0 * 0,31$	$1 * 0,23$	$1 * 0,25$
P6	$1 * 0,42$	$0 * 0,43$	$0 * 0,23$	$0 * 0,31$	$0 * 0,23$	$1 * 0,25$
P7	$0 * 0,42$	$1 * 0,43$	$0 * 0,23$	$1 * 0,31$	$0 * 0,23$	$0 * 0,25$
P8	$1 * 0,42$	$1 * 0,43$	$0 * 0,23$	$1 * 0,31$	$0 * 0,23$	$1 * 0,25$

Cuadro 5.25: Producto entre interés y características

	Cat 1	**Cat 2**	**Cat 3**	**Met 1**	**Met 2**	**Met 3**
P5	0	0	0,23	0	0,23	0,25
P6	0,42	0	0	0	0	0,25
P7	0	0,43	0	0,31	0	0
P8	0,42	0,43	0	0,31	0	0,25

Cuadro 5.26: Resultado del producto entre interés y características

- **Promedio de interés estimado para cada proyecto**
  - Interés estimado para el proyecto 5: $\frac{0+0+0,23+0+0,23+0,25}{3} = 0,24$
  - Interés estimado para el proyecto 6: $\frac{0,42+0+0+0+0+0,25}{2} = 0,34$
  - Interés estimado para el proyecto 7: $\frac{0+0,43+0+0,31+0+0}{2} = 0,37$
  - Interés estimado para el proyecto 8: $\frac{0,42+0,43+0+0,31+0+0,25}{4} = 0,35$

Paso 6

Como último paso, se ordenan los resultados por el porcentaje de interés en orden descendente, permitiendo identificar los proyectos desde el de mayor a menor interés.

Proyecto	**Interes**
P7	0,37
P8	0,35
P6	0,34
P5	0,24

Cuadro 5.27: Tabla de recomendaciones

Capítulo 6

Diseño e Implementación

En el siguiente capítulo, se definirá la tecnología empleada para desarrollar el sistema de recomendación de Ágora. Además, se describirá el modelado de la base de datos y la integración de Neo4j en la aplicación Ágora desarrollada en Java. Por último, se explicará y detallará cómo se programó el algoritmo de recomendación, definido en el capítulo anterior, utilizando consultas en Neo4j.

6.1. Elección de Tecnología para el Sistema de Recomendación

En la aplicación Ágora, se utiliza actualmente MongoDB como base de datos principal. A pesar de que MongoDB es eficiente en el manejo de datos semiestructurados y posee una alta escalabilidad, presenta limitaciones al gestionar relaciones complejas entre datos, un elemento esencial para sistemas de recomendación. Si bien los algoritmos de Machine Learning (ML) son una opción común para estos sistemas, Neo4j se presenta como una alternativa prometedora. Mientras que ML ha surgido como una herramienta poderosa en el ámbito de la inteligencia artificial, con el propósito de enseñar a las máquinas a manejar y aprender de los datos a través de diversos algoritmos, Neo4j se presenta como una alternativa especializada para sistemas de recomendación. Como se detalló en la sección *Tecnologías para Sistemas de Recomendación*4.5, Neo4j es una base de datos orientada a grafos nativa, diseñada para gestionar y consultar datos en forma de grafos, lo cual es particularmente útil para sistemas que requieren comprender relaciones entre entidades. A diferencia de ML, donde adaptar modelos a nuevos requisitos puede ser un proceso tedioso que requiere reajustes y reentrenamientos, Neo4j ofrece una mayor flexibilidad. Su falta

de esquema definido permite reestructurar el grafo según las necesidades cambiantes y su lenguaje de consulta, Cypher, facilita la adaptación y edición de patrones y algoritmos sin necesidad de reentrenamiento. Además, Neo4j se destaca en la generación de recomendaciones en tiempo real, aprovechando su modelo de grafos para soportar información rica y contextualizada, permitiendo realizar recomendaciones relevantes y en tiempo real. Gracias a su motor de grafos nativo y la adyacencia libre de índices, Neo4j puede recorrer el grafo en tiempo constante, evitando operaciones costosas. Finalmente, Neo4j ofrece la librería Neo4j Graph Data Science (GDS) que proporciona algoritmos de grafos eficientemente implementados y pipelines de machine learning, facilitando aún más la implementación de sistemas de recomendación. En conclusión, la capacidad de Neo4j para entender y analizar relaciones complejas en tiempo real lo posiciona como una opción en comparación con los enfoques tradicionales de ML, especialmente en contextos donde las relaciones y el contexto son cruciales para hacer recomendaciones precisas y relevantes. Por ello, se propone la adopción de Neo4j para el desarrollo del sistema de recomendación, complementando las capacidades de MongoDB. La propuesta no busca reemplazar a MongoDB, sino establecer una arquitectura híbrida entre Neo4j y MongoDB, enriqueciendo las funcionalidades de la aplicación. Esta estrategia permitirá mantener MongoDB para otros aspectos del sistema, aprovechando simultáneamente las ventajas de Neo4j en el ámbito de las recomendaciones.

6.2. Modelo Propuesto

En esta tesina se implementará únicamente el sistema de recomendación de proyectos a usuarios de manera aislada, es decir, sin integrarlo con la aplicación de Ágora. En el sistema de recomendación propuesto, se sugiere un diseño de base de datos basado en el estándar PPSR-Core. Dicho modelo facilita el desarrollo de un sistema de recomendación inicial y cuenta con propiedades que permitirán su mejora y la incorporación de diferentes tipos de recomendaciones en el futuro. El modelo de datos sugerido para Ágora se fundamenta en el modelo general para proyectos de ciencia ciudadana (figura 6.1).

En el modelo presentado, las entidades se describen de la siguiente forma:

- **User**: define la información correspondiente a un usuario que se almacenará en la base de datos.
- **ProjectScienceType**: define la información correspondiente a una categoría de ciencia ciudadana que se almacenará en la base de datos.

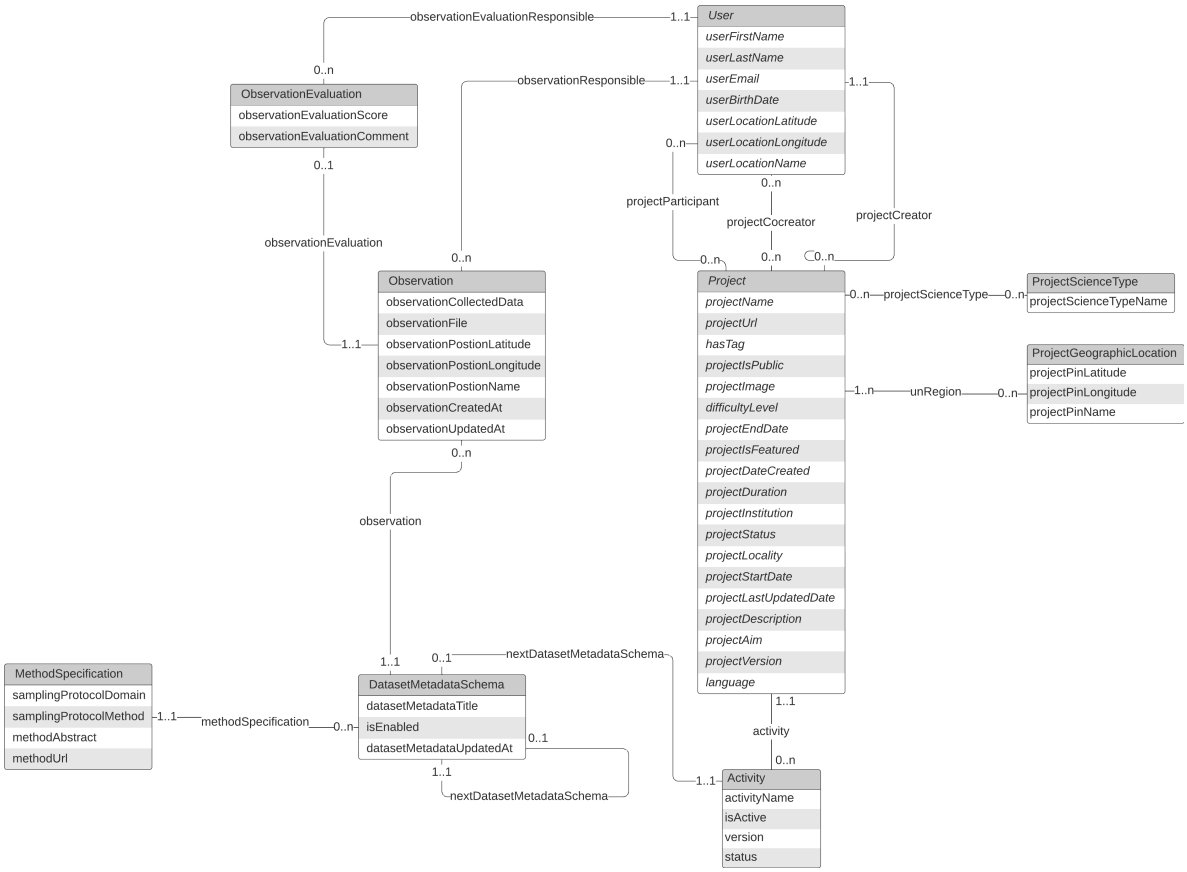

Figura 6.1: Modelo de datos para Ágora, basado en PPSR Core

- **ProjectGeographicLocation**: define la información correspondiente a una ubicación geográfica que se almacenará en la base de datos.
- **Project**: define la información correspondiente a un proyecto que se almacenará en la base de datos. Este se relaciona con los usuarios a través de tres relaciones: *projectCreator* (usuario creador del proyecto), *projectCocreator* (usuario cocreador del proyecto) y *projectParticipant* (participante del proyecto). Cada proyecto puede estar asociado a múltiples categorías (*projectScienceType*), ubicaciones geográficas (*unRegion*) y protocolos (*activity*).
- **Activity**: define la información correspondiente a un protocolo que se almacenará en la base de datos. Este se relaciona con la primera tarea (o paso) del protocolo (*nextDatasetMetadataSchema*).

- **DatasetMetadataSchema**: define la información correspondiente a una tarea que se almacenará en la base de datos. Esta se relaciona con un método de recolección de información (*methodSpecification*), con la siguiente tarea dentro del protocolo (*nextDatasetMetadataSchema*) y con múltiples observaciones (*observation*).
- **MethodSpecification**: define la información correspondiente a un método de recolección de información que se almacenará en la base de datos.
- **Observation**: define la información correspondiente a una observación o aporte que se almacenará en la base de datos. Este se relaciona con el usuario responsable del aporte (*observationResponsible*).
- **ObservationEvaluation**: define la información correspondiente a una evaluación de una observación o aporte que se almacenará en la base de datos. Esta se relaciona con el usuario que evaluó la observación (*observationEvaluationResponsible*).

A partir del modelo desarrollado, se implementó el sistema de recomendación propuesto en la estrategia de la tesina (ver 5.2).

6.3. Implementación

En esta sección, se detallan los pasos necesarios para integrar Neo4j en nuestra aplicación Java. Se abordan aspectos como la integración de dependencias, así como el diseño de modelos y repositorios que cumplen con ciertos criterios y patrones de diseño.

6.3.1. Dependencias

Como primer paso, se instaló el driver y las dependencias necesarias para el mapeo de los modelos escritos en Java a nodos y relaciones en la base de datos. Neo4j ofrece el driver para conectar una aplicación con su base de datos, a través del protocolo Bolt[2]. Para ello, existe una librería denominada «quarkus-neo4j», por lo que en el archivo «pom.xml» se agregó [29]:

```
<dependency>
    <groupId>io.quarkus</groupId>
    <artifactId>quarkus-neo4j</artifactId>
```

```
</dependency>
```

Además, para el mapeo de objetos a nodos y relaciones, se empleó la librería Neo4j-OGM [14]. Dicha librería proporciona funcionalidades como:

- Mapeo de objetos a nodos y relaciones en el grafo.
- Interacción con Neo4j a través de la clase Neo4jSession.
- Rápido escaneo de metadatos en las clases definidas.
- Gestión optimizada de la carga de datos y del seguimiento de los cambios para minimizar las transferencias de datos.
- Proyección de los resultados de las consultas en DTOs.
- Manejo de eventos del ciclo de vida de la persistencia de datos.

Para utilizar dichas librerías, se deben agregar las siguientes dependencias en el «pom.xml»:

```
<dependency>
    <groupId>org.neo4j</groupId>
    <artifactId>neo4j-ogm-bolt-driver</artifactId>
    <version>3.2.23</version>
</dependency>
<dependency>
    <groupId>org.neo4j</groupId>
    <artifactId>neo4j-ogm-bolt-native-types</artifactId>
    <version>3.2.27</version>
</dependency>
```

6.3.2. Modelos

Para comenzar, se llevó a cabo la creación de todas las clases requeridas para cumplir con el modelo planteado en la sección 6.2. Dichas clases, junto con sus atributos y relaciones, para ser persistidas con Neo4j, deben emplear las siguientes anotaciones proporcionadas por Neo4j-OGM:

- **@NodeEntity**: anotación utilizada en una clase para indicar que esta será mapeada a un nodo en la base de datos.

- **@Id** y **@GeneratedValue**: la anotación @Id, se utiliza en un atributo de la clase, para indicar que es el identificador de la misma. @GeneratedValue, por su parte, se utiliza para indicar que el valor de dicho atributo, se generara de forma automática. Combinándolos, identifican y generan automáticamente el identificador único del nodo.
- **@Property** (opcional): anotación utilizada para indicar que un atributo se mapeará como propiedad del nodo.
- **@Relationship**: anotación utilizada para indicar la relación los nodos.

A continuación, se presenta un ejemplo con la clase *Project*:

```
package ar.edu.unlp.info.lifia.agoraprojects.model;

import org.neo4j.ogm.annotation.GeneratedValue;
import org.neo4j.ogm.annotation.Id;
import org.neo4j.ogm.annotation.NodeEntity;
import org.neo4j.ogm.annotation.Property;
import org.neo4j.ogm.annotation.Relationship;

@NodeEntity
public class Project {

	/**
	 * Es el identificador de este objeto.
	 */
	@Id
	@GeneratedValue
	public Long id;

	/**
	 * Es el nombre público del proyecto.
	 */
	@Property
	public String projectName;

	/**
	 * Es el usuario creador del proyecto.
	 */
```

```
        @Relationship
        public User projectCreator;

}
```

6.3.3. Repositorios

Los repositorios desempeñan un papel esencial en el acceso y la manipulación de los datos. Primero, se crea un repositorio denominado *AbstractRepository*. Este es responsable de generar y mantener una única instancia de conexión con la base de datos. A continuación, se presentan las clases relevantes:

- *SessionFactory*: esta clase es provista por Neo4j-OGM. Las instancias de esta clase, poseen una conexión contra la base de datos Neo4j. Esta es necesaria para poder obtener instancias de la clase *Session*.
- *Session*: esta clase es provista por Neo4j-OGM. Las instancias de esta clase proveen la funcionalidad central para persistir y cargar objetos, mapeándolos de manera sencilla a clases Java.[15]
- *Driver*: esta clase es provista por Neo4j. Las instancias de esta clase, poseen una conexion contra la base de datos. A diferencia de *SessionFactory*, esta se utiliza para un mayor control a bajo nivel, de operaciones y transacciones complejas.

```
package ar.edu.unlp.info.lifia.agoraprojects.repositories;

import javax.enterprise.context.ApplicationScoped;
import javax.inject.Inject;

import org.neo4j.driver.Driver;
import org.neo4j.ogm.session.Session;
import org.neo4j.ogm.session.SessionFactory;

@ApplicationScoped
public class AbstractRepository {

    /**
```

```
     * Esta variable posee una instancia de conexion
     * a la base de datos Neo4j
     */
    @Inject
    SessionFactory sessionFactory;

    /**
     * Esta variable posee una sesión contra la base de datos de Neo4j.
     * Posee todas las funciones provistas por el framework Neo4j OGM
     * Utilizada para operaciones CRUD y mapeo directo a clases Java
     */
    private Session session;

    /**
     * Esta variable posee una sesión contra la base de datos de Neo4j.
     * Utilizada para operaciones y transacciones complejas.
     */
    @Inject
    protected Driver driver;

    /**
     * @return una sesion contra la base de datos de Neo4j.
     */
    protected Session getSession(){
        if( this.session == null ){
            this.session = this.sessionFactory.openSession();
        }

        return this.session;
    }
}
```

Posteriormente, se crean repositorios para gestionar las instancias de cada clase. Estos repositorios extienden de la clase *AbstractRepository* y contienen la funcionalidad específica para el manejo de su información.

Ejemplo en la clase *ProjectsRepository*:

```
/**
 * Este paquete contiene las definiciones de los repositorios que se
```

```
 * deben utilizar para acceder a la información.
 */
package ar.edu.unlp.info.lifia.agoraprojects.repositories;

import javax.enterprise.context.ApplicationScoped;

import ar.edu.unlp.info.lifia.agoraprojects.model.Project;

/**
 * Esta clase define el comportamiento de un repositorio de proyectos.
 *
 */
@ApplicationScoped
public class ProjectsRepository extends AbstractRepository{

    /**
     * Retorna la cantidad de proyectos creados.
     *
     * @return la cantidad de proyectos.
     */
    public long count(){
        return this.getSession().countEntitiesOfType(Project.class);
    }

}
```

6.4. Funcionamiento del Algoritmo KNN en Neo4j

El algoritmo KNN (K-Nearest Neighbors) en Neo4j se adapta para calcular la distancia entre nodos basándose en sus propiedades y crear relaciones entre cada nodo y sus K vecinos más cercanos. A diferencia de una implementación genérica de KNN, la versión de Neo4j opera sobre un grafo monopartito, donde los nodos son de la misma clase o etiqueta, y las relaciones existentes previas son ignoradas para la construcción de nuevas conexiones basadas en la similitud.

La implementación de Neo4j del algoritmo KNN utiliza una variedad de medidas de similitud, dependiendo del tipo de propiedades de los nodos. Para propiedades escalares, se emplea una función que da resultados en el rango (0,1], mientras que

para listas de enteros y números de punto flotante, se utilizan medidas como la Similitud Jaccard, la Similitud Coseno, la Correlación Pearson y la Distancia Euclidiana, todas normalizadas para producir valores en el rango [0,1].

La ejecución del algoritmo se realiza sobre una proyección del grafo, que es una vista materializada que contiene solo la información topológica y de propiedades relevantes para el análisis. Esta proyección se almacena en memoria utilizando estructuras de datos optimizadas, y se gestiona a través del catálogo de grafos de la biblioteca GDS de Neo4j.

Para la ejecución del algoritmo KNN se utiliza la siguiente sintaxis en Cypher:

```
CALL gds.knn.[stream|stats|mutate|write](
    graphName: String,
    configuration: Map
)
```

Donde *graphName* corresponde al nombre de la proyección del grafo y *configuration* detalla los parámetros específicos del algoritmo, como el número de vecinos a considerar (*K*), el número máximo de iteraciones (*maxIterations*), el umbral de cambio mínimo para la detención anticipada (*deltaThreshold*), y la tasa de muestreo (*sampleRate*) que equilibra precisión y rendimiento en tiempo de ejecución.

En el presente trabajo, se empleó la implementación del algoritmo KNN de Neo4j, adaptándola a las necesidades específicas del sistema de recomendación desarrollado. Los parámetros de configuración fueron ajustados para optimizar la precisión de las recomendaciones y el rendimiento, teniendo en cuenta la estructura y el tamaño del grafo de la aplicación Ágora. Los detalles sobre la utilización del algoritmo se detallarán en las secciones siguientes.

6.5. Recomendaciones en Ágora

El objetivo principal de la tesina es facilitar la recomendación de proyectos a usuarios para incentivar su participación. Para realizar esta tarea, es necesario disponer de acceso a la información histórica sobre proyectos, sus tareas, participantes y observaciones. El método seleccionado para efectuar las recomendaciones es el filtrado basado en contenido (content-based filtering). Se optó por este enfoque dado que las recomendaciones se construyen a partir de las descripciones de los ítems (proyectos) y de los perfiles o preferencias de los usuarios (participantes). El objetivo es identificar ítems con contenido similar a aquellos en los que el usuario ha participado y ha mostrado un mayor interés actual.

Las recomendaciones se generan mediante consultas en Cypher, proporcionando resultados en tiempo real.

En esta sección, se detallará y mostrará cómo se desarrollaron los distintos componentes del algoritmo de recomendación, utilizando Java y Cypher.

6.5.1. Endpoint

La funcionalidad responsable de generar recomendaciones de proyectos para los usuarios consta de un endpoint que recibe el identificador del usuario para el cual se buscarán recomendaciones y devuelve una lista paginada de proyectos sugeridos. De manera opcional, este endpoint también puede recibir el número de página y el tamaño de la página para la lista. La estructura del endpoint se define de la siguiente manera:

```
GET /api/projects/recommendations_for_user/{userId}?page={pageIndex}&
    size={pageSize}
```

Donde:

- *userId*: recibe el identificador del usuario al cual se le buscarán las recomendaciones.
- *page* (opcional): recibe el número de página a obtener del listado de recomendaciones.
- *size* (opcional): recibe el tamaño de página a obtener del listado de recomendaciones.

El procedimiento de ejecución del endpoint para obtener recomendaciones se describe mediante el siguiente flujo:

1. Al recibir Quarkus la solicitud de recomendaciones para un usuario, se activa el método *getRecommendedProjectsForUser(userId, pageIndex, pageSize)* en la clase *ProjectsResource*, con los parámetros proporcionados en la URL. Los valores por defecto para *pageIndex* y *pageSize* son 0 y 20, respectivamente.
2. Desde el método anterior, se interactúa con la capa de servicios de los proyectos, denominada *ProjectsService*. El método

   getRecommendedProjectsForUser(userId, pageIndex, pageSize) es llamado para obtener el listado de proyectos recomendados. Este, a su vez, interactúa con el repositorio encargado de realizar el filtrado de la información para generar las recomendaciones.

3. El repositorio que se encarga de realizar el filtrado de la información para generar las recomendaciones se denomina *RecommendationFilterRepository*. Allí se invoca al metodo *getRecommendedProjectsForUser(anUserId, pageIndex, pageSize)*, el cual obtiene y retorna los proyectos a recomendar. Este, arma la consulta en Cypher para realizar el filtrado, la cual es ejecutada en Neo4j.

4. Los proyectos son retornados en una colección (*Collection<Project>*) hasta llegar al metodo inicial (*getRecommendedProjectsForUser* de la clase *ProjectsResource*), donde se arma la respuesta (utilizando la clase *ResponseDTO*) y se retorna el listado en formato JSON al cliente (creando un objeto *Response*). En la respuesta tenemos dos campos:

    - *metadata*: describe los metadatos de la respuesta. Estos metadatos son el total de proyectos encontrados (*totalCount*), el tamaño de la página (*pageSize*) y el número de página actual (*pageIndex*).
    - *results*: es un array con cada uno de los proyectos retornados. En caso de no existir proyectos, se retorna un array vacío.

6.5.2. Clases y Métodos Utilizados

La funcionalidad para el cálculo y generación de recomendaciones se distribuye entre diversas clases y métodos. Estos se encargan exclusivamente de armar y ejecutar consultas en Neo4j, aprovechando el potencial del lenguaje Cypher. Con el fin de crear y actualizar los perfiles de usuarios y proyectos, así como para generar las recomendaciones, se han creado tres repositorios encargados de la interacción con la base de datos:

- *ProjectProfileRepository*: este repositorio se encarga de la creación y actualización de los perfiles de los proyectos. También se encarga de calcular la similitud entre proyectos.
- *UserProfileRepository*: este repositorio se encarga de la creación y actualización de los perfiles de los usuarios.
- *RecommendationFilterRepository*: este repositorio se encarga de realizar el filtrado de la información para generar las recomendaciones de proyectos.

Desarrollo del Perfil del Proyecto - ProjectProfileRepository

El repositorio en cuestión se responsabiliza de la creación y actualización de los perfiles de los proyectos, así como del cálculo de la similitud entre ellos. En Neo4j, el perfil de un proyecto se representa mediante las siguientes propiedades:

- *profileDataBoolean*: esta propiedad almacena un array que, utilizando la técnica de *One hot encoding*, indica mediante valores booleanos qué categorías de ciencia ciudadana y qué métodos de recolección de información están asociados al proyecto. Para cada proyecto, *profileDataBoolean* corresponde a cada fila de la tabla que se desarrolló en el capítulo anterior (ver *Content Analizer* 5.2.1).

- *projectProfileIds*: esta propiedad contiene un array que incluye los identificadores de las categorías de ciencia ciudadana y de los métodos de recolección asociados al proyecto. Esta característica es necesaria para que el algoritmo KNN realice comparaciones entre proyectos, calcular la similitud entre ellos y, así, determine los K vecinos más cercanos de cada uno.

Para la creación y actualización del perfil de los proyectos, la funcionalidad se ha dividido en tres métodos principales.

- *setProjectProfileForProjectToUserRecommendations(Project aProject)*: el método recibe, opcionalmente, un proyecto. En el caso de recibir un proyecto, inicializa o actualiza el perfil para el proyecto recibido. Caso contrario, inicializa o actualiza el perfil para todos los proyectos. La consulta que genera el perfil para los proyectos es la siguiente:

```
MATCH
   (project:Project)
WHERE
     id(project) = $projectId
MATCH
       (projectScienceType:ProjectScienceType)
WITH
       project,
       projectScienceType
ORDER BY id(projectScienceType)
WITH
       project,
```

```
        (CASE EXISTS((project)-[:PROJECT_SCIENCE_TYPES]->(
            projectScienceType)) WHEN false THEN 0 ELSE 1 END) AS
            existsRelationPST
WITH
        project,
        collect(existsRelationPST) as existsRelationPST
MATCH
        (methodSpecification:MethodSpecification)
WITH
        project,
        existsRelationPST,
        methodSpecification
ORDER BY id(methodSpecification)
WITH
        project,
        existsRelationPST,
        (CASE EXISTS((project)-[:ACTIVE_PROTOCOL]->(:Activity)-[:
            NEXT_DATASET_METADATA_SCHEMA*]->(:DatasetMetadataSchema)
            -[:METHOD_SPECIFICATION]->(methodSpecification)) WHEN
            false THEN 0 ELSE 1 END) as existsRelationMS
WITH
        project,
        existsRelationPST,
        collect(existsRelationMS) as existsRelationMS
WITH
        project,
        (existsRelationPST + existsRelationMS) as
            projectProfileBoolean
MATCH
    (project)-[:PROJECT_SCIENCE_TYPES]->(projectScienceType:
        ProjectScienceType),
    (project)-[:ACTIVE_PROTOCOL]->(:Activity)-[:
        NEXT_DATASET_METADATA_SCHEMA*]->(:DatasetMetadataSchema)-[:
        METHOD_SPECIFICATION]->(methodSpecification:
        MethodSpecification)
WITH
    project,
    projectProfileBoolean,
    collect(distinct(id(projectScienceType))) as projectScienceTypes,
    collect(distinct(id(methodSpecification))) as
        projectMethodSpecification
```

```
WITH
    project,
    projectProfileBoolean,
    (projectScienceTypes + projectMethodSpecification) as
        projectProfileIds
SET
    project.projectProfileIds = projectProfileIds,
    project.profileDataBoolean = projectProfileBoolean
```

- *initializeProjectsProfilesForProjectToUserRecommendations()*: el método inicializa el perfil para todos los proyectos. Es decir, ejecuta el método anterior sin pasarle ningún proyecto específico y no se aplica el filtro por proyecto.

- *calculateSimilarityUsingKNN(int K, double similarityCutOff)*: el método recibe un valor para *K*, que determina el número de vecinos más cercanos a buscar para cada proyecto, y un valor para *similarityCutOff*, que establece el valor mínimo de similitud entre proyectos. El algoritmo KNN, proporcionado por la librería Graph Data Science de Neo4j, se ejecuta tras realizar una proyección de los nodos que representan los proyectos. Esta proyección crea un subgrafo que incluye todos los proyectos y la propiedad *projectProfileIds*, utilizada para calcular la similitud entre ellos. La proyección se lleva a cabo mediante la ejecución de la función *gds.graph.project()*:

  ```
  CALL gds.graph.project(
      $graphName,
      'Project',
      '*',
      {
          nodeProperties: 'projectProfileIds'
      }
  )
  ```

  Una vez proyectados los nodos, se calcula el KNN mediante la ejecución de *gds.knn.write()*, que obtiene los K vecinos más cercanos para cada proyecto y registra en la base de datos una relación de similitud entre ellos:

  ```
  CALL gds.knn.write(
      $graphName,
      {
          writeRelationshipType: 'SIMILARITY',
          writeProperty: 'jaccard',
  ```

```
            nodeProperties: {
                projectProfileIds: 'JACCARD'
            },
            sampleRate: 1,
            similarityCutoff: $similarityCutOff,
            topK: $k
        }
    );
```

El algoritmo se configura para calcular la similitud utilizando la similitud Jaccard, obtener los K vecinos especificados por parámetro y utilizar, como similitud mínima entre proyectos, el valor de *similarityCutOff*. La representación de la relación de similitud entre proyectos se visualiza de la siguiente manera:

Figura 6.2: Similitud entre proyectos en Neo4j

Desarrollo del Perfil del Usuario - UserProfileRepository

El perfil de usuario es un elemento clave en un sistema de recomendación. La creación y actualización de este perfil permite conocer las preferencias y patrones de comportamiento de los usuarios, lo que se traduce en recomendaciones más precisas y personalizadas. En el proyecto, se implementa el perfil del usuario en el repositorio *UserProfileRepository*, que proporciona la funcionalidad necesaria para crear y actualizar el perfil de usuario para cada proyecto en el que ha participado. El perfil de un usuario para un proyecto consta de tres propiedades:

- *lastObservationDate*: esta propiedad posee la fecha de la última vez que el usuario aportó una muestra en un proyecto.
- *totalObservations*: esta propiedad posee la cantidad de observaciones que realizó el usuario en un proyecto.

- *userActivityRate*: esta propiedad posee la tasa de actividad del usuario en un proyecto.

La clase *UserProfileRepository* contiene la funcionalidad necesaria para crear y actualizar el perfil de usuario correspondiente a cada proyecto en el que ha participado. La funcionalidad se divide en tres métodos principales:

1. *initializeUserProfilesForProjectToUserRecommendations(User anUser, Project aProject)*: este método se encarga de inicializar el perfil de un usuario para un proyecto, en el caso de que haya participado. Si no se recibe información de usuario o proyecto, se inicializa el perfil para todos los usuarios en todos los proyectos en los que han participado. Si se recibe información solo del usuario, se inicializa el perfil para dicho usuario en todos los proyectos en los que ha participado. Si se recibe información solo del proyecto, se inicializa el perfil para todos los usuarios que han participado en él. Si se reciben ambos, usuario y proyecto, se inicializa el perfil para el usuario específico en el proyecto indicado. La representación de la relación del usuario con su perfil para un proyecto específico se visualiza de la siguiente manera:

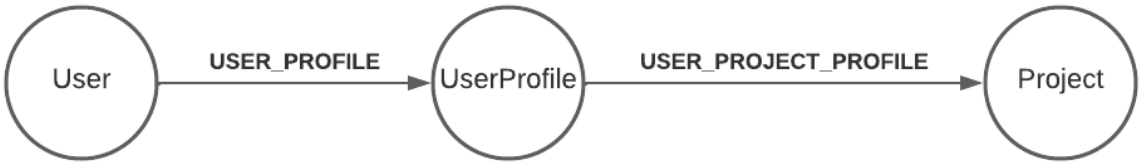

Figura 6.3: Perfil del usuario en proyecto en Neo4j

La consulta creada para calcular y almacenar el perfil del usuario es la siguiente:

```
MATCH
(user:User)<-[:PROJECT_PARTICIPANTS]-(project:Project)
WHERE
     id(user) = $userId
     AND id(project) = $projectId
OPTIONAL MATCH
    (project)-[:ACTIVE_PROTOCOL]->(:Activity)-[:
       NEXT_DATASET_METADATA_SCHEMA*]->(:DatasetMetadataSchema)-[:
       OBSERVATIONS]->(observationForProject:Observation)-[:
       OBSERVATION_RESPONSIBLE]->(user)
```

```
WITH
    user,
    project,
    COUNT(DISTINCT(observationForProject.observationCreatedAt)) AS
        totalObservationsForProject,
    MAX(DATETIME(observationForProject.observationCreatedAt)) AS
        lastObservationDate,
    MIN(DATETIME(observationForProject.observationCreatedAt)) AS
        firstObservationDate
WITH
    user,
    project,
    totalObservationsForProject,
    lastObservationDate,
    duration.inDays(firstObservationDate, lastObservationDate).days
        AS daysBetween
WITH
    user,
    project,
    totalObservationsForProject,
    lastObservationDate,
    daysBetween,
    ( (totalObservationsForProject*1.0) / (daysBetween*1.0) ) AS
        userActivityRate
CREATE
    (project)<-[:USER_PROJECT_PROFILE]-(up:UserProfile{
        totalObservations: totalObservationsForProject,
        lastObservationDate: lastObservationDate, userActivityRate:
        CASE totalObservationsForProject WHEN 0 THEN 0 ELSE
        userActivityRate END})<-[:USER_PROFILE]-(user);
```

2. *initializeUsersProfilesForProjectToUserRecommendations()*: este método inicializa los perfiles de todos los usuarios para todos los proyectos en los que cada uno participó, invocando al método previamente mencionado sin aplicar filtros de usuario y proyecto.

3. *updateUserProfilesForProjectToUserRecommendations(User anUser, Project aProject)*: este método se encarga de actualizar el perfil de un usuario para un proyecto. La consulta para la actualización del perfil es similar a la de la creación, con la diferencia de que utiliza la cláusula *SET* en lugar de *CREATE* para

actualizar los valores de *totalObservations*, *lastObservationDate* y *userActivityRate*.

Desarrollo del Filtrado para Generar la Recomendación - RecommendationFilterRepository

En el repositorio denominado *RecommendationFilterRepository*, se lleva a cabo el proceso de filtrado de la información para generar recomendaciones de proyectos a usuarios. Dicho proceso considera el interés actual del usuario en cada proyecto en el que participa. La funcionalidad para realizar el filtrado y proporcionar los proyectos recomendados está encapsulada en la clase *RecommendationFilterRepository*. Si bien un único método ejecuta la consulta de filtrado y retorna los proyectos obtenidos, la construcción del string de la consulta se divide en varios métodos para clarificar cada bloque de código dentro de la consulta, facilitando así la comprensión del proceso de filtrado paso a paso. Los métodos desarrollados para esta tarea son:

- *getUserProfileAndSimilarProjectsQuery()*: el método retorna el string de la consulta encargada de obtener el perfil del usuario y los proyectos más similares a cada uno en los que ha participado:

```
MATCH
    (participant:User)<-[:PROJECT_PARTICIPANTS]-(participantProject:
       Project)<-[:USER_PROJECT_PROFILE]-(participantProfile:
       UserProfile)<-[:USER_PROFILE]-(participant)
WHERE
    id(participant) = $userId
WITH
    participantProfile,
    participantProject,
    participant
OPTIONAL MATCH
    (participantProject)-[similarity:SIMILARITY]->(similarProject:
       Project)
WHERE
    NOT EXISTS((participant)<-[:PROJECT_PARTICIPANTS]-(similarProject
       ))
    AND similarProject.profileDataBoolean IS NOT NULL
```

- *getUserCalculatedProfileAndValuesForNormalizationQuery()*: el método retorna el string de la consulta que calcula, para el perfil del usuario en cada proyecto, el valor de la escala definida (en 5.2.2) correspondiente a la fecha de la

última observación y genera los valores mínimos y máximos para normalizar cada propiedad:

```
WITH
    apoc.coll.toSet(collect(participantProfile)) AS
        participantProfiles,
    apoc.coll.toSet(collect(similarProject)) AS similarProjects,
    datetime() AS now
UNWIND participantProfiles AS participantProfile
MATCH
    (participantProfile)-[:USER_PROJECT_PROFILE]->(participantProject
        :Project)
WITH
    collect({projectProfileDataBoolean: participantProject.
        profileDataBoolean, totalObservations: participantProfile.
        totalObservations, lastObservationDate: participantProfile.
        lastObservationDate, userActivityRate: participantProfile.
        userActivityRate, calculatedValueForDate: CASE WHEN duration.
        inMonths(datetime(participantProfile.lastObservationDate),
        now).months < 12 THEN 3 WHEN 12 < duration.inMonths(datetime(
        participantProfile.lastObservationDate), now).months < 24
        THEN 2 ELSE 1 END}) AS evaluatedParticipantProfiles,
    similarProjects,
    toFloat(MIN(participantProfile.totalObservations)) AS
        minForObservationsNormalization,
    toFloat(MAX(participantProfile.totalObservations)) AS
        maxForObservationsNormalization,
    toFloat(MIN(participantProfile.userActivityRate)) AS
        minForUserActivityRateNormalization,
    toFloat(MAX(participantProfile.userActivityRate)) AS
        maxForUserActivityRateNormalization
WITH
    evaluatedParticipantProfiles,
    similarProjects,
    minForObservationsNormalization,
    maxForObservationsNormalization,
    minForUserActivityRateNormalization,
    maxForUserActivityRateNormalization
UNWIND evaluatedParticipantProfiles AS evaluatedParticipantProfile
WITH
    collect(evaluatedParticipantProfile) AS
```

```
    evaluatedParticipantProfiles,
similarProjects,
minForObservationsNormalization,
maxForObservationsNormalization,
minForUserActivityRateNormalization,
maxForUserActivityRateNormalization,
toFloat(MIN(evaluatedParticipantProfile.calculatedValueForDate))
    AS minForCalculatedValueForDateNormalization,
toFloat(MAX(evaluatedParticipantProfile.calculatedValueForDate))
    AS maxForCalculatedValueForDateNormalization
```

- *getUserNormalizedProfileWithInterestQuery()*: el método retorna la consulta que normaliza los valores correspondientes al total de observaciones, el valor de la escala definida para la fecha de la última observación y la tasa de actividad del usuario. A cada valor normalizado se le aplica el peso obtenido en la tabla de comparación por pares (AHP), calculado para cada característica (ver tabla 5.10). Finalmente, se calcula el nivel de interés actual del usuario en cada proyecto en el que ha participado:

```
WITH
    similarProjects,
    [
        evaluatedParticipantProfile IN evaluatedParticipantProfiles |
            evaluatedParticipantProfile{
                .*,
                normalizedDate: CASE WHEN
                    minForCalculatedValueForDateNormalization =
                    maxForCalculatedValueForDateNormalization THEN 0.5
                    * 0.45 ELSE ((evaluatedParticipantProfile.
                    calculatedValueForDate -
                    minForCalculatedValueForDateNormalization)/(
                    maxForCalculatedValueForDateNormalization-
                    minForCalculatedValueForDateNormalization)) * 0.45
                    END,
                normalizedActivityRate: CASE WHEN
                    maxForUserActivityRateNormalization = 0 AND
                    minForUserActivityRateNormalization = 0 THEN 0 WHEN
                     maxForUserActivityRateNormalization =
                    minForUserActivityRateNormalization THEN 0.5 * 0.55
                     ELSE ((evaluatedParticipantProfile.
                    userActivityRate-
```

```
                minForUserActivityRateNormalization)/(
                maxForUserActivityRateNormalization-
                minForUserActivityRateNormalization)) * 0.55 END,
            normalizedObservations: 0}
    ] AS normalizedParticipantProfiles
WITH
    similarProjects,
    [
        normalizedParticipantProfile IN normalizedParticipantProfiles
            |
        normalizedParticipantProfile{.*, actualInterest:
            normalizedParticipantProfile.normalizedObservations +
            normalizedParticipantProfile.normalizedDate +
            normalizedParticipantProfile.normalizedActivityRate }
    ] AS normalizedParticipantProfilesWithInterest
```

- *getUserInterestPerProjectFeatureQuery()*: el método retorna la consulta que calcula el interés del usuario para cada característica, incluyendo categorías y métodos de recolección, de los proyectos en los que ha participado:

```
WITH
    similarProjects,
    [
        normalizedParticipantProfile IN
            normalizedParticipantProfilesWithInterest |
        {
            interestPerFeature: [
                projectFeatureBoolean IN normalizedParticipantProfile
                    .projectProfileDataBoolean |
                projectFeatureBoolean * normalizedParticipantProfile.
                    actualInterest
            ]
        }
    ] AS participantInterestPerProjectFeature
```

- *getFeatureInterestPercentageQuery()*: el método retorna la consulta que genera un acumulador del interés general del usuario para cada característica de cada proyecto:

```
UNWIND
    range(0, size(participantInterestPerProjectFeature[0].
        interestPerFeature)-1) AS featurePosition
```

```
WITH
    similarProjects,
    [
        interest IN participantInterestPerProjectFeature | interest.
            interestPerFeature[featurePosition]
    ] as interestAccumulator
WITH
    similarProjects,
    collect(apoc.coll.sum(interestAccumulator)/size(
        interestAccumulator)) as interestPerFeature
```

- *getProjectsWithPossibleInterestPerProjectQuery()*: el método retorna la consulta que calcula el interés estimado para cada proyecto similar encontrado, basándose en el interés general en cada característica, calculado en el método anterior. En este punto de la consulta, también se retornan los proyectos recomendados, ordenados por el nivel de interés calculado de manera descendente:

```
UNWIND similarProjects as similarProject
WITH
    similarProject,
    interestPerFeature
UNWIND
    range(0, size(interestPerFeature)-1) AS featureInterestPosition
WITH
    similarProject,
collect([similarProject.profileDataBoolean[featureInterestPosition],
    interestPerFeature[featureInterestPosition]]) AS
    featureInterestPairsForMultiplication
WITH
    similarProject,
    [
        featurePair IN featureInterestPairsForMultiplication |
            featurePair[0]*featurePair[1]
    ] AS featureInterestPairMultiplied
WITH
    similarProject,
    ( apoc.coll.sum(featureInterestPairMultiplied) / apoc.coll.sum(
        similarProject.profileDataBoolean) ) AS calculatedInterest
ORDER BY calculatedInterest DESC
RETURN calculatedInterest, similarProject
```

- *getRecommendedProjectsForUser(String anUserId)*: el método ejecuta la consulta de recomendación de proyectos a usuarios para participar. Retorna los proyectos recomendados. Este, arma la consulta concatenando cada string de cada método mencionado previamente y la ejecuta en Neo4j. Recibe como parámetro el identificador del usuario al cual se le harán las recomendaciones (*anUserId*):

```
public Collection<Map<String, Object>> getRecommendedProjectsForUser
    (String anUserId, int pageIndex, int pageSize, int limit){
    String recommendationFilterQuery = this.
        getUserProfileAndSimilarProjectsQuery()
        + this.getUserCalculatedProfileAndValuesForNormalizationQuery
            ()
        + this.getUserNormalizedProfileWithInterestQuery()
        + this.getUserInterestPerProjectFeatureQuery()
        + this.getFeatureInterestPercentageQuery()
        + this.getProjectsWithPossibleInterestPerProjectQuery()
        + this.getLimitQuery(limit);
        + this.getPaginationQuery(pageIndex, pageSize);

    Map<String, Object> queryParams = new HashMap<String, Object>();
    queryParams.put("userId", Long.parseLong(anUserId));
    try( var session = this.driver.session() ){

        return session.readTransaction(transaction -> {
            return (Collection<Map<String, Object>>) transaction.run(
                recommendationFilterQuery, queryParams).list(record
                ->{
                return record.asMap();
            });
        });

    }

}
```

Capítulo 7

Evaluación

En este capítulo, se presentarán los resultados obtenidos tras múltiples ejecuciones del algoritmo de recomendación de proyectos a usuarios. Estos resultados se expresarán en términos de *Precision*, *Recall* y *F1-Score*, proporcionando una evaluación completa del desempeño del sistema de recomendación basado en contenido en Ágora.

La evaluación del algoritmo de recomendación es un aspecto fundamental para determinar su efectividad y valorar su capacidad para proporcionar sugerencias relevantes y personalizadas a los usuarios. Para llevar a cabo dicha evaluación, se realizaron pruebas exhaustivas utilizando una variedad de conjuntos de datos y escenarios, buscando abarcar diferentes situaciones y contextos.

Finalmente, se propone una configuración para la ejecución del algoritmo KNN destinada a obtener los proyectos más similares, la cual ha demostrado generar recomendaciones con un alto promedio de aciertos.

7.1. Métricas de Evaluación

En esta sección, se explicarán las métricas de evaluación utilizadas: *Precision*, *Recall* y *F1-Score* [26].

7.1.1. Precision

La métrica *Precision* se utiliza en la evaluación de sistemas de recomendación para medir la proporción de aciertos dentro del conjunto de recomendaciones realizadas. Refleja la habilidad del sistema para ofrecer sugerencias relevantes y de valor

para los usuarios. Se calcula dividiendo el número de recomendaciones correctas entre el total de recomendaciones efectuadas:

$$\text{Precision} = \frac{|\text{documentos relevantes} \cap \text{documentos recuperados}|}{\text{documentos recuperados}}$$

7.1.2. Recall

La métrica *Recall* mide la capacidad del sistema para identificar y recuperar proyectos relevantes de entre el total disponible. Es una métrica que evalúa la cantidad de ítems relevantes que el sistema puede recuperar correctamente. Se calcula dividiendo el número de recomendaciones correctas entre el número total de ítems relevantes presentes en el conjunto de datos:

$$\text{Recall} = \frac{|\text{documentos relevantes} \cap \text{documentos recuperados}|}{\text{documentos relevantes}}$$

7.1.3. F1-Score

El *F1-Score* combina tanto la *Precision* como el *Recall* en un valor único que refleja el equilibrio entre la precisión y la exhaustividad del sistema. Se utiliza como una medida general del desempeño del sistema de recomendación. El *F1-Score* se calcula como la media de la *Precision* y el *Recall*, y proporciona una visión más completa del rendimiento del sistema al considerar tanto la capacidad de realizar recomendaciones precisas como la capacidad de recuperar proyectos relevantes:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Además de los resultados obtenidos a través de los indicadores mencionados, se considerarán otros aspectos de evaluación, como la diversidad de las recomendaciones, la serendipia y la satisfacción de los usuarios en relación con las recomendaciones recibidas.

Mediante el análisis y la interpretación de los resultados de evaluación, se podrá determinar la efectividad y utilidad del sistema de recomendación basado en contenido desarrollado, así como identificar posibles mejoras y áreas de oportunidad para futuras investigaciones.

En resumen, este capítulo proporcionará una visión completa de la evaluación del sistema de recomendación para Ágora, mostrando los resultados obtenidos en

términos de *Precision*, *Recall* y *F1-Score*, y analizando aspectos adicionales relacionados con la calidad y la satisfacción de las recomendaciones brindadas a los usuarios.

7.2. Conjunto de Datos

En esta sección, se describe el conjunto de datos utilizado para la evaluación del sistema de recomendación.

7.2.1. Inicialización de Proyectos

Los proyectos utilizados se obtuvieron de la plataforma de ciencia ciudadana https://www.citizenscience.gov/catalog/. Dichos proyectos se descargaron en formato CSV y se procesaron mediante un algoritmo en Java para su almacenamiento en la base de datos Neo4j. Para cada proyecto, se disponía de la información necesaria para calcular la similitud entre ellos, es decir, las categorías de ciencia ciudadana y los métodos de recolección de información correspondientes. Los proyectos se inicializaron en la base de datos con su nombre, categorías y métodos de recolección asociados. En total, se utilizaron 468 proyectos. El dataset utilizado consiste en una colección de registros de proyectos de ciencia ciudadana. Cada registro en el dataset está caracterizado por una serie de atributos que proporcionan información detallada sobre el proyecto. Los campos incluidos en el dataset son:

- **project_id**: Identificador único para cada proyecto.
- **project_name**: Nombre del proyecto.
- **project_url_on_catalog**: URL del proyecto en el catálogo de ciencia ciudadana.
- **project_url_external**: URL externa del proyecto.
- **project_description**: Descripción detallada del proyecto.
- **keywords**: Palabras clave asociadas con el proyecto.
- **fields_of_science**: Campos de la ciencia relacionados con el proyecto. En nuestro sistema, es el listado de categorías de ciencia ciudadana a las que pertenece el proyecto.

- **project_status**: Estado actual del proyecto.
- **agency_sponsor**: Agencia o entidad patrocinadora del proyecto.
- **agency_sponsor_other**: Información adicional sobre el patrocinador.
- **gov_contact**: Contacto gubernamental para el proyecto.
- **gov_contact_email**: Correo electrónico del contacto gubernamental.
- **geographic_scope**: Alcance geográfico del proyecto.
- **participant_age**: Rango de edad de los participantes en el proyecto.
- **participation_tasks**: Tareas en las que participan los ciudadanos. En nuestro sistema, es el listado de métodos de recolección de informacion que utiliza el protocolo del proyecto.
- **scistarter**: Indicador de si el proyecto está asociado con SciStarter.
- **email**: Correo electrónico de contacto para el proyecto.
- **start_date**: Fecha de inicio del proyecto.
- **project_goals**: Metas y objetivos del proyecto.

En la tabla 7.1, se presenta un fragmento del dataset en formato CSV.

7.2.2. Inicialización de Usuarios

Se generaron 1000 usuarios de manera aleatoria utilizando la herramienta Mockaroo. Los datos de los usuarios se exportaron en un archivo CSV y se procesaron mediante un algoritmo en Java para almacenarlos en la base de datos Neo4j. Cada registro en el dataset representa un usuario único y está caracterizado por los siguientes atributos:

- **userId**: Identificador único para cada usuario.
- **userFirstName**: Nombre del usuario.
- **userLastName**: Apellido del usuario.
- **userEmail**: Dirección de correo electrónico del usuario.

- **userDocument**: Número de documento o identificación del usuario.
- **username**: Nombre de usuario utilizado para el inicio de sesión.

En la tabla 7.2, se presenta un fragmento del dataset de usuarios en una tabla vertical:

7.2.3. Inicialización de perfiles de usuarios

Para inicializar el perfil de cada usuario, se realizaron los siguientes pasos:

- Se iteró sobre cada uno de los proyectos creados, obteniendo aquellos con un grado de similitud mayor o igual a 0.5, 0.6, 0.7, 0.8 o 0.9 (detalles que serán explicados más adelante). Por lo tanto, en este punto se dispone de X arrays de proyectos similares, siendo X el número total de proyectos en la base de datos.
- Para cada array de proyectos similares, se añadió un usuario que no participaba en ningún otro proyecto (usuario U) como participante en los mismos. Esto permitió que el usuario U se convirtiera en participante de proyectos similares.
- Se generó un perfil para cada usuario que participó en un proyecto, con el objetivo de reflejar un elevado interés en los proyectos en los que participaban, es decir, un alto interés de participación en proyectos similares. Los datos generados para los perfiles son los siguientes:
    - El total de observaciones es un número entero elegido al azar entre 100 y 200.
    - La fecha de la última muestra es una fecha elegida al azar entre enero y octubre del 2022.
    - La tasa de actividad es un número elegido al azar entre 0.8 y 1.0.

7.3. Proceso de Evaluación

En esta sección, se explica el cálculo de las métricas de *Precision*, *Recall* y *F1-Score* empleado para la evaluación del sistema de recomendación para Ágora.

Para calcular la *Precision*, *Recall* y *F1-Score* del sistema de recomendación para un usuario, se siguieron los pasos detallados a continuación:

1. Se calcula la similitud entre proyectos utilizando el algoritmo KNN con un valor específico de K. Dado que la base de datos contiene 468 proyectos, para las pruebas mencionadas, los valores de K seleccionados son 5, 25, 50 y 150.

2. Se elimina la relación de participación entre un usuario U y la mitad de los proyectos en los que participó. Por lo tanto, si previamente se calculó el perfil del usuario U en un conjunto de 20 proyectos similares, en este paso se suprime la relación de U con 10 de esos proyectos.

3. Se ejecuta el algoritmo de filtrado de información para generar las recomendaciones para el usuario U.

4. Se calculan las métricas *Precision*, *Recall* y *F1-Score* utilizando las fórmulas mencionadas, tomando como **documentos relevantes** aquellos proyectos a los cuales se les eliminó la relación de participación (los que se deberían ser recomendados) y como **documentos recuperados** los proyectos efectivamente recomendados.

5. Se restablece la relación de participación entre el usuario U y los proyectos para volver al estado inicial, permitiendo así la ejecución del mismo procedimiento con un nuevo valor de K.

Este proceso se repitió para diferentes niveles de similitud entre proyectos. Es decir, en la configuración del algoritmo KNN, se encontraron los K vecinos más cercanos con un valor de similitud en el rango de [X, 1], siendo X el valor mínimo de similitud entre proyectos. Para las pruebas realizadas, el valor de X se estableció en 0.5, 0.6, 0.7, 0.8 y 0.9. El pseudocódigo utilizado para la ejecución de las pruebas se presenta a continuación:

Algorithm 1 Evaluación del sistema de recomendación

```
similitudes ← [0,5, 0,6, 0,7, 0,8, 0,9]
Ks ← [5, 25, 50, 150]
for cada similitud en similitudes do
    CalcularSimilitudEntreTodosLosProyectos()
    for cada P en Grafo do
        proyectosSimilares ← ObtenerProyectosConSimilitud(P, similitud)
        proyectosSimilares ←AgregarProyectoAConjunto(P, proyectosSimilares)
        usuarioU ← SeleccionarUsuarioNoParticipanteEnNingunProyecto()
        AgregarUsuarioComoParticipante(proyectosSimilares, usuarioU)
        CrearPerfilUsuario(usuarioU, proyectosSimilares)
    end for
    EliminarRelacionesDeSimilitudEnTodaLaBD()
    usuariosParticipantes ← ObtenerUsuariosParticipantes()
    for cada K en Ks do
        EjecutarAlgoritmoKNN(K, similitud)
        for cada usuarioU en usuariosParticipantes do
            intereses ← CalcularInteresesActuales(usuarioU)
            intereses ← OrdenarInteresesDeMayorAMenor(intereses)
            proyectosRelevantes ← SeleccionarProyectosConMayorInteres(intereses)
            EliminarRelacionDeParticipacion(usuarioU, proyectosRelevantes)
            recomendaciones ← EjecutarAlgoritmoRecomendacion(usuarioU)
            precision ← CalcularPrecision(recomendaciones, proyectosRelevantes)
            recall ← CalcularRecall(recomendaciones, proyectosRelevantes)
            f1score ← CalcularF1Score(precision, recall)
        end for
        RestaurarRelacionDeParticipacionUsuario(usuarioU, proyectosRelevantes)
    end for
end for
```

Campo	Descripción
project_id	3
project_name	NOAA NWS SKYWARN® Weather Spotter Program
project_url_on_catalog	https://www.citizenscience.gov/catalog/3/
project_url_external	http://www.nws.noaa.gov/skywarn/
project_description	SKYWARN® is a National Weather Service (NWS) program developed in the 1960s...
keywords	Skywarn, weather, spotter, flood, tornado, thunderstorm, hurricane, typhoon, snow, ice, wind, damage, storm, NOAA, NWS, training, meteorology
fields_of_science	Climate and weather
project_status	active
agency_sponsor	National Oceanic and Atmospheric Administration (NOAA)
agency_sponsor_other	
gov_contact	John McLaughlin
gov_contact_email	john.mclaughlin@noaa.gov
geographic_scope	National
participant_age	families, general_public, seniors, teens
participation_tasks	audio_video_recording, identification, learning, measurement, observation, photography
scistarter	true
email	ron.gird@noaa.gov
start_date	1965
project_goals	To train and engage volunteers in reporting severe and hazardous weather conditions to assist meteorologists and emergency managers in making life-saving decisions.

Cuadro 7.1: Ejemplo de Dataset de Proyectos de Ciencia Ciudadana

Campo	Descripción
userId	1
userFirstName	Waneta
userLastName	Burdekin
userEmail	wburdekin0@reverbnation.com
userDocument	56209641
username	wburdekin0

Cuadro 7.2: Ejemplo del Dataset de Usuarios

En el algoritmo se observa que, en cada iteración, se selecciona un usuario que no participa en ningún proyecto para asignarlo como participante en proyectos similares. Por lo tanto, de los 1000 usuarios existentes en la base de datos, solo algunos (aquellos que son participantes en algún proyecto) recibirán recomendaciones. Los demás no recibirán recomendaciones, ya que no poseen un perfil de usuario creado para ningún proyecto y, por ende, no se puede predecir su interés en el resto de los proyectos para obtener recomendaciones. Este fenómeno se conoce como *cold start* o arranque en frío (sección 4.4.3), donde los usuarios que aún no han participado en ningún proyecto no tienen proyectos similares identificados para predecir un nivel de interés y recomendar.

7.4. Evaluación de Resultados

En esta sección, se presentan los resultados obtenidos al evaluar el sistema de recomendación utilizando dos conjuntos de datos: uno con 468 proyectos y otro con 300 proyectos. Se analiza el desempeño del sistema en términos de *Precision*, *Recall* y *F1-Score*, considerando diferentes niveles de similitud y valores de K. Se ha utilizado el algoritmo KNN con diferentes valores de K, evaluando los resultados en un rango de similitud entre proyectos de [0.5, 1.0]. Los resultados se promediaron para obtener un valor representativo de *Precision*, *Recall* y *F1-Score* para cada configuración de K.

7.4.1. Resultados con 468 Proyectos

A continuación, se muestran los resultados de la evaluación utilizando 468 proyectos. Se presentan los gráficos de *Precision*, *Recall* y *F1-Score*, donde se observa la variación de las métricas de evaluación para cada valor de K:

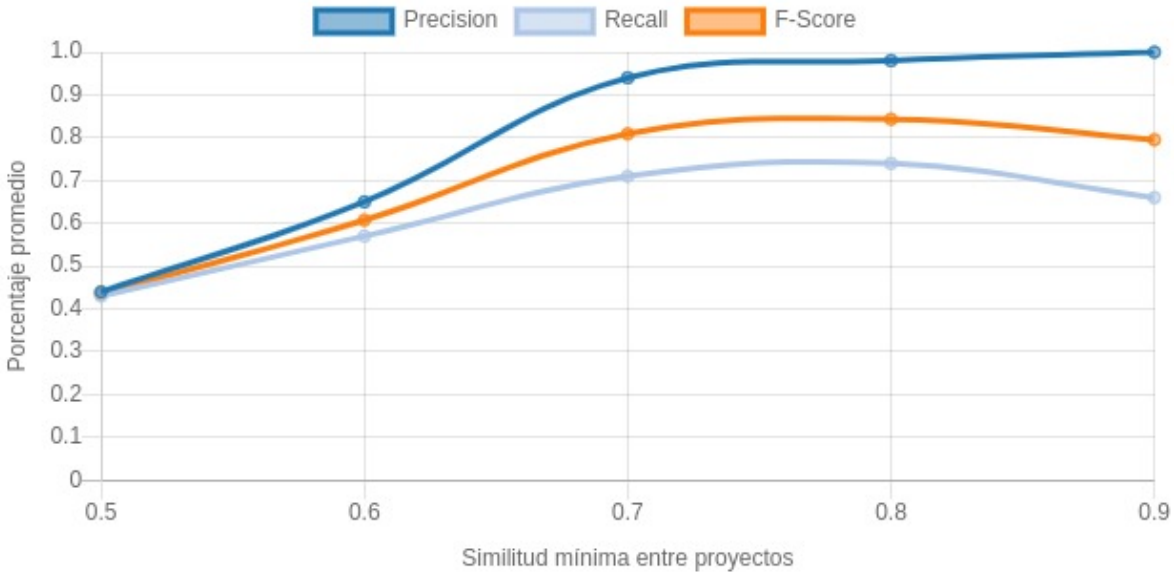

Figura 7.1: Valor promedio de Precision, Recall y F1-Score para K-5

El gráfico muestra un ascenso consistente de *Precision* a medida que se incrementa la similitud mínima, estabilizándose cerca de una similitud de 0.7. Esto quiere decir que la configuración es más efectiva en identificar proyectos relevantes con un valor de similitud más estricto, pero no se beneficia mucho de un valor más alto que 0.7. *Recall* también aumenta con la similitud mínima y alcanza un punto óptimo alrededor de 0.7, después del cual se estabiliza. Esto indica que el sistema es capaz de recuperar la mayoría de los proyectos relevantes sin necesidad de incrementar excesivamente la similitud mínima. *F1-Score* muestra un pico de rendimiento cerca de una similitud mínima de 0.7, después del cual el incremento en la similitud mínima no mejora la armonía entre la precisión y el recall.

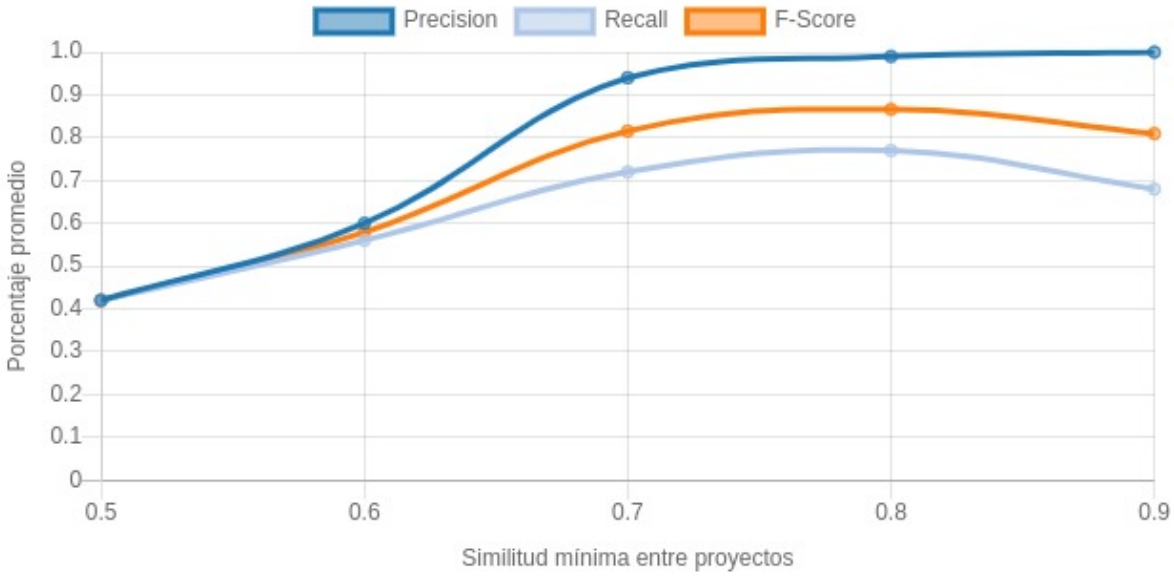

Figura 7.2: Valor promedio de Precision, Recall y F1-Score para K-25

En este gráfico, *Precision* presenta una mejora inicial significativa y luego se nivela pasada una similitud mínima de 0.7, lo que puede indicar que el sistema ha encontrado un equilibrio entre la inclusión de proyectos relevantes y la exclusión de los no relevantes. *Recall* muestra un comportamiento similar *Precision*, mejorando inicialmente y luego estabilizándose, lo que sugiere que se han identificado efectivamente la mayoría de los proyectos relevantes dentro de este rango de similitud mínima. *F1-Score* sigue la tendencia de la precisión y del recall, indicando que el sistema logra un buen equilibrio en la calidad de las recomendaciones alrededor de una similitud mínima de 0.7.

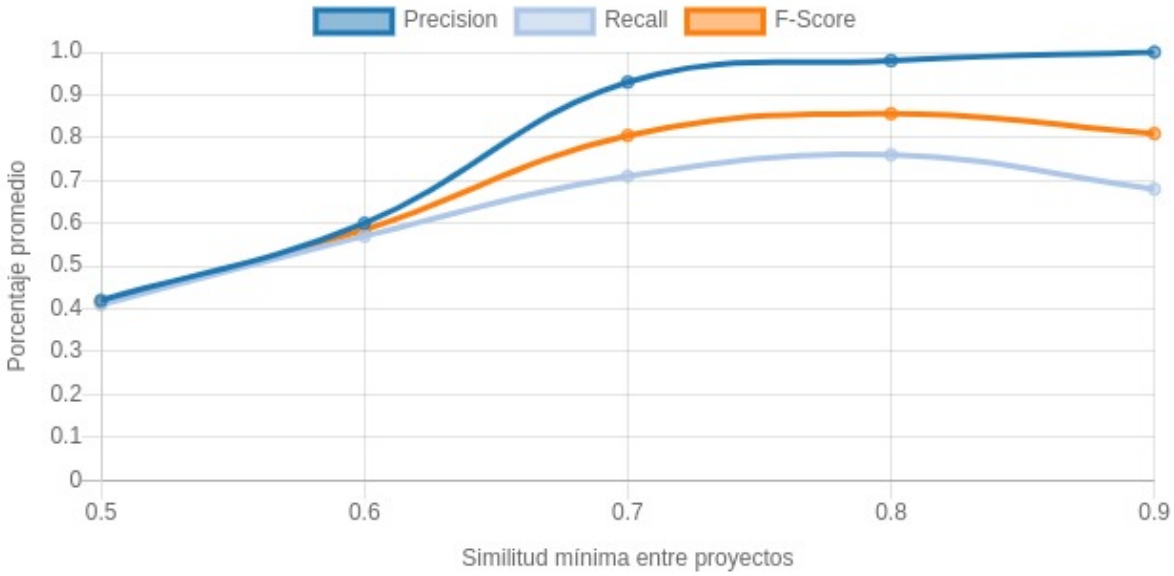

Figura 7.3: Valor promedio de Precision, Recall y F1-Score para K-50

En este gráfico no se ve una diferencia tan marcada en comparación con el gráfico de K-25. Esto quiere decir que con un valor de K más alto que 25, no se obtienen beneficios significativos y, por lo tanto, podría ser más eficiente, computacionalmente y efectivo en términos de calidad de las recomendaciones, mantener un valor de K no tan alto.

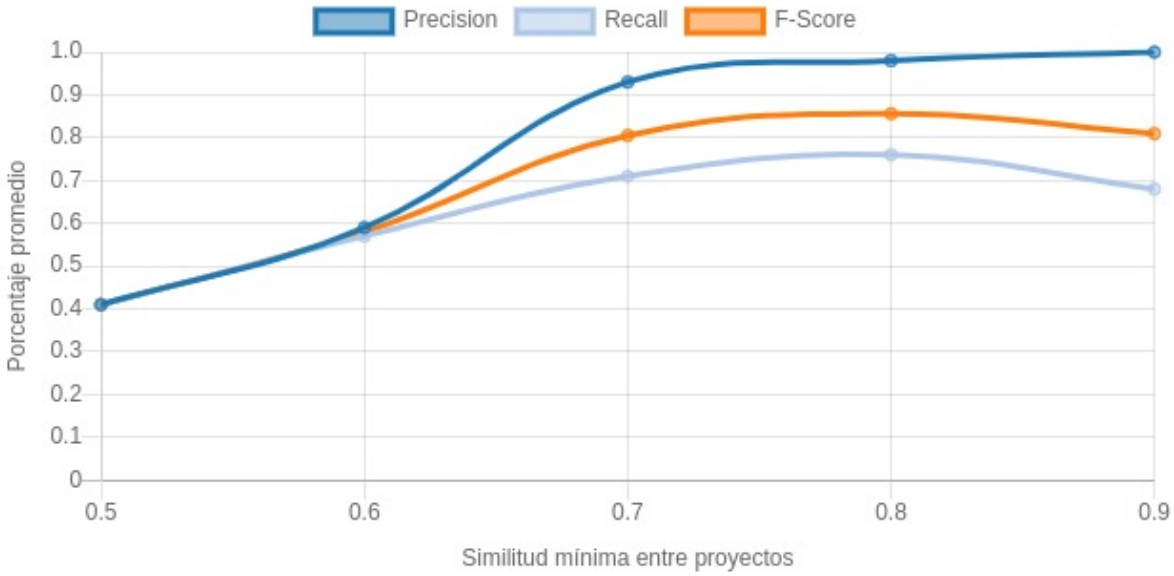

Figura 7.4: Valor promedio de Precision, Recall y F1-Score para K-150

En este gráfico no se ve una diferencia tan marcada en comparación con el gráfico de K-50. Esto quiere decir que con un valor de K más alto que 50, no se obtienen beneficios significativos y, por lo tanto, podría ser más eficiente, computacionalmente y efectivo en términos de calidad de las recomendaciones, mantener un valor de K no tan alto.

7.4.2. Resultados con 300 proyectos

De manera similar a la sección anterior, se presentan los resultados de la evaluación utilizando 300 proyectos. Se presentan los gráficos de *Precision*, *Recall* y *F1-Score*, donde se observa la variación de las métricas de evaluación para cada valor de K:

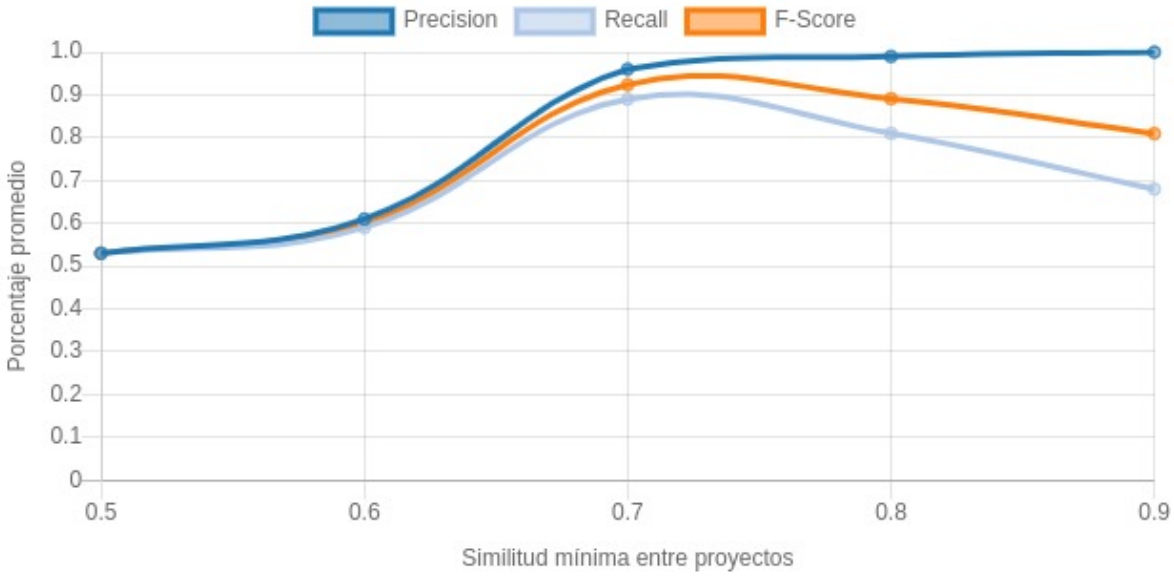

Figura 7.5: Valor promedio de Precision, Recall y F1-Score para K-5

El gráfico muestra que *Precision* comienza con un aumento pronunciado a medida que la similitud mínima se incrementa de 0.5 a 0.7, lo que indica una mejora significativa en la capacidad del sistema para hacer recomendaciones precisas bajo dicho valor de similitud. Sin embargo, *Precision* se estabiliza y muestra un ligero descenso después de una similitud mínima de 0.8, lo que podría indicar que la configuración se vuelve demasiado restrictiva, excluyendo potencialmente algunas recomendaciones relevantes. *Recall* mejora de manera menos pronunciada y comienza a estabilizarse después de una similitud mínima de 0.7. Esto sugiere que, aunque el sistema se vuelve mejor en identificar proyectos relevantes, hay un límite en su capacidad para recuperar todos los proyectos relevantes posibles a medida que la similitud mínima se incrementa. *F1-Score*, que equilibra *Precision* y *Recall*, sigue un patrón similar a *Precision*, alcanzando un máximo alrededor de 0.7 de similitud mínima y luego disminuyendo ligeramente. Esto refleja un equilibrio entre las dos métricas hasta que el sistema se vuelve demasiado selectivo.

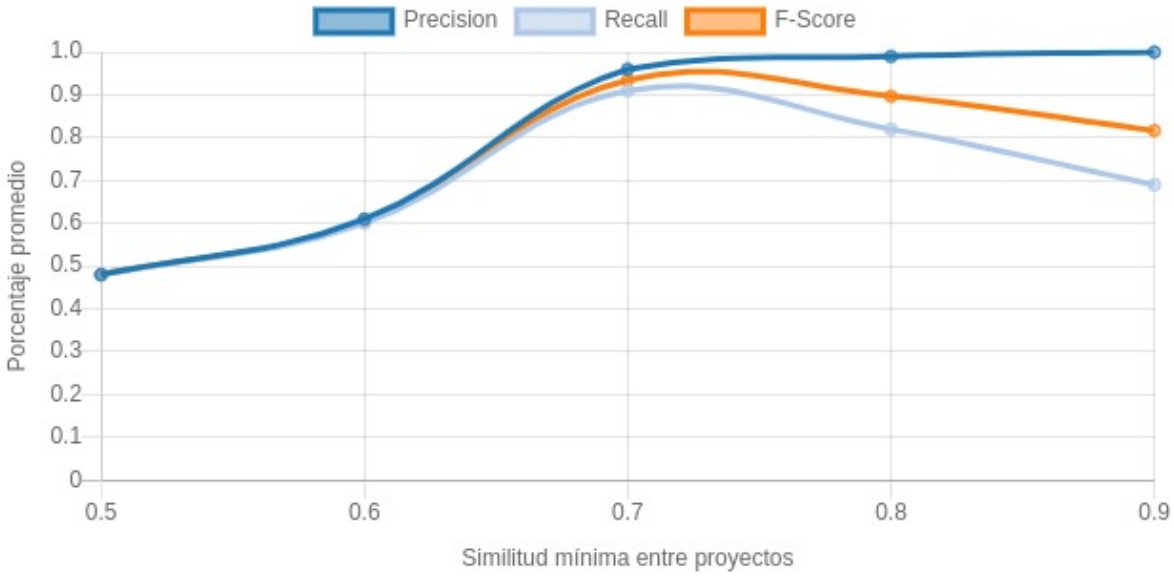

Figura 7.6: Valor promedio de Precision, Recall y F1-Score para K-25

En este gráfico, *Precision* muestra un aumento sostenido hasta una similitud mínima de 0.7 y luego se estabiliza, lo que indica que un mayor número de vecinos contribuye a una calidad constante de recomendaciones precisas hasta cierto punto de similitud mínima. También muestra un ligero descenso después de una similitud mínima de 0.8. *Recall* muestra una tendencia creciente hasta el mismo punto de 0.7 y luego se nivela, lo que sugiere que la capacidad del sistema para recuperar proyectos relevantes no mejora al incrementar la similitud mínima más allá de este punto. *F1-Score* refleja un máximo de rendimiento alrededor de la similitud mínima de 0.7, manteniendo luego un nivel constante. Esto sugiere que el sistema de recomendación ha encontrado un buen equilibrio entre precisión y recall en este punto de similitud mínima.

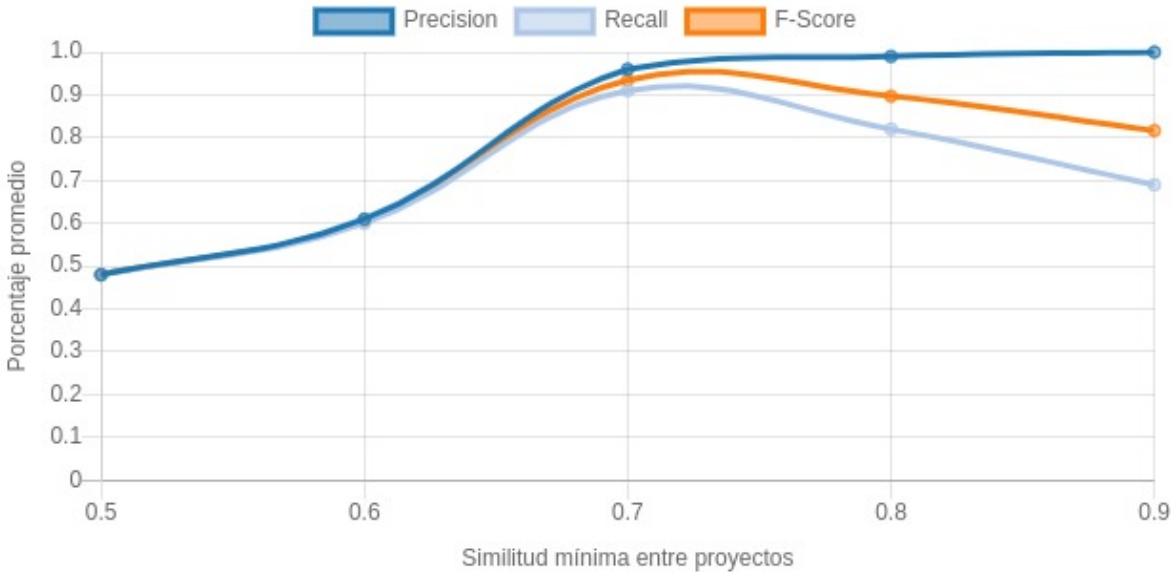

Figura 7.7: Valor promedio de Precision, Recall y F1-Score para K-50

En este gráfico, podemos observar una tendencia similar a las obtenidas para K-5 y K-25.

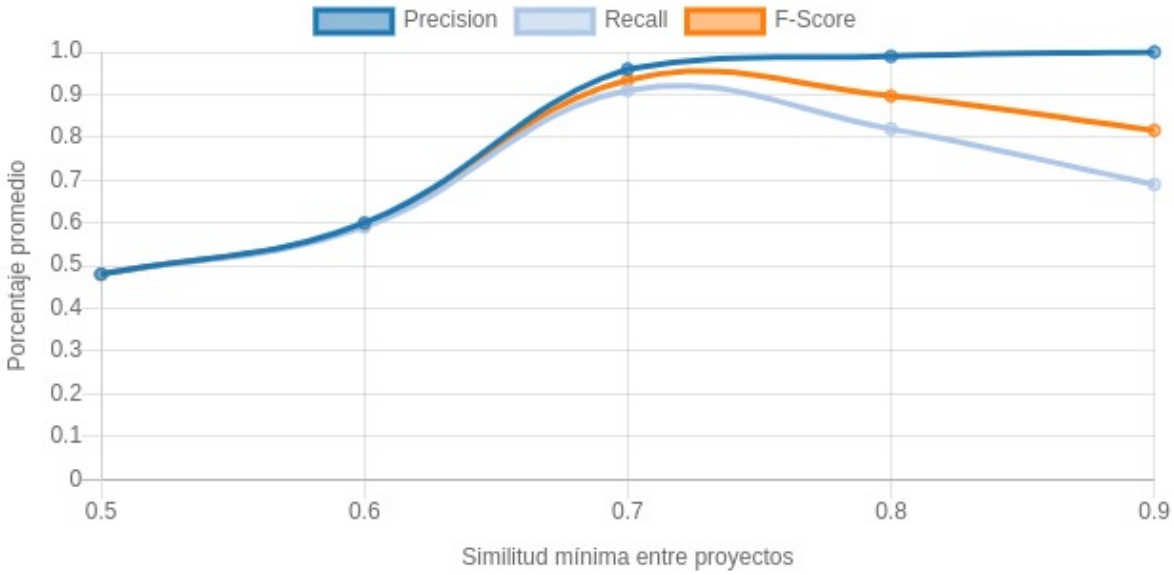

Figura 7.8: Valor promedio de Precision, Recall y F1-Score para K-150

En este gráfico, podemos observar una tendencia similar a las obtenidas para K-5, K-25 y K-150.

7.4.3. Análisis y Conclusiones sobre la Efectividad del Sistema de Recomendación

Se observa que, a medida que aumenta el valor mínimo de similitud entre proyectos, también lo hacen los promedios de *Precision* y *Recall*. Esto resulta lógico, ya que al incrementar el valor de similitud, se reduce el número máximo de proyectos similares identificados. En consecuencia, el algoritmo KNN tiende a encontrar vecinos con una mayor probabilidad de coincidir con los proyectos que se desean recomendar a un usuario. Sin embargo, al aumentar el valor de K, se nota una disminución en los promedios de *Precision* y *Recall*, debido a que un mayor valor de K implica una mayor variedad de proyectos vecinos.

De las pruebas realizadas, se deduce que una configuración eficaz del sistema de recomendación podría ser utilizar un valor de K igual a la raíz cuadrada de N ($K=\sqrt{N}$), donde N es el total de proyectos en el sistema. Este enfoque proporciona un valor de K proporcionalmente pequeño con relación al total de proyectos, ajustándose automáticamente con la adición o eliminación de proyectos. Los gráficos muestran que valores pequeños de K se asocian con un mayor promedio de aciertos. En cuanto al valor mínimo de similitud para calcular los vecinos de los proyectos, se sugiere un valor intermedio para evitar limitarse a proyectos idénticos o con poca similitud, proporcionando así una variedad adecuada de vecinos similares. Se considera que un valor de similitud de 0.65 podría ser adecuado, observándose que con un valor mínimo de 0.7 se registra una mayor precisión en las recomendaciones. Más allá de este punto, no se evidencian mejoras significativas o incluso se presentan disminuciones en la efectividad.

Para evitar el problema del *cold start* para aquellos usuarios que aun no han participado en ningun proyecto, se podría considerar recomendar a estos usuarios los proyectos más actuales, aquellos con mayor número de participantes, los más cercanos a su ubicación geográfica actual o realizar una encuesta de preferencias para seleccionar sus intereses y hacer recomendaciones en base a ellos.

A continuación, se realizará una evaluación del sistema de recomendación utilizando la configuración propuesta para el algoritmo KNN para conocer su *Precision*, *Recall* y *F1-Score*.

7.4.4. Evaluación del Sistema con la Configuración Propuesta

Para evaluar el rendimiento del sistema de recomendación con la configuración propuesta, se llevaron a cabo pruebas adicionales. Estas pruebas consideraron distintas cantidades de proyectos en el sistema, específicamente 50, 300 y 468 proyectos. En cada escenario, el sistema de recomendación se ejecutó 10 veces, generando recomendaciones para los usuarios y calculando el promedio de *Precision*, *Recall* y *F1-Score* en cada ocasión. Posteriormente, se calculó el promedio de estas métricas a lo largo de las 10 ejecuciones para cada cantidad de proyectos evaluada. Los resultados obtenidos se presentan a continuación.

Cabe recordar que la configuración propuesta para el algoritmo KNN es:

- Valor de K: se establece como la raíz cuadrada del número total de proyectos en el sistema ($K = \sqrt{N}$).
- Similitud mínima entre proyectos: se fija en 0.65.

A continuación, se presenta una tabla con los promedios de *Precision*, *Recall* y *F1-Score* obtenidos para cada cantidad de proyectos evaluada:

Cantidad de Proyectos	Precision	Recall	F1-Score
50	0.75	0.75	0.75
300	0.82	0.82	0.82
468	0.82	0.82	0.82

Los resultados promedio mostrados en la tabla corresponden a las pruebas de recomendación realizadas para distintas cantidades de proyectos. Se observa un aumento en los valores de *Precision*, *Recall* y *F1-Score* conforme se incrementa la cantidad de proyectos en el sistema. Esto sugiere que la configuración propuesta del sistema de recomendación presenta un mejor desempeño con una mayor disponibilidad de proyectos para generar recomendaciones.

Estos resultados respaldan la decisión de adoptar $K=\sqrt{N}$ como el valor de K y una similitud mínima de 0.65 entre proyectos. Con esta configuración, se alcanza un balance óptimo entre *Precision* y *Recall* de las recomendaciones, proporcionando así una experiencia de usuario satisfactoria.

Capítulo 8

Conclusión

En esta sección, se presentan las conclusiones obtenidas a partir de la evaluación del sistema de recomendación y se proponen algunas consideraciones para mejorar su desempeño.

8.1. Conclusión

En este trabajo de investigación, se ha llevado a cabo el análisis, diseño e implementación de un sistema de recomendación basado en contenido, con el propósito de mejorar y potenciar la interacción y compromiso de los usuarios en proyectos de ciencia ciudadana. Un aspecto clave de este enfoque ha sido el estudio de las motivaciones tanto de los participantes como de los líderes de proyectos, revelando diferencias en sus objetivos y aspiraciones, lo cual ha enriquecido la perspectiva desde la que se ha desarrollado el sistema. Dicho sistema de recomendación se ha centrado en los participantes y sus intereses para sugerir proyectos para participar.

Al emplear MongoDB y Neo4j en conjunto, se ha aprovechado la estructura flexible y escalable de MongoDB, complementándola con la capacidad de Neo4j para identificar y gestionar relaciones y patrones entre datos. En lugar de optar por un modelo basado en Machine Learning, se optó por Neo4j por su facilidad de adaptación. Este enfoque permite ajustar las recomendaciones simplemente modificando las consultas, ofreciendo mayor flexibilidad ante cambios en los criterios o variables, sin la necesidad de extensos reentrenamientos.

Desde un punto de vista técnico, un sistema de recomendación se compone típicamente de tres elementos clave: *Content Analyzer*, *Profile Learner* y *Filtering Component*. En esta investigación, se ha realizado una descripción completa de ellos,

explicando qué información utilizan y cómo funcionan en conjunto para generar las recomendaciones. Se han detallado las funciones matemáticas necesarias para el cálculo de interés de usuarios en proyectos y se ha enfatizado en el uso del algoritmo KNN, su configuración y su importancia para encontrar proyectos similares dentro del grafo creado en Neo4j.

La evaluación del sistema mediante las métricas de *Precision*, *Recall* y *F1-Score* ha proporcionado una comprensión objetiva y cuantificable de su rendimiento. Aunque los resultados han sido en general alentadores, ciertos retos como el *cold start* han surgido como áreas que podrían beneficiarse de investigaciones y soluciones adicionales en el futuro. Tras las evaluaciones realizadas, la propuesta de usar $K=\sqrt{N}$, donde N es el total de proyectos, para la configuración del algoritmo KNN, ofrece una adaptación dinámica a la escala del sistema y un balance óptimo entre *Precision* y *Recall* en las recomendaciones.

Como conclusión final, esta investigación se distingue en el ámbito de los sistemas de recomendación, es su enfoque integral en la ciencia ciudadana, un área que está en continuo crecimiento. Mientras que la mayoría de los sistemas de recomendación se diseñan con el propósito principal de mejorar la experiencia del usuario en plataformas comerciales o de entretenimiento, esta tesina ha profundizado en cómo las recomendaciones pueden tener un impacto significativo en la participación y compromiso ciudadano en proyectos de su interés, enriqueciendo así la contribución colectiva al conocimiento y la investigación. La combinación de un estudio detallado sobre las motivaciones de los usuarios, junto con la adaptabilidad y precisión técnica proporcionada por Neo4j y el algoritmo KNN, sitúa este trabajo en una posición única, con la capacidad de establecer nuevos estándares y prácticas para los sistemas de recomendación basado en contenido, en el contexto de la ciencia ciudadana.

8.2. Trabajos Futuros

En el transcurso de esta tesina, se ha desarrollado un sistema de recomendación basado en contenido para la aplicación Ágora, centrado en sugerir proyectos a los usuarios en función de sus intereses. El sistema de recomendación propuesto actualmente utiliza tres variables para determinar el interés de los usuarios: el número total de muestras, la fecha de la última muestra y la tasa de actividad. A continuacion, se presentan una serie de mejoras al sistema de recomendación desarrollado.

8.2.1. Optimización

Una estrategia de optimización para acelerar el proceso de generación de recomendaciones sería la pre computación y almacenamiento anticipado de estas en nuestra base de datos. En lugar de calcular las recomendaciones en tiempo real cada vez que se solicitan, se podría diseñar un sistema donde el algoritmo, específicamente el *Filtering Component*, se ejecute en respuesta a determinados eventos (por ejemplo, actualizaciones en el perfil de un usuario o de un proyecto). Al adoptar este enfoque, las relaciones de recomendación entre usuarios y proyectos se almacenarían previamente en la base de datos. Posteriormente, para obtener recomendaciones para un usuario específico, sería suficiente con ejecutar una consulta en Cypher que retorne los proyectos previamente identificados como recomendables para ese usuario.

8.2.2. Inclusión de la Ubicación Geográfica

Una mejora a futuro sería la incorporación de la ubicación geográfica tanto de los proyectos como de los usuarios como una variable adicional. En el modelo de datos propuesto, cada proyecto tiene la opción de definir su alcance geográfico, mientras que los usuarios pueden especificar su ubicación geográfica actual. Teniendo en cuenta estos datos, se podría refinar la lista de proyectos recomendados para incluir solo aquellos que son geográficamente más accesibles para los usuarios.

8.2.3. Recomendación de Usuarios a Proyectos

Otra posibilidad es el desarrollo de un algoritmo que recomiende usuarios a proyectos como posibles participantes. Esto permitiría a los líderes de proyectos acceder a un conjunto de candidatos que podrían ser invitados a colaborar en sus proyectos. En esta variante, se podría hacer uso de evaluaciones (propuesto como parte del modelo de datos, ver 6.2) proporcionadas por los líderes del proyecto a las observaciones de los participantes para mejorar la calidad de las recomendaciones.

8.2.4. Sistema de Recomendación Colaborativo

Un enfoque adicional que puede enriquecer las recomendaciones es la implementación de un sistema de recomendación colaborativo. Este tipo de sistema sugiere proyectos a usuarios basándose en las preferencias o comportamientos de

otros usuarios que tienen intereses similares, lo que podría ampliar la cantidad y diversidad de proyectos recomendados.

8.2.5. Integración con Ágora

Finalmente, hay que recordar que el sistema de recomendación desarrollado es externo a Ágora. Por lo tanto, una tarea pendiente a futuro es la integración de ambos sistemas. Este proceso requerirá modificaciones tanto en el backend como en el frontend de Ágora para gestionar el número de datos adicionales propuestos en el modelo. Además, será necesario importar la información existente de Ágora y desarrollar un mecanismo que permita mantener actualizada la base de datos de Neo4j con respecto a los datos almacenados en MongoDB.

Bibliografía

[1] Charu C. Aggarwal. *Recommender Systems: The Textbook*. Springer International Publishing Switzerland, 2016.

[2] «Bolt Protocol». En: (2021). URL: https://boltprotocol.org/.

[3] «Citizen Science Association». En: (2021). URL: https://citizenscience.org/about/.

[4] «Dataset Metadata Model». En: (2021). URL: https://core.citizenscience.org/docs/dataset.

[5] «Docker». En: (2021). URL: https://www.docker.com/.

[6] «Ionic React». En: (2021). URL: https://ionic.io/resources/articles/ionic-react-vs-react-native#h-what-is-ionic-react.

[7] «K-Nearest Neighbors Algorithm». En: (2021). URL: https://www.ibm.com/topics/knn.

[8] «Keyclaok». En: (2021). URL: https://www.keycloak.org/about.

[9] LIFIA. «Cientopolis». En: (2021). URL: https://www.cientopolis.org/inicio/.

[10] Batta Mahesh. «Machine Learning Algorithms - A Review». En: (2019).

[11] «MongoDB». En: (2021). URL: https://www.mongodb.com.

[12] Alessandro Negro. *Graph Powered Machine Learning*. URL: https://dokumen.pub/graph-powered-machine-learning-1617295647-9781617295645.html.

[13] «Neo4j». En: (2021). URL: https://neo4j.com/docs/getting-started/.

[14] «Neo4j - OGM Object Graph Mapper». En: (2021). URL: https://neo4j.com/developer/neo4j-ogm/.

[15] «Neo4j OGM - Reference». En: (2021). URL: https://neo4j.com/docs/ogm-manual/current/reference/.

[16] «Non-Personalized Recommender Systems and Userbased Collaborative Recommender Systems». En: (2014). URL: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=44af1b6b1cdf467bf5df8101e8ce9ec673023982.

[17] «Observation Data Model». En: (2021). URL: https://core.citizenscience.org/docs/observation.

[18] «One-Hot Encoding». En: (2021). URL: https://en.wikipedia.org/wiki/One-hot.

[19] «OpenAPI». En: (2021). URL: https://oai.github.io/Documentation.

[20] Marco de Gemmis Pasquale Lops y Giovanni Semeraro. «Content-based Recommender Systems: State of the Art and Trends». En: (2011). URL: https://www.researchgate.net/publication/226098747_Content-based_Recommender_Systems_State_of_the_Art_and_Trends.

[21] Víctor Yepes Piqueras. «Proceso Analítico Jerárquico (Analytic Hierarchy Process, AHP)». En: (2018). URL: https://victoryepes.blogs.upv.es/2018/11/27/proceso-analitico-jerarquico-ahp/.

[22] Ivens Portugal, Paulo Alencar y Donald Cowan. «The Use of Machine Learning Algorithms in Recommender Systems: A Systematic Review». En: *Expert Systems with Applications* (2015). DOI: 10.1016/j.eswa.2017.12.020. URL: https://www.researchgate.net/publication/284219925.

[23] «Powering Real-Time Recommendations with Graph Database Technology». En: (2021). URL: https://neo4j.com/whitepapers/recommendations-graph-database-business/.

[24] «PPSR Core Documentation». En: (2021). URL: https://core.citizenscience.org/docs/.

[25] «PPSR (Public Participation in Scientific Research) Core». En: (2021). URL: https://core.citizenscience.org/.

[26] «Precision and recall». En: (2022). URL: https://en.wikipedia.org/wiki/Precision_and_recall.

[27] «Project Metadata Model». En: (2021). URL: https://core.citizenscience.org/docs/project.

[28] «Quarkus». En: (2021). URL: https://www.redhat.com/en/topics/cloud-native-apps/what-is-quarkus.

[29] «Quarkus, Helidon, Micronaut». En: (2021). URL: https://neo4j.com/developer/java-frameworks/#quarkus-integration.

[30] «ReactJS». En: (2021). URL: https://reactjs.org/.

[31] Thomas L. Saaty. *The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. McGraw-Hill, 1980.

[32] Katrin Vohland y col. *The Science of Citizen Science*. Springer, 2021.

[33] «What is a Graph Database?» En: (2021). URL: https://neo4j.com/developer/graph-database/.

[34] «What Is a Recommendation Engine and How Does It Work?» En: (2021). URL: https://www.appier.com/blog/what-is-a-recommendation-engine-and-how-does-it-work.

[35] «Why Neo4j? Top Ten Reasons». En: (2021). URL: https://neo4j.com/top-ten-reasons/.

[36] Rui Song Xin Dong Tong Li y Zhimig Ding. «Profiling users via their reviews: an extended systematic mapping study». En: (2020). URL: https://doi.org/10.1007/s10270-020-00790-w.

[37] Yang Zhang y col. «How to Retrain Recommender System? A Sequential Meta-Learning Method». En: (2020).