Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Máster Universitario en Inteligencia Artificial

Trabajo Fin de Máster

Exploración de Sistemas Recomendadores para la Recomendación de Propuestas en Organizaciones Autónomas Descentralizadas

Autor: David Davó Laviña
Tutores: Damiano Zanardini & Javier Arroyo

Madrid, Mayo 2024

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Máster
Máster Universitario en Inteligencia Artificial

Título: Exploración de Sistemas Recomendadores para la Recomendación de Propuestas en Organizaciones Autónomas Descentralizadas

Mayo 2024

Autor: David Davó Laviña
Tutores: Damiano Zanardini
Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Javier Arroyo
Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Agradecimientos

A mi pareja por estar siempre a mi lado, apoyarme en los momentos más difíciles y recordarme que es necesario descansar de vez en cuando.

A mi familia por su respaldo incondicional y por inspirarme a ser curioso e ingenioso.

A mis tutores actuales por su orientación, apoyo, y grandes ideas, fundamentales para empezar y terminar este trabajo. Y a todos los buenos profesores y educadores del pasado que creyeron en mí, sin los que no me habría sido posible llegar hasta aquí.

Al personal de la facultad: cafetería, limpieza, biblioteca, administración, mantenimiento, etc. por darnos un buen lugar en el que trabajar estando cómodos.

A todas las personas que resuelven dudas y preguntas desinteresadamente en foros y Discords y Slacks, y dejan tras de sí un rastro de conocimiento experto que no puede encontrarse en ninguna biblioteca.

A todos aquellos gigantes y hackers sobre los que se apoya nuestro trabajo.

Finalmente, al Instituto de Tecnología del Conocimiento por proporcionar los recursos computacionales necesarios para el desarrollo de este trabajo.

Partes de este Trabajo de Fin de Máster se han realizado bajo el proyecto «Evaluación de Organizaciones Autónomas Descentralizadas basadas en Blockchain para la gestión de proyectos DeFi, NFT y Metaverso», PID2021-127956OB-I00, financiado por el Ministerio de Ciencia e Innovación y cofinanciado por la Unión Europea.

Mis más sinceros agradecimientos

Resumen

Las Organizaciones Autónomas Descentralizadas (*Decentralized Autonomous Organizations, DAOs*) han surgido como un nuevo enfoque hacia la gobernanza colectiva, facilitado por la tecnología blockchain. Las DAOs fomentan procesos de votación democráticos, permitiendo a los miembros proponer y votar sobre propuestas, moldeando así colectivamente el futuro de la organización. Sin embargo, la efectividad y legitimidad de la toma de decisiones dentro de las DAOs puede verse afectada por la baja participación de los votantes, un desafío común con otras comunidades en línea y los sistemas de votación tradicionales. Dada el tiempo limitado de votación de cada propuesta, las técnicas convencionales en sistemas recomendadores son inadecuadas. Por esa razón, este Trabajo de Fin de Máster introduce el primer sistema de recomendación diseñado específicamente para DAOs. En particular, se han desarrollado nuevas técnicas de validación y una nueva linea base. Este enfoque ha sido probado en tres modelos: un primero basado en filtrado colaborativo que utiliza Redes Neuronales de Grafos para explotar el grafo bipartito miembro-propuesta formado por la votación en DAOs; un segundo basado en contenido que utilizando Procesamiento del Lenguaje Natural; y un tercer modelo híbrido que combina los resultados de los dos anteriores. Aunque el proyecto se ha realizado con la organización de Decentraland en mente, que gobierna colectivamente una plataforma de metaverso con más de 35 000 miembros y 2 000 propuestas votadas, también se ha probado y comparado con otras organizaciones con diferentes características. Estos resultados no solo demuestran el potencial de los sistemas de recomendación para mejorar la personalización de propuestas y mejorar la participación de los votantes en las DAOs, sino que también se alinean con las mejoras observadas en la participación de los usuarios en otros proyectos colaborativos en línea que han implementado sistemas similares. Además, el sistema de recomendación basado en GNN con restricciones temporales podría ser adaptado a contextos similares, como la recomendación de eventos.

Palabras clave: Organizaciones Autónomas Descentralizadas, Sistemas Recomendadores, Redes Neuronales de Grafos, Procesamiento de Lenguaje Natural, Blockchain

Abstract

Decentralized Autonomous Organizations (DAOs) have emerged as a novel approach to collective governance, facilitated by blockchain technology. DAOs foster democratic voting processes, allowing members to put forward and vote on proposals, thereby collectively shaping the organization's future. However, the effectiveness and legitimacy of decision-making within DAOs can be compromised by low voter turnout, a challenge shared with traditional online communities and voting systems. Given the limited lifespan of each proposal, conventional recommender system techniques are unsuitable. In response to these issues, this Master's Thesis introduces a recommender system specifically designed for DAOs. In particular, new validation techniques and a baseline had to be developed. The approach has been tested on three models, a model that leverages Graph Neural Networks (GNN) for collaborative filtering, effectively exploiting the member-proposal bipartite graph inherent in DAOs voting, a second one that utilizes Natural Language Processing as a content-based approach, and a third hybrid model that combines the results of the previous two. While the project has been made with the Decentraland DAO organization in mind, which collectively governs a metaverse platform with over 35,000 members and 2,000 proposals voted, it has also been tested on and compared with other organizations with different characteristics. We compare our approach with a baseline that recommends the most popular open proposals at the time of recommendation. These models accurately predict future voters, surpassing the proposed baseline. These results not only underscore the potential of recommender systems in enhancing voter participation within DAOs but also align with the observed improvements in user engagement in other online collaborative projects that have implemented similar systems. Furthermore, our GNN-based recommendation systems with temporal constraints could be adapted to other settings such as event recommendation.

Keywords: Decentralized Autonomous Organizations, Recommender Systems, Graph Neural Networks, Natural Language Processing, Blockchain

Tabla de contenidos

Índice de figuras

Capítulo 1

Introducción

Una Organización Autónoma Descentralizada (DAO, Decentralized Autonomous Organization) es un sistema basado en blockchain que opera de forma autónoma y permite tomar decisiones mediante contratos inteligentes, sin necesidad de una autoridad central [1]. Son una innovadora forma de gobernanza que permite a las comunidades decidir colectivamente sobre asuntos que las afecten.

Según el portal de analíticas Deepdao[1], hay aproximadamente 20 000 DAOs con una tesorería total valorada en más de 37.1 miles de millones de dólares estadounidenses, y con una base de usuarios activos de sobre 3 millones de votantes y contribuidores de propuestas. La escala de estas organizaciones varía enormemente, con DAOs pequeñas con una docena de miembros, a otras con millones de usuarios. La mayoría de ellas son pequeñas, con solo 400 DAOs de más de 100 miembros recogidas por Deepdao.

En principio, son horizontales y cualquiera puede realizar propuestas que serán votadas por los miembros de la organización. Sin embargo, estas plataformas enfrentan importantes retos: la distribución desigual del poder de votación y la disparidad en la participación [2], problema que caracteriza a las comunidades en línea [3], y en común con la baja participación en las elecciones tradicionales [4].

La baja participación es un tema complejo y difícil de analizar, pero una posible hipótesis es que en las organizaciones más grandes a los usuarios puede resultarles inabarcable el volumen de propuestas abiertas cada día, dificultando que tomen un rol activo en su gobernanza. «Es necesario tener tiempo, información, experiencia y reputación para obtener poder de decisión. Cuanto más altos sean los costes, menos gente querrá participar, lo que en la práctica contribuirá a la descentralización» [5].

La falta de atención en estas organizaciones es un problema abierto [6] que amenaza las aspiraciones originales de descentralización [7], y que puede incluso usarse para explotar los requisitos de mayoría simple para drenar los fondos de una organización, habiendo cientos de millones de dólares en riesgo [8].

La solución propuesta en este Trabajo de Fin de Máster es la de implementar un Sistema Recomendador, encargado de hacer esa curación de propuestas personalizada y de manera automática, complementando a los humanos encargados de esas

[1] `https://deepdao.io`. Accedido Feb. 15, 2024.

tareas y haciendo el sistema más resiliente a interferencias externas. Además, simplifica la experiencia del usuario al facilitarle propuestas en las que votar basándose en sus intereses e historial. Esta mejora podría incluso aumentar la participación, consiguiendo una representación más fiel de la opinión de los usuarios.

Aunque este sistema ha sido aplicado al contexto de las DAOs debido a la disponibilidad del conjunto de datos, es una prueba de concepto de un posible sistema similar para otras plataformas de trabajo colaborativo. Este enfoque podría ser extrapolado con éxito a entornos análogos, como son las plataformas de participación ciudadana, donde los ciudadanos pueden ejercer su voto en propuestas para que las implemente el ayuntamiento.

1.1. Objetivos

El objetivo principal de este trabajo de fin de máster consiste en la creación y evaluación de un sistema recomendador para propuestas en Organizaciones Autónomas Descentralizadas. Se proponen varios sistemas usando distintas técnicas y algoritmos, y se evalúa cada uno de ellos, habiendo sido necesario crear un marco de técnicas de evaluación específico para este caso concreto debido a la naturaleza temporal de los elementos a recomendar. Se definen los siguientes objetivos:

- Conocer el funcionamiento de los sistemas recomendadores y librerías para su implementación.
- Entender las peculiaridades de la aplicación al campo de las DAOs y exponer las diferencias con un sistema recomendador clásico.
- Crear un marco de técnicas de evaluación específico para este caso, habiendo considerado la naturaleza temporal de los elementos a recomendar en las DAOs.
- Diseñar y programar un sistema híbrido, basado en contenido y filtrado colaborativo, realizando una evaluación realista del posible despliegue del sistema.
- Obtener conclusiones y proponer posibles mejoras para una posible aplicación del sistema.

1.2. Estructura

Esta memoria se divide en cuatro partes. Una primera que contiene esta introducción; una segunda que contiene el estado del arte y definiciones previas necesarias para definir el problema; una tercera parte en la que se presenta el problema, un marco de validación, y se definen los experimentos; y una última parte que presenta los resultados del sistema y conclusiones.

Capítulo 2

Fundamentos tecnológicos

El objetivo de este capítulo es presentar una introducción y unos fundamentos sobre la tecnología que da soporte a las organizaciones sobre las que se ejecutará el sistema recomendador, pues se considera que es importante entender sus limitaciones y el valor de estas tecnologías.

2.1. Blockchain

Un *blockchain* (cadena de bloques) es un libro contable o *ledger* distribuido que almacena una lista de transacciones. Cada bloque está firmado digitalmente utilizando los datos del bloque anterior, formando una cadena de tal manera que, de modificar un bloque, cambiaría la firma de todos los posteriores. Al estar esta lista distribuida, si uno de los nodos que almacena la lista cambia los datos, se produciría una incongruencia que sería detectada por el resto de nodos, por lo que podemos considerar el blockchain como una base de datos distribuida e inmutable, a la que solo se le pueden añadir datos.

Para realizar una nueva operación, el usuario utiliza una clave privada para firmar la transacción y la envía a algún nodo que participe en la red para que verifique la firma utilizando su clave pública, y se añada a la cadena. Sin embargo, los nodos que participan en la red necesitan de cierta comisión para realizar la operación. El *hash* de la clave pública es la dirección que identifica al usuario, y que deberá usarse para realizar una transferencia. El programa o dispositivo que almacena esta clave privada se le conoce como cartera o *wallet*, por lo que es común utilizar indistintamente este término para identificar a los usuarios. Nótese que al igual que en otras plataformas hay ciertos usuarios con varias cuentas, en blockchain también hay usuarios con varios *wallets*, es decir, varias direcciones.

Tras procesar todas las transacciones realizadas desde el comienzo de la cadena de bloques, se obtiene el estado actual, con las cantidades obtenidas por cada usuario.

En 2008, una persona o grupo de personas conocido por el seudónimo de Satoshi Nakamoto desplegó satisfactoriamente Bitcoin, creando así la primera criptomoneda. Años después, en 2015 se lanzaría Ethereum [9], un blockchain desarrollado por el programador Vitalik Buterin que añadiría la revolucionaria capacidad de desplegar aplicaciones descentralizadas.

El conjunto de nodos participantes en la red de Ethereum forma una máquina virtual determinista basada en pila conocida como Máquina Virtual de Ethereum (EVM, Ethereum Virtual Machine). Esta máquina virtual ejecuta contratos inteligentes o *smart contracts* que son desplegados por los usuarios. En este caso, una dirección de Ethereum puede representar tanto a un usuario como a un contrato inteligente, pero como estos últimos tienen las mismas capacidades que cualquier usuario, no se suele realizar ninguna distinción y se les considera simplemente otro tipo de agentes dentro del sistema.

Los contratos inteligentes se programan en un lenguaje de alto nivel y son compilados a *bytecode* turing-completo del EVM, por lo que aunque su binario es público, el código fuente no lo es. Es común publicar el código fuente y las opciones del compilador utilizado para que los usuarios puedan verificar que compila al binario desplegado. Al igual que en Bitcoin, los usuarios deben pagar una comisión a los nodos para que su interacción con el contrato inteligente sea ejecutada.

Una de las mayores aplicaciones de este tipo de contratos es Finanzas Descentralizadas (DeFi, Decentralized Finance), con la que se ofrecen diversos instrumentos financieros: préstamos, bonos, bolsas de valores, *crowdfunding*... Sin embargo, las aplicaciones descentralizadas no se limitan sólo a las finanzas, también hay aplicaciones de juegos, energía, salud, ciencia, multimedia, entre otras [10].

Para facilitar su uso, las aplicaciones descentralizadas utilizan una interfaz web a la que cualquiera puede acceder y leer los datos, aunque es necesario una extensión de navegador que haga de cartera para poder interactuar.

2.2. Organizaciones Autónomas Descentralizadas

Una Organización Autónoma Descentralizada (DAO, Decentralized Autonomous Organization) es un sistema basado en blockchain que permite a un grupo de personas coordinarse y auto-gobernarse mediante contratos inteligentes, y cuya gobernanza es descentralizada [1]. El término fue inventado en 2014 por Vitalik Buterin, programador de Ethereum [11].

En la práctica, una DAO es un conjunto de contratos inteligentes que permiten a un conjunto de usuarios votar en propuestas, que tienen como efecto el realizar transacciones con su tesorería común, aceptar nuevos miembros, o incluso hacer llamadas a otros contratos inteligentes. En algunas organizaciones, es posible realizar propuestas aunque no se sea miembro, aunque sólo los miembros pueden votar.

Estas organizaciones han sido utilizadas para diversos propósitos, como votar propuestas de micromecenazgo y asignar presupuestos para invertir en la comunidad de manera participativa, similar a los presupuestos participativos. Además, existen organizaciones encargadas de administrar protocolos descentralizados, como las DAOs de Uniswap o Decentraland, así como de realizar compras de gran valor de manera conjunta, como el caso destacado de ConstitutionDAO, que recaudó 47 millones de dólares para adquirir una de las copias de la constitución estadounidense [12]. También se encuentran organizaciones más pequeñas que sirven para gestionar y administrar colectivos de desarrolladores o artistas, adoptando una estructura similar a la de una cooperativa [13].

Debido a que es posible que una única persona tenga múltiples direcciones, en lugar

de utilizar el esquema de un voto por persona, la organización asigna un poder de votación (VP, voting power) a cada usuario. La manera de asignar el VP varía en cada comunidad, pero puede estar vinculado a la aportación de capital, a la realización de trabajos previos, o incluso ser asignada mediante propuestas de petición de VP.

Debido a la inherente transparencia de Ethereum, el resultado de la votación y qué votó cada uno de los usuarios es siempre visible. Aunque podría ofuscarse o, incluso, encriptarse, las plataformas lo muestran sin ningún problema en la interfaz web.

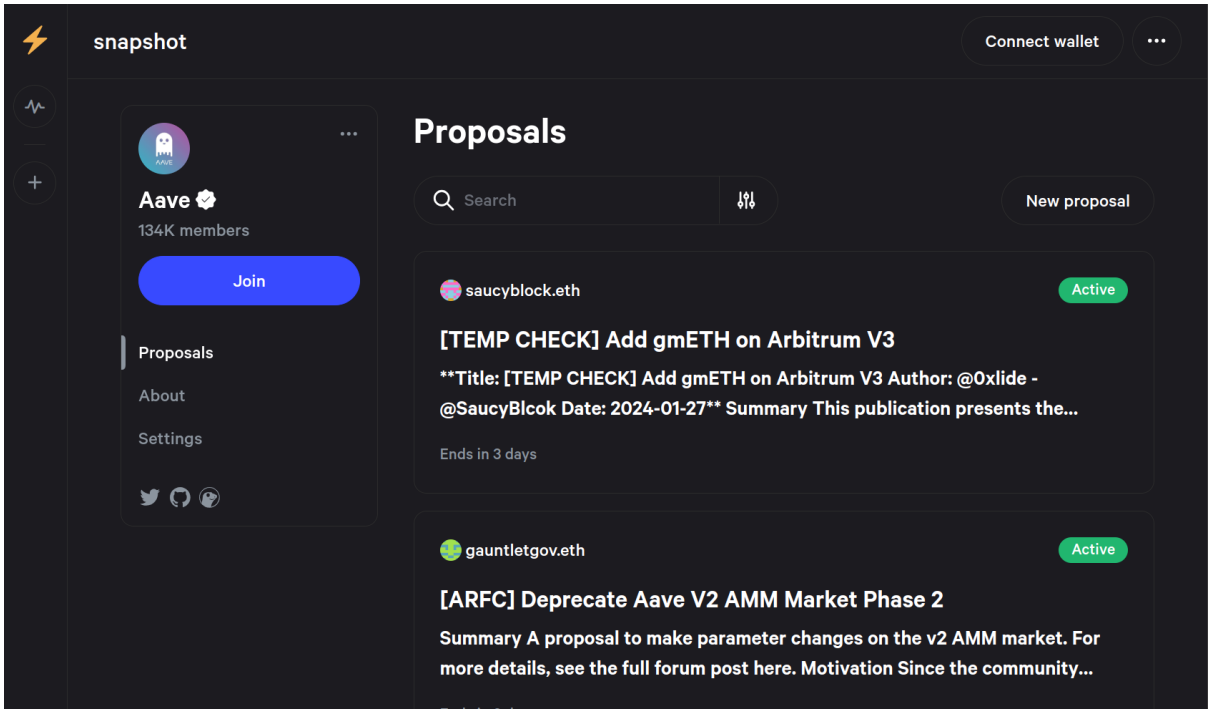
Para evitar tener que programar una DAO desde cero, surgieron distintas plataformas para facilitar su despliegue a través de plantillas y sin necesidad de conocimientos de programación [14], dando lugar a un auge en la creación de organizaciones, llegando actualmente a registrarse más de 3 millones de miembros activos y manejando más de 30 mil millones de dólares estadounidenses en sus tesorerías, según el portal de analíticas Deepdao [15].

Aun así, las comunidades sufren de distintos problemas de seguridad, eficiencia, efectividad, y descentralización [16, 17], que intentan paliar utilizando métodos de votación distintos a la mayoría absoluta o relativa [18]. Estos sistemas pueden llegar a ser muy complejos y difíciles de comprender por los usuarios, dificultando la participación.

Algunas comunidades intentan paliar estas cuestiones con medios externos a la blockchain. Por ejemplo, una práctica común es discutir minuciosamente una propuesta en foros o chats antes de subir la propuesta formalmente a la plataforma, asegurándose así de que será aprobada. Sin embargo, se reduce la transparencia del sistema pues no toda la comunidad participa en dichos canales, y algunos de ellos son jardines vallados a los que es imposible acceder sin registro previo, dependientes de una entidad central. Otros sistemas de votación permiten *delegar* tu poder de votación, confiando en el juicio de individuos de confianza. En ambos casos, es necesario que al menos un usuario asuma la responsabilidad de estar al tanto de todas las propuestas, filtrarlas, y difundir su información o actuar de manera congruente. La ausencia de este usuario en momentos críticos de votación, o su decisión de modificar radicalmente su postura, podría desencadenar consecuencias adversas para la organización. Por ejemplo, si se lleva a cabo una votación para remunerar a un individuo que ha contribuido a la DAO, y dicha propuesta no obtiene aprobación, la confianza de otros colaboradores puede disminuir, desincentivando futuras colaboraciones.

Para mejorar la participación y disminuir los efectos del precio de Ethereum en la comunidad [19], se ha buscado utilizar cadenas alternativas compatibles con la EVM y, por lo tanto, compatibles con el código de los contratos inteligentes desarrollados previamente. Estas cadenas suelen tener unos costes de operación menores, y un tiempo de bloque menor (es decir, tarda menos en ejecutarse cada operación), o están basadas en criptomonedas con precios estables, mejorando notablemente la usabilidad y obteniendo una rápida adopción.

Finalmente, en 2021 se lanzó la plataforma *off-chain* y, por lo tanto, sin costes de operación, pero manteniendo la seguridad y descentralización necesarias para las DAOs llamada Snapshot. Para mantener la descentralización, los votos están firmados digitalmente usando la clave privada del usuario, y tanto la interfaz como los resultados son almacenados en InterPlanetary File System (IPFS), un protocolo de hi-



Fuente: `https://snapshot.org/#/aave.eth`
Figura 2.1: Captura de pantalla de la interfaz web de la plataforma Snapshot para la gobernanza de la organización Aave.

pertexto descentralizado. Sigue teniendo las mismas capacidades de conectarse con una tesorería, realizar transferencias y ejecutar cualquier contrato inteligente, por lo que a efectos prácticos es como una DAO sin blockchain y mejor usabilidad. En la figura 2.1 se muestra una captura de la interfaz web.

Esta plataforma elevó en varios órdenes de magnitud el alcance de las DAOs, alojando actualmente más de 30 mil organizaciones que suman más de 3 millones de usuarios y 200 mil propuestas realizadas [20].

Capítulo 3

Sistemas Recomendadores

3.1. Trabajo relacionado

Hasta donde sabemos, hasta ahora nadie ha estudiado el uso de Sistemas Recomendadores para DAOs. Por lo tanto, en esta sección se proporcionará una visión general de la aplicación de Sistemas Recomendadores en áreas cercanas.

Los Sistemas Recomendadores han sido usados para abordar el problema de la participación en proyectos colaborativos. Por ejemplo, en Wikipedia se han usado para recomendar tareas y artículos para traducir. SuggestBot, uno de los primeros de estos sistemas desplegado en 2007, realiza asignación de tareas (limpieza, arreglar formato, eliminar o fusionar artículos...) de manera personalizada usando filtrado colaborativo [21]. Para llenar las lagunas en la traducción de artículos de manera eficiente y recomendar artículos que crear en otros idiomas, [22] ordena los artículos por relevancia en cada idioma y después los asigna a editores basándose en el texto de sus contribuciones previas. WikiRecNet [23] es un sistema creado sobre *representation learning*, con *Doc2Vec* y *GraphSAGE* (una Graph Convolutional Network (GCN)) para ayudar a los editores a lidiar con el gran volumen de artículos potenciales que pueden requerir de su atención. Notablemente, estos sistemas consiguieron satisfactoriamente mejorar las contribuciones de los miembros.

Este tipo de sistemas también han sido usados en plataformas de desarrollo colaborativo como Bugzilla o GitHub para realizar triaje de *bugs*, inicialmente en 2004 utilizando *Naïve Bayes* [24], y más recientemente utilizando técnicas más modernas como *Hierarchical Attention Networks* [25]. Estos sistemas también se han usado para ampliar el número de revisores recomendando desarrolladores adecuados que hayan trabajado en los mismos ficheros o similares [26, 27, 28] e incluso teniendo en cuenta el tiempo como contexto para realizar la recomendación usando Graph Neural Network (GNN) [29], o asignar *bugs* a desarrolladores basándose en elementos textuales de *bugs* arreglados en el pasado [30, 31], entre otras muchas tareas.

Por otro lado, podemos coger inspiración del uso de sistemas recomendadores en democracia electrónica o *e-Democracy*. En elecciones tradicionales en las que se elige un partido o candidato en un momento concreto, se han utilizado *Voting Advice Applications* para abordar el problema de la participación [32]. Sin embargo, estas aplicaciones principalmente asisten al usuario a elegir un candidato próximo a sus tendencias y preferencias [33]. Por ejemplo, Terán y Meier [34] provee información

sobre candidatos próximos a las tendencias del usuario, y Buryakov et al. [35] utiliza datos abiertos del gobierno para facilitar la creación automática de estos sistemas, que normalmente utilizan datos de encuestas. En cambio, nuestro sistema trataría de recomendar propuestas que los usuarios puedan encontrar interesantes, sin dirigir a los usuarios hacia una opción en concreto de la propuesta, y sin tener en cuenta sus elecciones anteriores.

El trabajo más cercano en este propósito sería el uso de sistemas recomendadores en plataformas de participación ciudadana [36] y presupuestos participativos [37]. En estas plataformas los ciudadanos deciden qué propuestas la administración debería priorizar, normalmente las que cuentan con más apoyos. Las propuestas que son ignoradas no tienen impacto, mientras que en el contexto de las DAOs todas las propuestas deberían ser tenidas en cuenta. Además, ha de tenerse en cuenta de que los usuarios no pueden votar en contra de una propuesta, y las propuestas suelen permanecer abiertas durante un tiempo indefinido o muy largo, pues su implementación suele llevarse a cabo en los próximos meses o años y no inmediatamente, a diferencia de en las DAOs.

Finalmente, la característica temporal de las propuestas en las DAOs nos lleva al dominio de la recomendación de eventos en Redes Sociales Basadas en Eventos (EBSN, Event-Based Social Networks), fijándonos sobre todo en la evaluación de dichos sistemas. Zhang y Wang [38] utilizan un modelo bayesiano, pero no parecen tener en cuenta el tiempo. Pham et al. [39] utilizan grafos heterogéneos y consideran el tiempo (día de la semana, hora del día) una parte esencial del contexto para realizar la recomendación, sin embargo, no se tiene el tiempo en cuenta a la hora de evaluar el sistema. Minkov et al. [40] recomiendan conferencias científicas utilizando filtrado colaborativo. Se evalúa la recomendación de ítems de los que no existe feedback previo, y se realiza un estudio con usuarios obteniendo feedback explícito y simulando recomendaciones semanales. En el caso de feedback implícito, Macedo, Marinho y Santos [41] realizan una especie de validación cruzada dividiendo los datos históricos en entrenamiento y test según un momento temporal. Sin embargo, estos sistemas tienden a depender demasiado del contexto para realizar las recomendaciones, mientras que en nuestro caso solo contamos con la información textual de las propuestas.

Cabe mencionar que, aunque sí se han utilizado técnicas de aprendizaje automático e inteligencia artificial para mejorar las aplicaciones basadas en blockchain [42], este parece ser el primer trabajo de aplicación de sistemas recomendadores a este tipo de organizaciones.

3.2. Librerías de sistemas recomendadores

En esta sección se realiza una exploración de potenciales librerías de sistemas recomendadores para la implementación del sistema propuesto en este trabajo. Se han buscado librerías en *GitHub* y *Papers With Code*, desarrolladas en los lenguajes de programación Python y/o R, que cuenten con una sólida base de usuarios y una documentación exhaustiva, y que al mismo tiempo incorporen los algoritmos estado del arte. A ser posible, basados en GNN.

A continuación se detallan las librerías consideradas durante esta exploración, sin seguir un orden específico:

Spotlight [43] Con 2.9k estrellas en GitHub, es una de las librerías de sistemas recomendadores más famosas, actualmente está basada en PyTorch [44] y su objetivo es ser una herramienta para la exploración rápida y el prototipado de nuevos modelos, utilizando sus funciones de pérdida, herramientas de validación cruzada y evaluación, o capas creadas. Sin embargo, también incluye implementaciones de modelos de factorización, basados en representaciones latentes, o incluso de secuencia. No obstante, no contiene modelos basados en grafos, y lleva 4 años sin recibir actualizaciones.

Surprise [45] Es un scikit (add-on basado en SciPy [46]) para sistemas recomendadores que hace gala de su extensa documentación y comunidad, con 6k estrellas en GitHub. Tiene interfaces muy similares a las de scikit-learn, por lo que es fácil de aprender a usar, y cuenta con varios algoritmos de filtrado colaborativo y hace sencillo el implementar nuevos algoritmos. Sin embargo, no cuenta con nada para sistemas basados en contenido.

Implicit [47] Como su nombre indica, esta librería se especializa en conjuntos de datos de feedback implícito como es nuestro caso. Su mayor fortaleza parece ser la rápida ejecución de sus modelos, algunos optimizados con código propio en CUDA. Sin embargo, al igual que surprise, no tiene ningún modelo basado en contenido.

Crab [48] Es otro scikit como surprise, pero creado mucho tiempo antes, en 2010. No cuenta con actualizaciones desde hace 13 años, por lo que se ha descartado su uso para evitar problemas de compatibilidad de dependencias. Sin embargo, se mantiene en esta lista pues tiene pocas dependencias y la implementación de algunos de los algoritmos podría servir de material de referencia.

Pytorch Geometric [49] Su repositorio en GitHub cuenta con 20 mil estrellas, y está respaldada por la extensa comunidad de PyTorch [44], por lo que contará con actualizaciones constantes a lo largo del tiempo. Aunque cuenta con muy buenas herramientas para GNN, al comienzo de este trabajo no era factible usarla para sistemas recomendadores, pues habría que definir las métricas offline de evaluación, entre muchas otras cosas. En un principio se decidió usar esta librería, pero al tratarse de un nuevo conjunto de datos que no se ha probado anteriormente en sistemas recomendadores, con peculiaridades que lo hacen muy distinto a otros sistemas recomendadores (véase el capitulo 4), se encontraron muchos problemas de desarrollo. Al tener que crear tanto código nuevo (conversión de datos a grafo, sampling, evaluación...), y tratarse de una librería en muy bajo nivel, era muy difícil depurar el código. En la versión 2.5.0 de febrero 2024 añadieron soporte para sistemas recomendadores, implementando métricas de recuperación como la precisión en k o el nDCG.

Recommenders: Best Practices on Recommendation Systems [50] (anteriormente Microsoft's Recommenders). Librería creada por Microsoft con implementación de varios modelos de filtrado colaborativo, basado en contenido e híbridos. También cuenta con funciones de preparación de datos, métricas de evaluación offline, y herramientas para el despliegue de los modelos en un entorno de producción en Azure. De entre todas las librerías exploradas, consideramos que esta es la más completa y más adecuada para nuestro caso de uso.

LibRecomender [51] Aunque su uso aún no está muy extendido y su versión 1.0

se lanzó en 2023, es una librería muy prometedora. Su enfoque integral abarca todo el proceso, desde el entrenamiento hasta el despliegue de los modelos. Cuenta con más de 20 modelos implementados en Tensorflow y PyTorch a su disposición, entre ellos muchos basados en GNN como Deepwalk, LightGCN, NGCF o GraphSAGE, así como modelos híbridos que permiten tener en cuenta la información textual de las propuestas. También cuenta con soporte para características dinámicas y recomendaciones secuenciales. La única razón por la que no se eligió utilizar esta librería es porque se desconocía su existencia al comenzar el trabajo, y sólo se descubrió al realizar una revisión del estado del arte.

Finalmente, se decidió utilizar la librería Microsoft Recommenders [50] para realizar un prototipado rápido y poder experimentar con varios modelos. Además, el repositorio cuenta con multitud de ejemplos de cada modelo que ilustran buenas prácticas y recomendaciones en el diseño de sistemas recomendadores.

3.3. Modelos

En esta sección se presentan los diferentes modelos de sistemas recomendadores que han sido considerados para este trabajo, junto con cierto contexto que permita la comprensión de su funcionamiento.

En sistemas recomendadores, podemos distinguir dos tipos de modelos según los datos con los que trabajen. Los modelos basados en Filtrado Colaborativo (CF, Collaborative Filtering) se basan en las interacciones entre usuarios e ítems para realizar las recomendaciones [52], mientras que los modelos basados en contenido utilizan otros atributos de los usuarios e ítems, tal como la información textual, etiquetas, o ubicación. Los modelos híbridos combinan ambos enfoques para mejorar las recomendaciones y reducir sus problemas.

Aunque también existen sistemas recomendadores basados en conocimiento, no se tratan en este trabajo debido a que es necesario cierta interactividad con la que no contamos.

Por otro lado, los sistemas recomendadores dependen de la fuente de los datos de entrada que se emplean para realizar las recomendaciones. En el *feedback explícito* el usuario da una opinión directa sobre la interacción, como puede ser seleccionar «me gusta» o «no me gusta», dar estrellas, o escribir una reseña. Por otro lado, en el *feedback implícito* se observa el comportamiento del usuario, como puede ser el historial de acciones del usuario o el tiempo dedicado a un contenido [53]. Como se detalla en el capítulo 4, los datos utilizados en este trabajo son feedback implícito, por lo que en el resto de esta sección nos centraremos en modelos que acepten este tipo de datos.

3.3.1. Filtrado colaborativo

La idea principal de este tipo de modelos es utilizar la similitud entre las preferencias de distintos usuarios. Los métodos de recomendación basados en filtrado colaborativo se pueden dividir en basados en memoria y basados en modelos.

En los métodos basados en memoria, también llamados basados en vecindario, el

score para cada par ítem-usuario se genera basándose en los vecinos inmediatos. En concreto, si se genera basándose en los ítems con los que interactúa cada usuario, se denomina user-based collaborative filtering (UserCF), y si se utilizan los vecinos de cada ítem se denomina item-based collaborative filtering (ItemCF).

Por otra parte, los métodos basados en modelos utilizan modelos de predicción basados en aprendizaje automático y minería de datos, aprendiendo los parámetros necesarios para minimizar una función de pérdida con unos datos de entrenamiento.

Las implementaciones más sencillas de ItemCF y UserCF con feedback explícito crean una matriz de ítems-ítems o usuarios-usuarios completa y completan los espacios vacíos de la matriz utilizando las similitudes entre filas, por ejemplo, mediante la similitud del coseno [51].

3.3.1.1. Factorización de Matrices

Para mejorar tanto la eficiencia como la calidad de las recomendaciones, se utilizan métodos de reducción de la dimensionalidad, representando los ítems y usuarios en un espacio latente de menos dimensiones. La principal familia de este tipo de métodos es la Factorización de Matrices (MF, Matrix Factorization), en la que la matriz original se descompone en matrices rectangulares tal que su producto sea similar a la matriz original [54]. Originalmente se propuso un método parecido a Single Value Decomposition (SVD) [55], que posteriormente fue mejorado como SVD++ teniendo en cuenta el *bias* o sesgo de cada ítem o usuario para poder abordar también feedback implícito [56]. Recientemente, se han propuesto técnicas de descomposición de matrices mediante arquitecturas no lineales, como Neural Collaborative Filtering (NCF) [57] que utiliza redes neuronales.

3.3.1.2. Modelos basados en grafos

De entre todos los modelos de aprendizaje profundo para recomendación, consideramos los métodos basados en grafos para la tarea de filtrado colaborativo [58]. En este tipo de modelos, se utilizan técnicas de *embedding* de grafos para modelar las relaciones entre los nodos [59]. Los sistemas recomendadores tradicionales, como MF, solo tienen en cuenta los elementos con los que el usuario interactúa directamente. En cambio, las GNN utilizan información de un vecindario extendido. Esto les permite capturar relaciones más complejas y aprovechar mejor la información estructural [60]. Se espera que este tipo de modelos puedan captar los matices de las relaciones presentes entre los miembros de la DAO, como lo han demostrado en tareas similares, como la identificación de usuarios malintencionados [61].

Uno de los sistemas pioneros en este campo es DeepWalk [62], que utiliza paseos aleatorios para aprender la representación de nodos. Poco después surgieron las Graph Convolutional Network (GCN) [63], que combinan la convolución de grafos y las redes neuronales para profundizar en la estructura de los subgrafos con vecinos a varios saltos. GraphSAGE fue uno de los primeros modelos en seguir este enfoque [64], y Neural Graph Collaborative Filtering (NGCF) [65] introdujo los este tipo de redes al campo del filtrado colaborativo, logrando resultados estado del arte. Para este proyecto, se decidió utilizar LightGCN, una popular modificación de NGCF, debido a su simplicidad y buenos resultados en *user-item collaborative filtering*, superando otros modelos basados en GCN previos [66].

LightGCN está basado en el modelo Neural Graph Collaborative Filtering (NGCF) [65], pero se simplifica al eliminar las matrices de transformación de características que se aprenden en cada convolución. En su lugar, la operación de convolución de grafos (también conocido como regla de propagación), denominada Light Graph Convolution (LGC), realiza una suma ponderada normalizada de los *embeddings* de los vecinos, como se detalla en la ecuación 3.1

$$e_u^{(k+1)} = \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}} e_i^{(k)};\ e_i^{(k+1)} = \sum_{i \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_i|}\sqrt{|\mathcal{N}_u|}} e_u^{(k)} \tag{3.1}$$

Por lo tanto, los únicos parámetros entrenables del modelo son los de la primera capa, y la matriz de predicciones final es la media de los embeddings de cada capa, como se detalla en la ecuación 3.2.

$$e_u = \sum_{k=0}^{K} e_u^{(k)};\ e_i = \sum_{k=0}^{K} e_i^{(k)} \tag{3.2}$$

La puntuación para una determinada interacción usuario-ítem será $y_u = e_u^\top e_i$. Dado que los únicos parámetros que son aprendidos son los de la primera capa del modelo, la complejidad de este algoritmo es similar a MF, dependiente del número de dimensiones del espacio latente y el número total de usuarios e ítems. La arquitectura de este modelo queda definida por el número de dimensiones del espacio latente y el número de capas de convolución.

La función de pérdida utilizada por el modelo es Bayesian Personalized Ranking (BPR) [67]. Creado para feedback implícito, en lugar de reemplazar las no-interacciones con ceros asumiendo que es equivalente a feedback negativo, se comparan muestras positivas y muestras negativas para cada usuario, buscando maximizar la probabilidad de que el usuario prefiera la muestra positiva frente a la negativa.

En este trabajo se ha utilizado la implementación de este modelo de la librería Microsoft Recommenders [50], aunque ha sido necesario modificarlo para tener en cuenta las propuestas cerradas añadiendo un post-filtrado, como se detalla en la subsección 6.3.2. El código modificado se encuentra disponible en el GitHub del proyecto [68].

3.3.2. Filtrado basado en contenido

Los sistemas basados en contenido se basan en los atributos del elemento a recomendar y el perfil de preferencias del usuario, que a su vez está basado en los atributos de elementos con los que ha interactuado en el pasado [52].

Este tipo de sistemas tienen una ventaja en el escenario de arranque en frío (*cold-start*) en items, en el que aún no se tiene suficiente feedback de otros usuarios como para realizar buenas recomendaciones con filtrado colaborativo. En tal caso, con tener suficiente información para formar un buen perfil del usuario es suficiente. Sin embargo, estos sistemas no pueden realizar recomendaciones a nuevos usuarios, al no tener información con la que formar un perfil.

Un sistema recomendador basado en contenido generalmente sigue tres pasos: el preprocesado y extracción de atributos de los ítems, el aprendizaje de los perfiles de los usuarios, y finalmente el filtrado y recomendación.

En la extracción de atributos se crea un vector para cada elemento, y es un paso muy dependiente del dominio. Por ejemplo, con información textual, pueden usarse métodos clásicos como *bag-of-words*, TF-IDF, o *embeddings* preentrenados. El aprendizaje de los modelos de usuario es similar al campo de clasificación de texto, y se usan métodos similares como k vecinos más cercanos (kNN, k-nearest neighbors), similitud del coseno, o clasificadores bayesianos. Finalmente, se utilizan los resultados del modelo anterior para realizar la recomendación, por ejemplo recomendando los k ítems con mayor similitud al usuario, o que mayor probabilidad tienen de pertenecer a la clase de ítems interactuados. Este último paso suele ser muy eficiente y puede ejecutarse en tiempo real.

En la sección 6.3.1 se explora la implementación usada en este trabajo, que hace uso de *embeddings* preentrenados para las características, y explora varios métodos de creación del perfil del usuario. El código del modelo desarrollado se puede encontrar en el GitHub del proyecto [68].

3.3.3. Híbridos

Los sistemas híbridos permiten combinar múltiples fuentes de datos para realizar mejores recomendaciones. Según el diseño del sistema, podemos distinguir las siguientes maneras de desarrollar un sistema recomendador híbrido [52]:

1. **Diseño monolítico**: Se crea un sistema recomendador que integra varias fuentes de datos, sin que existan componentes individuales. Un ejemplo de estos sistemas son las Factorization Machine (FM) [69], que combinan las ventajas de los Support Vector Machines (SVMs) con la Factorización de Matrices [70].
2. **Sistemas mixtos** Utilizan múltiples sistemas recomendadores preexistentes y se presentan sus resultados a los usuarios en paralelo.
3. **Diseño de ensamblado** Al igual que en los sistemas mixtos, se utilizan varios sistemas recomendadores preexistentes que son combinados en una salida más robusta. El quid de este tipo de sistemas es elegir una metodología para combinar la salida de estos sistemas, que puede utilizar o bien los *scores* de cada sistema o bien la posición de las recomendaciones realizadas.

El sistema híbrido desarrollado en este trabajo es un sistema de ensamblado que combina los otros dos sistemas desarrollados utilizando la posición de sus recomendaciones, y su especificación se detalla en la sección 6.3.3.

Capítulo 4

Planteamiento del problema

Las DAO permiten a sus miembros la toma de decisiones mediante el voto en propuestas. Si bien cualquier individuo, miembro o no, puede proponer, únicamente los miembros tienen el derecho de votar sobre estas propuestas. No obstante, la abundancia de propuestas y la baja participación plantean un desafío en el funcionamiento de estas organizaciones. En respuesta a esta problemática, se propone la implementación de un sistema recomendador. Este sistema tiene como objetivo sugerir a los usuarios propuestas en las cuales puedan interactuar dentro de una DAO, promoviendo así una mayor participación y compromiso por parte de los miembros.

Es importante resaltar que en algunas DAOs sin muchos miembros, no suelen unirse nuevos usuarios, la mayoría se unen en la creación de la DAO, por lo que el problema del *cold-start* en los miembros no debería ser demasiado notable. Sin embargo, cada día se agregan varias propuestas que, además, tienen un tiempo de vida establecido, por lo que el sistema tendrá que lidiar continuamente con el problema de *cold-start* con cada propuesta creada, y además no es conveniente realizar recomendaciones de propuestas que ya están cerradas, pues el usuario no podrá votar.

El usuario puede expresar su preferencia votando a favor o en contra de una propuesta, y ambas acciones se interpretan como una muestra positiva para el sistema recomendador. Es decir, se trata de feedback implícito [71]. Es relevante señalar que, dentro de una DAOs, el estado actual de la votación es transparente y no anónimo, lo que puede influir en el comportamiento de los usuarios y su decisión de participar en el proceso de votación. Si la propuesta ya tiene mayoría absoluta (o incluso una fuerte mayoría relativa), puede que el usuario no vote.

El sistema recomendador propuesto para DAOs sería similar a un sistema utilizado en otros contextos temporales, como eventos u obras de teatro, en el que no tiene sentido recomendar una obra que ya no está en cartelera, pues el usuario no podrá asistir a ella.

4.1. Definición formal del problema

El objetivo es realizar k recomendaciones de propuestas en las que votar a cada uno de los miembros de la organización en un momento determinado, teniendo en cuenta la imposibilidad de votar en propuestas que ya están cerradas o aún no se han abierto.

En términos de un sistema recomendador, los usuarios serían los votantes, los ítems serían las propuestas, y la interacción entre ambos sería cada uno de los votos. La interacción es feedback implícito, pues no contamos con una valoración explícita de las propuestas, y la no interacción no implica la aversión del usuario hacia la propuesta.

Además de la marca temporal, los ítems y las interacciones están anotadas con otros datos como la descripción de la propuesta, si el usuario votó a favor o en contra, qué usuario creó la propuesta, a qué otras organizaciones pertenece el usuario, o el poder de votación de los usuarios.

4.1.1. Modelo de la organización como un grafo

Si reducimos una DAO a las interacciones producidas entre los usuarios y las propuestas, podemos modelarla como un grafo bipartito $G = (M, P, E)$, donde los vértices representan a los miembros (M) y a las propuestas (P), y las aristas (E) denotan los votos en dichas propuestas. Este grafo es dinámico en el tiempo, ya que las propuestas tienen un período de apertura y cierre, y las interacciones entre usuarios y propuestas ocurren en momentos específicos. Aunque una propuesta se cierre, no desaparece del grafo, simplemente es una etiqueta que indica que no tiene sentido recomendarla.

En el contexto de grafos dinámicos, existen dos categorías [72]: los Grafos dinámicos de tiempo discreto (DTDG, Discrete-time dynamic graphs), donde cada elemento está etiquetado con un conjunto de instantes en los que existe, y los Continuous-time dynamic graphs (CTDG), donde cada elemento está marcado con el momento a partir del cual existe. En el caso de las votaciones en una DAO, se pueden definir como un CTDG $D(t) = (M, \mathcal{P}(t), \mathcal{E}(t))$, donde M representa los miembros, $\mathcal{P}(t)$ son las propuestas abiertas en el tiempo t, y $\mathcal{E}(t)$ son los votos realizados hasta ese instante.

Las propuestas (P) se caracterizan por un intervalo de tiempo (t_i, t_f) que indica su apertura y cierre, y $\mathcal{P}(t)$ consiste en las propuestas cuyo período de apertura inicia después del tiempo t, es decir, $t_i > t$. Los votos (E) se definen como una terna (m, p, t_v), donde m es el miembro que vota, p es la propuesta votada, y t_v es el momento de la votación. De este modo, $\mathcal{E}(t)$ comprende los votos realizados hasta el tiempo t, es decir, $t_v > t$. Además, todos los votos ocurren dentro del período de apertura de la propuesta, por lo que se debe cumplir la propiedad $t_i < t_v < t_f$ para todos los votos que interactúen con una propuesta.

Aunque los miembros también son variables en el tiempo, no se ha considerado en el modelo debido a que el conjunto de miembros de una organización tiende a ser bastante estático.

Además del tiempo, tanto los vértices como las aristas están anotados con otros datos, como la información textual de las propuestas, el usuario que la ha creado, si ha votado a favor o en contra, o el poder de votación de dicho usuario.

Modelar la organización como un grafo nos permite reducir el problema de recomendación a un problema de predicción de enlace, aumentando el número de modelos posibles a utilizar, y permitiendo extender el grafo en el futuro con otras relaciones.

4.2. Exploración de datos

Para el desarrollo del sistema, se ha utilizado una variante del DAO Analyzer Dataset [73], una recopilación de datos que abarca más de 30 000 despliegues en 7 plataformas de DAOs. La sección 6.1 contiene más información sobre la fuente de los datos utilizados en este trabajo. En este conjunto de datos, se registraron más de 5 millones de usuarios que emitieron más de 22 millones de votos en aproximadamente 180 mil propuestas. Se emplea el término *despliegue* en lugar de *DAO*, dado que se concibe a una DAO como una comunidad que puede haber sido desplegada en diversas plataformas.

Además de la información del grafo de votantes, votos y propuestas, se complementó el conjunto de datos con datos textuales de las propuestas de algunas plataformas importantes, como Aragon, DAOstack, DAOhaus y Snapshot. Esta información textual enriquece el análisis al permitir la creación de un sistema recomendador basado en contenido. En la tabla 4.1 se muestra un resumen de cada plataforma que forma el conjunto de datos. Aunque se ha obtenido información textual de todas las plataformas, seguimos contando con más de 25 000 despliegues, y el 70 % de las propuestas tienen información textual, ya sea un título o una descripción.

Para el análisis de las organizaciones, se excluyen aquellas consideradas "triviales", es decir, aquellas con 3 miembros o menos, o sin votos. Además, se identifican como una misma organización los despliegues con el mismo nombre, o que hemos identificado en otros análisis que están formados por la misma comunidad, como dxDAO/xDXdao o Aave/Aavegotchi. Tras este proceso, se obtienen 4 600 organizaciones. El 50 % de ellas cuentan con menos de 31 votantes, mientras que para superar los 100 votantes se debe alcanzar el percentil 72, y el 95 % para los 1 000 votantes. Solo el 99.3 % alcanza los 10 000 votantes, con solo 3 organizaciones superando los 100 000 votantes.

Aunque las organizaciones con pocos miembros pueden funcionar, la mayoría posee menos de 8 propuestas, con solo el 4 % superando las 100 propuestas realizadas. La distribución de votos sigue un patrón similar, con un 55 % de las organizaciones teniendo menos de 100 votos emitidos. Un 12 % cuenta con más de 1 000 votos, y menos del 3 % con más de 10 000.

Considerando lo anterior, el conjunto de organizaciones adecuado para entrenar y probar el sistema recomendador se ha limitado a 12 entidades con más de 100 votantes y más de 500 propuestas. Estas organizaciones se detallan en la tabla 4.2, que incluye sus características y diversos aspectos del grafo, como los Votos por Pro-

Plataforma	# deployments	# votos	# votantes	# propuestas	% texto
aragon	2 387	27 581	4 358	16 598	9 %
daohaus	3 528	52 033	5 816	48 866	36 %
daostack	58	12 331	519	3 575	99 %
snapshot	19 615	20 951 104	4 849 174	106 504	93 %
governor	885	413 578	84 299	885	–
realms	2 287	33 214	6 934	2 287	–
tally	2 375	556 988	187 976	2 375	–

Tabla 4.1: Número de despliegues, votos, votantes y propuestas en cada plataforma

Nombre	# Prop.	# Usu.	# Votos	‰ Dens.	VPP	VPV
DEAD FoundationsDAO[†]	5 591	3 469	17 738	1.83	3.17	5.11
PancakeSwap	2 691	129 978	532 831	3.05	198.00	4.10
dxDAO / xDXdao[†§]	2 226	193	8 479	39.47	3.81	43.93
Decentraland	2 060	7 334	116 880	15.47	56.74	15.94
Aave / Aavegotchi	1 140	87 593	2 360 660	47.28	2070.75	26.95
MetaCartel / MC Ventures	934	199	3 288	35.38	3.52	16.52
Index Coop[§]	874	2 871	24 032	19.15	27.50	8.37
gm DAO[*]	710	7 712	91 548	33.44	128.94	11.87
9K DAO[*]	590	8 170	102 321	42.45	173.43	12.52
WEALTHDAO[*]	585	1 041	4 008	13.16	6.85	3.85
HUWA-DAO[§]	572	1 331	4 151	10.90	7.26	3.12
Balancer[§]	509	9 107	111 988	48.32	220.02	12.30

[†] Organización extinta: sin votos desde hace más de un año o con una última propuesta que disuelve la organización
[*] Marcada como SPAM en la plataforma
[§] Insuficiente número de folds seguidos con los que probar el recomendador (véase capítulo 7)

Tabla 4.2: Organizaciones seleccionadas para su posible análisis. La densidad del grafo se presenta en milésimas.

puesta (VPP) que indican el grado medio de los nodos de tipo *propuesta*, y los Votos por Votante (VPV) que indican el grado medio de los nodos de tipo *votante*, así como la densidad del grafo bipartito: $\frac{2|V|}{|M||P|}$.

Finalmente, se ha optado por utilizar los datos de la comunidad Decentraland, dado que las dos organizaciones con más propuestas en la tabla presentan una densidad insuficiente y dxDAO ya no está en uso y carece de una cantidad significativa de usuarios y votos. Por ende, en el transcurso del trabajo, se centrará en Decentraland, aunque se examinará la ejecución del sistema en otras organizaciones mencionadas en los resultados (capítulo 7).

4.2.1. Exploración de la organización Decentraland

Decentraland es una plataforma de metaverso donde los usuarios pueden adquirir parcelas virtuales como Tokens no fungibles (NFTs, Non-Fungible Tokens) utilizando la criptomoneda MANA basada en Ethereum, con un volumen de 168 millones de dólares estadounidenses según DappRadar [74]. Su desarrollo comenzó en 2015 y en 2017 consiguió 26 millones de dólares en su Oferta inicial de criptomonedas (ICO, Initial Coin Offering) en la que fue lanzada al público [75], y en 2022 se valoró la plataforma con un valor de 1.2 mil millones de dólares [76]. Ha atraído la atención de grandes marcas como Samsung, Tommy Hilfiger o PricewaterhouseCoopers [77], quienes han adquirido «propiedades» en su mundo virtual, y por lo tanto son miembros con pleno derecho a voto.

Su proceso de votación está desplegado en Snapshot, donde se han presentado más de 2 000 propuestas y participan más de 7 000 votantes, emitiendo más de 115 000 votos en total. Sin embargo, como en la mayoría de las DAOs, la participación es baja, con la mayoría de los miembros votando como mucho en 3 propuestas. Aunque existen 197 miembros ($\sim 2.71\,\%$) que han emitido más de 100 votos cada uno, la

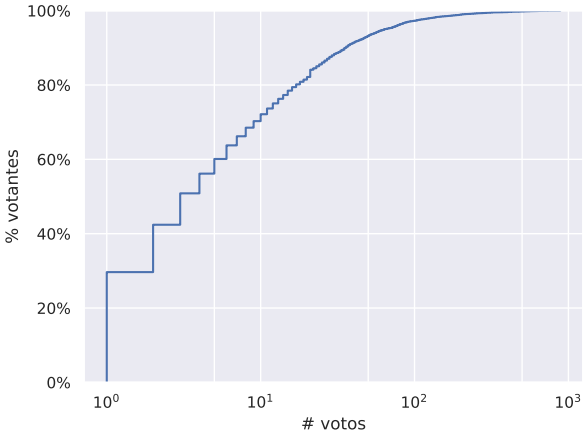


Figura 4.1: Función de Distribución Cumulativa de votos por usuario de Decentraland.

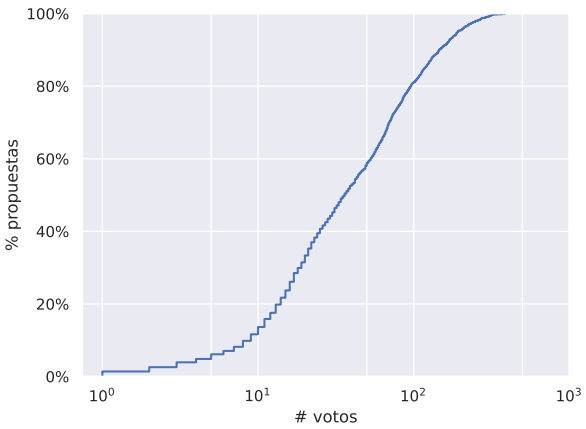


Figura 4.2: Función de Distribución Cumulativa de votos por propuesta de Decentraland.

distribución de votos en propuestas de la figura 4.1 muestra una gran cantidad de usuarios con menos de 10 votos y una pequeña minoría con más de 100.

En cuanto a las propuestas, el grafo es más denso, con una media de 56 votos por propuesta y más de 350 propuestas con más de 100 votos. Sin embargo, la propuesta más votada cuenta con votos de solo el 5 % del total de miembros (385 votos). La figura 4.2 muestra de forma detallada la distribución de votos por propuesta en la organización.

Como puede observarse en la figura 4.3, el número de usuarios activos a lo largo del tiempo ha tenido picos de hasta 1 400, pero parece situarse entre los 600 y 800, con una pequeña bajada durante el verano. Además, como se observa en la figura 4.5, se crean entre 10 y 20 propuestas semanalmente, aunque en ocasiones se han observado picos de hasta 70 propuestas, lo que puede dificultar la participación.

La distribución de las propuestas creadas a lo largo de la semana es monotónica y descendente como puede verse en la figura 4.4, con casi el doble de propuestas creadas un lunes que en un día de fin de semana.

En las distribuciones marginales de la figura 4.6, se pueden observar patrones interesantes en la distribución de los votos y la creación de propuestas en Decentraland a lo largo de los días de la semana. Sobre las votaciones, los usuarios muestran una tendencia uniforme a lo largo de la semana, con una ligera bajada de participación los sábados y domingos, y un ligero repunte los martes, prefiriendo los usuarios la actividad en los días hábiles.

En la sección central de la figura, se presenta un mapa de calor que ilustra la cantidad de propuestas y votos emitidos cada día de la semana. Se observa que las propuestas reciben la mayor cantidad de votos al día siguiente de su creación, y, a medida que transcurren los días, experimentan una disminución en la cantidad de votos recibidos. Esta regla sólo se rompe los lunes, que puede haber más votos que en Domingo para las propuestas creadas ese día, ya que la participación es menor como se indica en el párrafo anterior.

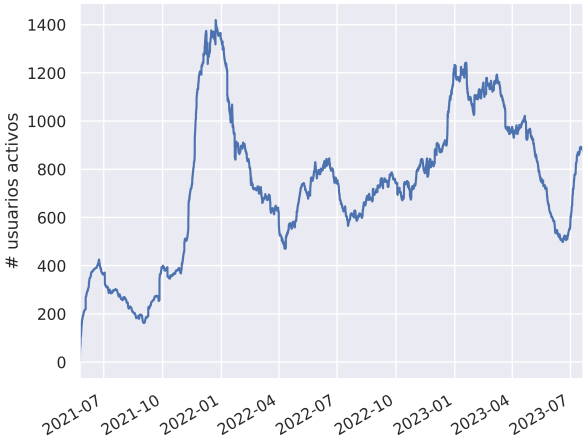


Figura 4.3: Número de usuarios activos en Decentraland a lo largo del tiempo. Un usuario se considera activo si ha realizado un voto en una propuesta en los últimos 30 días.

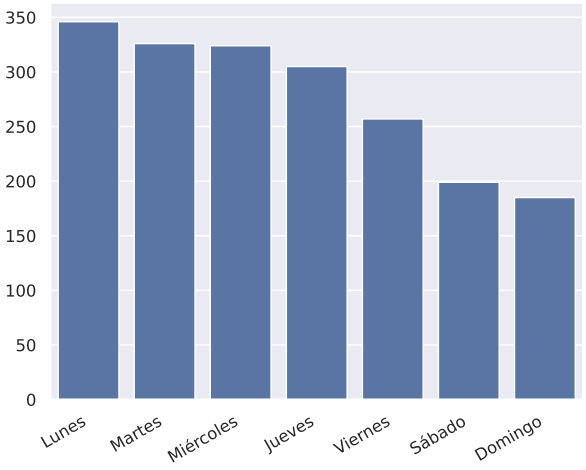


Figura 4.4: Número de propuestas creadas por día de la semana en Decentraland.

4.3. Propuesta de sistema

En esta sección se realiza una propuesta inicial del posible Sistema Recomendador (SR). Se buscará utilizar modelos de SRs ya validados y disponibles en librerías estándar. Se construirán tres modelos: uno siguiendo un enfoque basado en contenido, otro siguiendo un enfoque basado en Filtrado Colaborativo (CF, Collaborative Filtering), y un tercero que será un híbrido que combine los otros dos sistemas.

El modelo basado en contenido se limitará a analizar únicamente el título y la descripción de las propuestas, empleando técnicas de Procesamiento del Lenguaje Natural (PLN). Por otro lado, el modelo basado en filtrado colaborativo se basará en analizar el grafo bipartito formado por los votantes y las propuestas, utilizando una GNN. El sistema híbrido integrará los *rankings* producidos por los dos otros sistemas.

En ninguno de los dos sistemas propuestos se considerará más contexto ni información adicional como el tiempo o día de creación de la propuesta. No obstante, se aplicará un pos-filtrado para evitar recomendar aquellas propuestas cuyo periodo de votación haya concluido. Para ello, el método encargado de realizar las k recomendaciones utilizando el modelo recibirá el conjunto de propuestas consideradas *recomendables*, y se encargará de clasificarlas según su idoneidad, y devolviendo las k mejores recomendaciones para cada usuario.

De esta manera, el modelo se ejecutará en intervalos discretos de tiempo, y la evaluación será en consecuencia. Es decir, asumiendo que el recomendador se ejecutará de forma periódica (por ejemplo, cada semana o cada día) y se evaluará de acuerdo con esta periodicidad. Por ejemplo, para poder enviar un boletín o *newsletter* personalizado con dicha periodicidad y mejorar la participación [78]. No se contempla la ejecución continua, sino que presupone la posibilidad de reentrenar completamente el modelo en cada ocasión que se desee realizar nuevas recomendaciones. Por consiguiente, uno de los requisitos es que el entrenamiento de los sistemas desarrollados sea rápido, con tiempos de entrenamiento en el orden de minutos.

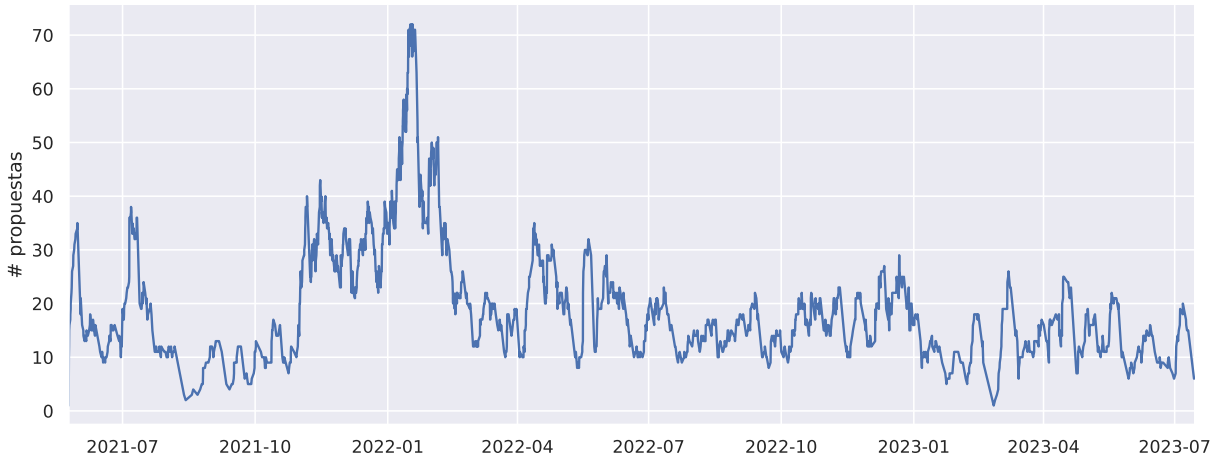


Figura 4.5: Suma móvil de propuestas abiertas en los últimos 7 días en la DAO Decentraland.

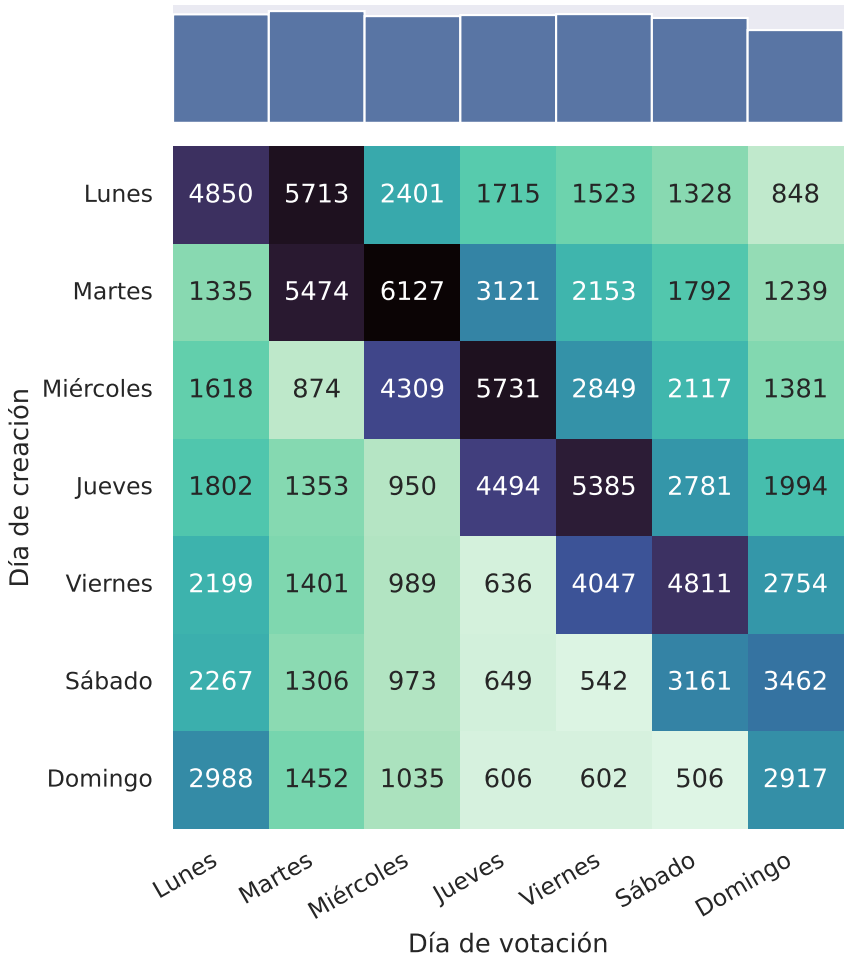


Figura 4.6: Mapa de calor con distribución marginal ilustrando la correlación entre el día en el que se produce un voto y el día de creación de la propuesta en la que se realizó el voto.

Capítulo 5

Entrenamiento y validación de los sistemas

En el ámbito de los sistemas recomendadores, existen dos paradigmas principales de evaluación [52]. Por un lado, la *evaluación online* utiliza la interacción directa de los usuarios con el sistema recomendador ya desplegado, permitiendo evaluar cómo responden los usuarios a las recomendaciones presentadas en tiempo real. Por otro lado, la *evaluación offline* se basa en el uso de datos históricos, donde se entrena un modelo utilizando las interacciones previas de los usuarios, junto con cualquier información contextual disponible.

La ventaja de utilizar un dataset offline radica en la posibilidad de comparar varios sistemas recomendadores bajo condiciones controladas, aunque también sería posible utilizar A/B testing en una evaluación online para probar distintos modelos, asignando modelos distintos a cada usuario. Sin embargo, es importante tener en cuenta que el enfoque offline no proporciona una visión completa de cómo se comportarán los sistemas en la realidad ni si serán capaces de adaptarse a los cambios en el conjunto de datos a lo largo del tiempo. En nuestro caso, debido a la disponibilidad de datos históricos, utilizaremos una evaluación offline para entrenar y validar nuestros sistemas recomendadores, pero añadiendo la variable temporal para hacer más robusta la validación de los modelos y evitar el *data leakage*, un riesgo presente en la evaluación offline de sistemas recomendadores [79].

La realización de una evaluación online requeriría una integración más estrecha del sistema recomendador desarrollado en la plataforma, lo que podría no ser factible utilizando únicamente datos de la blockchain o las APIs públicas disponibles. Sin embargo, al ser software libre, podría realizarse un fork de alguna de las plataformas modificando su interfaz para añadir el sistema recomendador, o crear una extensión de navegador que efectúe las recomendaciones.

Además, debido a las características específicas de estos datos, como se detalla en el capítulo 4, se ha tenido que desarrollar técnicas y enfoques adaptados a nuestro contexto, que tengan en cuenta que las propuestas están abiertas durante un tiempo limitado. El objetivo es emplear métricas clásicas de sistemas recomendadores, como precisión, nDCG o MAP para que sean más fácilmente interpretables, pero teniendo en cuenta la naturaleza temporal de nuestros datos.

5.1. División del conjunto de datos

La evaluación de sistemas recomendadores con elementos limitados en el tiempo presenta un desafío, ya que no existe una metodología estándar ampliamente aceptada para su evaluación.

La estrategia seguida para abordar este problema es dividir el conjunto de datos en dos partes distintas: una para el entrenamiento del modelo y otra para su evaluación. Sin embargo, dado que se busca validar el modelo con múltiples subconjuntos de datos, se implementará una variante de la validación cruzada temporal. En este enfoque, se utilizarán las marcas de tiempo asociadas a los votos para realizar una partición temporal del conjunto de datos.

Inicialmente, consideramos utilizar la clase `TimeSeriesSplit` de Scikit-learn [80] para la división del conjunto de datos. Esta clase ordena los datos por su marca temporal, y divide el conjunto en K intervalos, cada uno con un número igual de muestras. Además, las particiones son incrementales, de tal manera que en la partición k-ésima se retorna el split $k+1$ como conjunto de test, pero todos los splits de 0 a k para entrenamiento, siendo cada uno un superconjunto del anterior.

Sin embargo, en nuestro caso, la distribución de actividad no es uniforme en el tiempo, como se ilustra en la figura 4.5. Por lo tanto, optamos por una estrategia alternativa, en la que el número de elementos en cada fold no es necesariamente el mismo, pero el intervalo de tiempo entre particiones sí lo es, y manteniendo que cada conjunto de entrenamiento sea un superconjunto del anterior. De hecho, podemos identificar cada fold por la marca de tiempo que divide los dos conjuntos. Los votos realizados antes de ese momento t pertenecerán a entrenamiento y los realizados posteriormente, a la evaluación, lo que en el campo de *forecasting* se conoce como *out-of-sample testing* [81].

Además, es importante destacar que en el conjunto de entrenamiento, la mayoría de las propuestas ya han sido cerradas y, por lo tanto, no pueden ser objeto de recomendación pues los usuarios no podrían votar en ellas, aunque son útiles para

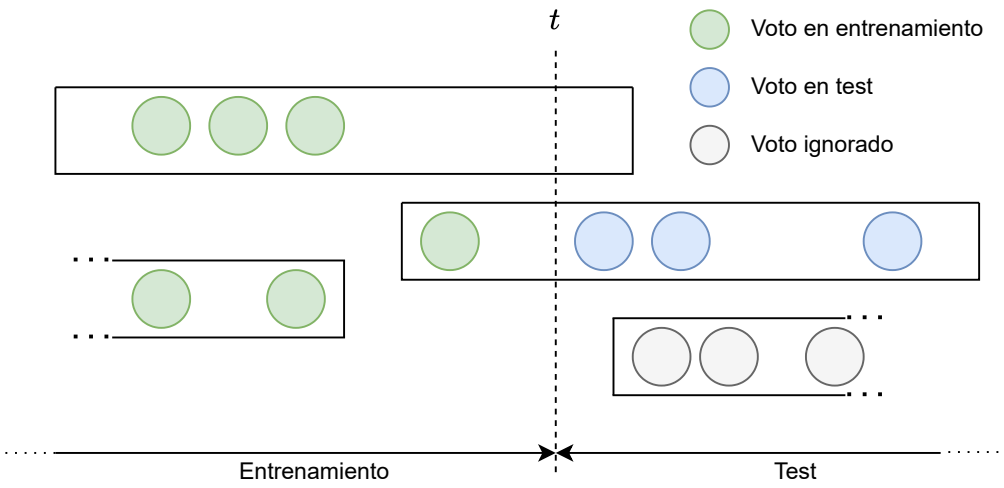


Figura 5.1: Esquema ejemplificando la creación de particiones de entrenamiento y test.

crear el modelo del usuario.

De manera similar, al considerar el conjunto de prueba, nos encontramos con que muchas propuestas aún no han sido creadas y, como resultado, no habrán podido ser votadas por los usuarios durante el conjunto de entrenamiento y una predicción realizada sobre esas propuestas usando filtrado colaborativo carecerá de validez. El recomendador basado en contenido sí que podrá realizar recomendaciones satisfactorias, mientras que el basado en filtrado colaborativo les asignará una prioridad mínima y probablemente no sean recomendadas.

Por esa razón, las métricas se calculan exclusivamente utilizando las interacciones generadas en el conjunto de prueba en propuestas que ya estaban creadas durante el intervalo de entrenamiento. Es decir, el conjunto de prueba se filtra para conservar únicamente las interacciones en propuestas abiertas. La figura 5.1 ejemplifica este procedimiento.

Tras desarrollar este sistema, se descubrió que Macedo, Marinho y Santos [41] ya habían implementado un método similar para un sistema de recomendación de eventos, aunque con la distinción de agregar un límite inferior al conjunto de entrenamiento.

Se ha implementado un código en Python para llevar a cabo este proceso, utilizando la tabla que contiene los votos y la que contiene las marcas de tiempo de las propuestas. Este código se encuentra en la función `timeFreqSplitCurrent`, ubicada en el módulo `src.model_selection`. Este módulo está disponible en el repositorio del proyecto en GitHub [68]. Además, se incluye una versión simplificada del código en esta memoria, que se puede encontrar en Código 5.1.

```
def timeFreqSplit(dfv: pd.DataFrame, freq: str, dfp: pd.DataFrame, normalize=True):
  times = pd.date_range(dfv['timestamp'].min(), dfv['timestamp'].max(),
                        freq=freq, normalize=normalize)
  for train_end in times[:-1]:
    train = df[df[timestamp_col] <= train_end]
    test = df[ (train_end < df['timestamp'])]

    open_proposals = dfp[ (dfp['start'] < train_end) & (train_end < dfp['end']) ]['id']
    test_filtered = test[test['itemID'].isin(open_proposals)]

    yield train, test_filtered
```

Código 5.1: Simplificación del método `timeFreqSplit` del módulo `src.model_selection`

Finalmente, para acomodar los datos al modelo LightGCN de Microsoft que necesita que los usuarios en test hayan estado previamente presentes en entrenamiento, se ignoran los votos de los usuarios nuevos: aquellos que aparecen tan sólo en el conjunto de test.

5.1.1. Exploración de la división en folds de los datos de Decentraland

Para desarrollar un sistema recomendador con los datos de la organización Decentraland, se ha optado por dividir los datos en folds de 7 días cada uno, pues es el período típico de duración de las propuestas dentro de la plataforma. Con los datos recopilados desde mayo 2021 hasta julio de 2023, se generan un total de 112 folds

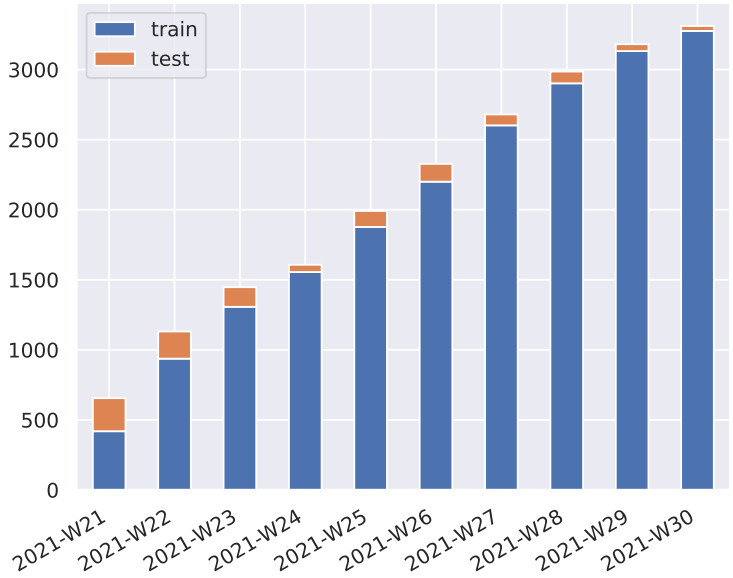


Figura 5.2: Cantidad de votos en entrenamiento y prueba de los primeros 10 folds de la organización Decentraland.

que podrían utilizarse para la evaluación del sistema. El conjunto de entrenamiento en estos folds es incremental, como se muestra en la figura 5.2.

Sin embargo, por restricciones computacionales, se ha limitado el análisis a los 10 últimos folds, que abarcan desde mayo hasta julio de 2023. Estos folds contienen en entrenamiento casi todos los votos disponibles, yendo desde 106 000 a 115 000 votos. Por otro lado, el conjunto de test contiene una media de 439 votos de 142 usuarios realizados en entre 13 y 25 propuestas.

En un sistema de filtrado colaborativo, es fundamental que las propuestas a recomendar ya cuenten con cierta cantidad de votos para poder evaluarlas. Estas propuestas están abiertas en el momento de realizar el split, y aparecerán también en la evaluación. En estos 10 folds, hay una media de 1283 votos por 325 usuarios en dichas propuestas.

En la tabla 5.1 se muestra más en profundidad los datos de cada uno de los 10 folds, pero hay que destacar que los votos por cada votante en validación son en general muy bajos, a penas superando los 3 VPV en dos de los folds.

5.2. Métricas utilizadas

En esta sección se presentan métricas offline comúnmente empleadas en el ámbito de la recuperación de información y sistemas recomendadores. Se han utilizado las métricas disponibles en la librería Microsoft Recommenders [50].

5.2.1. Precisión y exhaustividad

Las métricas de precisión y exhaustividad son ampliamente utilizadas en la evaluación de sistemas recomendadores, adoptadas desde el campo de la recuperación de información. La precisión se define como la probabilidad de que un ítem recuperado

Semana	Prop. abiertas	Solo entrenamiento				Solo validación			
		Votos	Usuarios	vpp	vpu	Votos	Usuarios	vpp	vpv
2023-W19	18	1627	358	90.39	4.54	354	139	19.67	2.55
2023-W20	25	1346	305	53.84	4.41	811	169	32.44	4.80
2023-W21	19	1483	305	78.05	4.86	332	122	17.47	2.72
2023-W22	13	819	247	63.00	3.32	289	101	22.23	2.86
2023-W23	13	631	191	48.54	3.30	341	118	26.23	2.89
2023-W24	16	872	225	54.50	3.88	391	132	24.44	2.96
2023-W25	17	1136	278	66.82	4.09	360	148	21.18	2.43
2023-W26	10	838	278	83.80	3.01	239	107	23.90	2.23
2023-W27	21	1591	469	75.76	3.39	890	249	42.38	3.57
2023-W28	23	2493	600	108.39	4.16	384	142	16.70	2.70

Tabla 5.1: Número de propuestas abiertas en cada uno de los folds durante las últimas 10 semanas en Decentraland. Bajo *Solo entrenamiento* se muestra el número de votos emitidos en esas propuestas que se han usado dentro del conjunto de entrenamiento, y los usuarios que han emitido esos votos. Un usuario puede haber votado en varias propuestas. A la derecha, bajo *Solo validación* se muestra el número de votos emitidos en las propuestas durante la fase de validación. *vpp* es el ratio de Votos Por Propuesta, y *vpv* es el ratio de votos por Votante.

sea relevante, expresada como el cociente entre el número de ítems relevantes recuperados y el número total de ítems recuperados. Por otro lado, la exhaustividad, también conocida como recall, indica la probabilidad de que un ítem seleccionado sea relevante, calculada como el cociente entre el número de ítems relevantes recuperados y el número total de ítems relevantes [82].

En el contexto de un sistema recomendador que clasifica los ítems para su recomendación, estas métricas pueden ser adaptadas para evaluar únicamente los k primeros ítems recomendados, conocidas como *precisión en k* ($precision@k$) y *exhaustividad en k* ($recall@k$). La fórmula de estas métricas se define en las siguientes ecuaciones:

$$precision@k = \frac{|\text{items rel. rec. en } k|}{k} \quad (5.1) \qquad recall@k = \frac{|\text{items rel. rec. en } k|}{|\text{items rel. en } k|} \quad (5.2)$$

Es importante tener en cuenta que el valor de estas métricas es altamente sensible al valor de k, por lo que no se pueden comparar métricas con distintos valores de k. Además, otro importante defecto de estas métricas ocurre cuando el número de elementos relevantes es menor que k, ya que incluso un sistema perfecto tendría un valor menor que 1 en estas métricas [83].

Por esa razón, se consideró usar la métrica $r - precision$, la cual tiene en cuenta el número de documentos relevantes (R) para una consulta y calcula la precision en R ($precision@R$), adaptando el valor de k al número de documentos relevantes [83]. Como $k = R =$ items rel. en k, esta métrica es equivalente al $recall@R$.

Sin embargo, a pesar de su potencial utilidad, no se empleó esta métrica debido a su ausencia en la librería Microsoft Recommenders [50]. Su implementación requeriría tiempo adicional y podría afectar significativamente al tiempo de ejecución de la evaluación del modelo, y sus resultados están correlacionados con el MAP, métrica que se explicará en la siguiente sección.

Finalmente, al evaluar el sistema recomendador al completo, para reportar una mé-

trica se hace la media aritmética de el valor de dicha métrica para todos los usuarios a los que se les ha realizado una recomendación.

5.2.2. Métricas de ranking

Las métricas presentadas en la subsección anterior no tienen en cuenta el orden de las recomendaciones realizadas. Sin embargo, los modelos desarrollados sí que ordenan las recomendaciones, de manera que la primera recomendación para un usuario tiene un score superior o igual al del segundo (o pérdida menor).

Además, la lista de ítems recomendables para algunos usuarios es demasiado pequeña, por lo que conviene usar métricas que pongan más atención en las primeras o mejores recomendaciones, pero con una métrica agnóstica al modelo.

Se utilizarán las métricas Ganacia Acumulativa Descontada Normalizada (nDCG, Normalized Discounted Cumulative Gain) y Precisión Media Promedio (MAP, Mean Average Precission), de uso extendido en SRs e implementadas en la librería Microsoft Recommenders [50].

La Ganacia Acumulativa Descontada (DCG, Discounted Cumulative Gain) de un usuario asigna un factor de descuento $log_2(i+1)$ al *ranking* de cada ítem recomendado para ese usuario, y luego suma estos valores descontados para obtener una medida de la utilidad acumulada de las recomendaciones [52]. Por lo general, en lugar de considerar todos los ítems, se calcula hasta un valor k específico, similar a las métricas discutidas en la sección anterior. Así, la fórmula para el Ganacia Acumulativa Descontada (DCG, Discounted Cumulative Gain) de un usuario se presenta en la ecuación 5.3. La Ganacia Acumulativa Descontada Normalizada (nDCG, Normalized Discounted Cumulative Gain) de cada usuario se calcula normalizando el DCG obtenido con respecto al DCG ideal (IDCG), y la fórmula está disponible en la ecuación 5.4.

$$\text{DCG@}k_u = \sum_{i=1}^{k} \frac{rel_i}{\log_2(i+1)} \qquad (5.3) \qquad\qquad \text{nDCG@}k_u = \frac{\text{DCG@}k_u}{\text{IDCG@}k_u} \qquad (5.4)$$

Una vez normalizada, el rango del valor de esta métrica es de 0 a 1, lo que facilita su interpretación y permite comparar distintos modelos para un mismo conjunto de datos. Para calcular el nDCG de un modelo dado, se obtiene la media de los nDCG de cada usuario.

La relación entre precisión y exhaustividad (*recall*) se puede representar mediante una curva de *precision-recall*. La precisión media de un usuario (AveP) se define como el valor medio de la precisión en esta curva. Y puede calcularse utilizando la siguiente suma finita [83]:

$$\text{AveP@}k = \frac{1}{k} \sum_{i=1}^{k} P(k) \cdot \text{rel}(k) \qquad (5.5)$$

Aquí, rel indica la relevancia de la recomendación, que es 1 si estaba en el conjunto de prueba y 0 en caso contrario, tratándose de feedback implícito. La Precisión Media Promedio (MAP, Mean Average Precission) se obtiene realizando la media del AveP@k de todos los usuarios, y también tiene un valor entre 0 y 1.

5.3. Línea base

Para desarrollar un sistema recomendador es esencial establecer una línea base para comparar entre modelos desarrollados. Esta línea base suele ser un algoritmo sencillo y sin personalización ni aprendizaje que proporciona un punto de referencia para comparar los resultados de los modelos. En este sistema recomendador, debido a la peculiar división en folds y los pocos elementos en test, los resultados de la evaluación son muy variables y dificultan la comparación entre los folds, siendo aún más necesaria esta línea base.

Una posible línea base pueden ser los resultados del estado del arte actual, si se tratase de un conjunto de datos extendido. En cualquier caso, la linea base debería ser un modelo establecido, pero debido a que no se han creado previamente sistemas recomendadores con estas características, es necesario introducir un nuevo modelo.

Una de las lineas base más simples, utilizada en el campo de la recuperación de información, consiste en realizar recomendaciones aleatorias entre los elementos recomendables, es decir, entre las propuestas abiertas, aunque esta linea base no proporciona un ranking y todas las recomendaciones tienen el mismo valor.

Sin embargo, la línea base más extendida y ampliamente utilizada en SRs es el modelo *Más Populares* o *MostPop*. Tampoco incorpora ningún nivel de personalización y simplemente recomienda los elementos con más interacciones registradas en el conjunto de datos. Esta linea base, además, destaca por su robustez, superando en ocasiones al filtrado colaborativo cuando los datos están altamente dispersos.

A pesar de su extensa popularidad, la implementación habitual de este modelo no siempre refleja fielmente la realidad [84]. Por ejemplo, no se considera la disponibilidad de los elementos en el momento de la recomendación, ni se tienen en cuenta la popularidad *actual* de un ítem, recomendando en ocasiones elementos con los que un usuario no podría interactuar porque aún no están disponibles [85].

5.3.1. El modelo OpenPop

El modelo desarrollado se denomina OpenPop y recomienda la propuesta abierta con más votos en un momento dado, siempre que el usuario aún no haya emitido su voto sobre ella. Esta estrategia se inspira en el modelo MostPop, y es similar al modelo RecentPop propuesto por Ji et al. [85], con la diferencia clave de que las propuestas, a diferencia de otros ítems, dejan de estar disponibles.

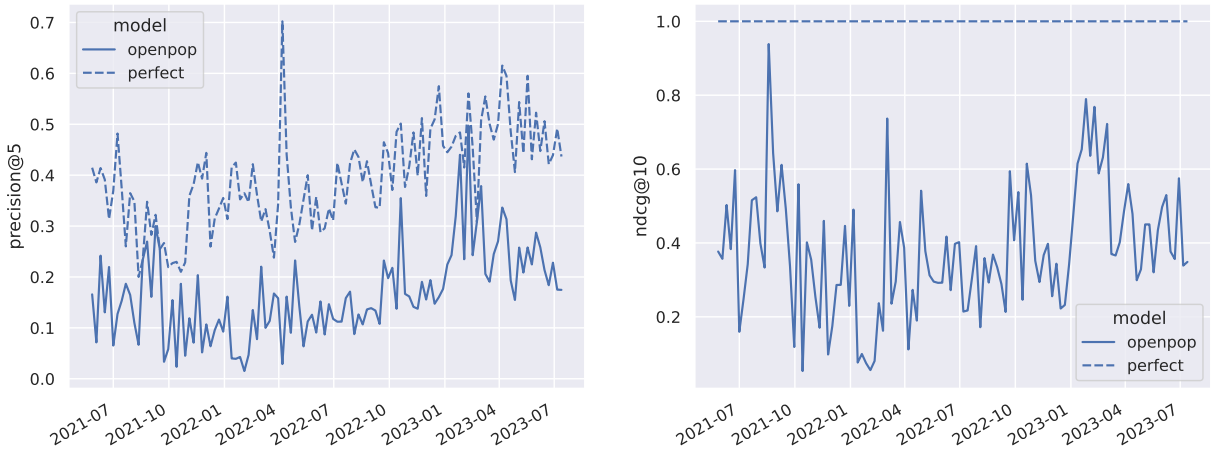
Desde el punto de vista de la implementación, dado un conjunto de datos de entrenamiento y prueba separados como se describe en la Sección 5.1, el modelo simplemente cuenta el número de votos en el conjunto de entrenamiento para las propuestas abiertas, y elimina las recomendaciones que ya están presentes en el conjunto de entrenamiento. El código que implementa dicho modelo está en Código 5.2.

5.3.2. Resultados de la línea base en Decentraland

Otra línea base utilizada, únicamente para compararlo con esta línea base, es el clasificador perfecto. Es necesario pues para algunas métricas utilizadas, al haber muy pocos votos disponibles en test por usuario, es imposible llegar a un valor de 1,

```python
import pandas as pd
from recommenders.datasets.pandas_df_utils import filter_by

def getBaselineRecommendations(train: pd.DataFrame, users, open_proposals, k: int = 5):
    # Number of votes in each proposal in train
    bestVotes = train[train['itemID'].isin(open_proposals)]['itemID'].value_counts()

    # Create pairs (userID, itemID) aka Microsoft's format
    df = pd.DataFrame(it.product(users, bestVotes.index), columns=['userID', 'itemID'])

    # Avoid recommending already voted proposals (they wont be in the test set)
    df = filter_by(df, train, ['userID', 'itemID'])

    # Get just top k recommendations (value_counts sorts by default)
    df = df.groupby('userID').head(k).reset_index(drop=True)

    df['prediction'] = True
    return df
```

Código 5.2: Código del modelo OpenPop.



(a) Precisión en 5

(b) nDCG en 10

Figura 5.3: Resultados del modelo OpenPop (línea base) a lo largo de todos los folds disponibles.

como se explica en la sección 5.2. Estos dos modelos y sus resultados se exploran en el *notebook* `10_baseline_mp.ipynb` del repositorio del proyecto [68].

El análisis de los resultados del modelo OpenPop, utilizando la división de datos de la subsección 5.1.1, revela una $precision@5$ media de 0.17 ± 0.09 en todos los folds. En comparación, un clasificador perfecto alcanza un valor de 0.39, con un intervalo de confianza similar, lo que sugiere un amplio margen para mejorar el rendimiento del modelo. Esto indica que los usuarios no se limitan únicamente a las propuestas populares, e introducir personalización en las recomendaciones podría mejorarlas. Por otro lado, el valor medio del $recall@5$ es de 0.39 ± 0.23.

En términos de métricas de *ranking*, el nDCG@10 alcanza una media de 0.38 ± 0.17 a lo largo de todos los folds. Por definición, un clasificador perfecto obtendría un nDCG de 1. El MAP@10 obtiene un valor de 0.28 ± 0.16.

La figura 5.3 ilustra cómo la precisión, tanto en el caso perfecto como en la línea

base, ha ido aumentando gradualmente con el tiempo, en paralelo al incremento de la participación. Sin embargo, el nDCG es muy variable.

Es importante destacar que, para el entrenamiento y la validación de los modelos, se han utilizado exclusivamente los últimos 10 folds disponibles. En estos folds finales, la $precision@5$ de la línea base alcanza un valor de 0.22 ± 0.04 y la del modelo perfecto alcanza 0.47 ± 0.06, un valor significativamente más alto que en el resto de folds. Por su parte, el nDCG@10 alcanza un valor de 0.42 ± 0.09, también mucho mejor.

Capítulo 6

Implementación y experimentos

6.1. Obtención de datos

Para la obtención de datos se ha utilizado el *DAO Analyzer dataset* [73], el único dataset sobre DAOs disponible en Kaggle o Zenodo a fecha de comienzo de este proyecto. Este dataset incluye datos de organizaciones en Aragon, DAOstack y DAOhaus, y en el que ha participado anteriormente el autor de este proyecto. Durante el desarrollo del TFM se ha modificado ligeramente el dataset para obtener la información textual (título y descripción) de las propuestas realizadas. Además, se ha aumentado este dataset con las DAOs de la plataforma Snapshot, utilizando parte de los scripts desarrollados durante la pasantía de verano de Andrew Schwartz en el Departamento de Ingeniería del Software e Inteligencia Artificial de la Universidad Complutense de Madrid, aunque también han necesitado ser modificados para descargar la información textual de las propuestas. Finalmente, los datos han sido publicados en Kaggle [86].

Debido a que sólo se han explorado organizaciones de las plataformas Aragon, DAOstack, DAOhaus y Snapshot, en esta sección solo se explorará la obtención de datos de estas plataformas, aunque las herramientas utilizadas son capaces de más.

En la figura 6.1 puede verse un resumen de las fuentes de los datos y los procesos realizados para la elaboración del dataset.

6.1.1. Aragon, DAOhaus y DAOstack

Para obtener los datos de las organizaciones de las plataformas Aragon, DAOstack y DAOhaus, el paquete dao-scripts [2] utiliza el servicio The Graph[1], una plataforma que permite indexar y consultar datos de Aplicaciones Descentralizadas (dApps, Decentralized Applications) de manera eficiente. Por como funciona la blockchain, la manera de conocer el estado actual de un programa en ejecución es descargarse toda la cadena de bloques y recorrerla bloque a bloque realizando las operaciones de reducción necesarias, de igual modo que para conocer el saldo de una cuenta bancaria es necesario computar todas las transferencias realizadas. The Graph provee de indexadores gratuitos que realizan esta tarea y dispone los resultados mediante una API en GraphQL, lo que evita tener un servidor indexando solo para nuestra aplicación.

[1] `https://thegraph.com/`

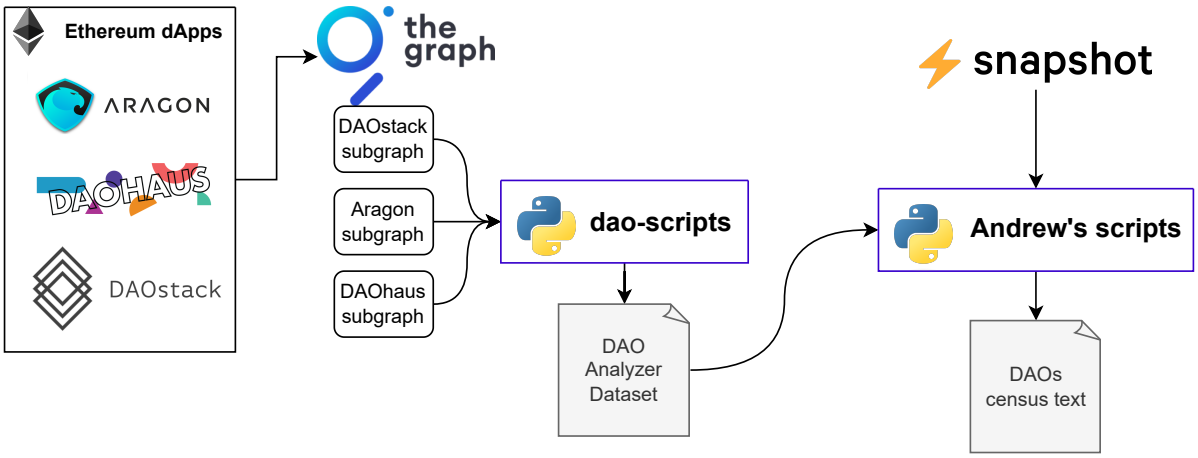


Figura 6.1: Esquema de la obtención de datos de las plataformas para su posterior análisis.

Para desplegar un subgrafo (cada uno de los repositorios de información indexada asociados a una dApp) es necesario programar en Typescript el código que ejecutará el indexador, y para ello es necesario tener acceso al código fuente de los contratos inteligentes que forman la aplicación. Esta tarea ya la realizaron los creadores de las tres plataformas y por lo tanto los subgrafos ya están desplegados.

Sin embargo, muchas de estas plataformas han abandonado su desarrollo, por lo que ha sido necesario realizar cierto mantenimiento y en ocasiones añadir información necesaria para que pueda ser obtenida por los scripts en los subgrafos de Aragon y DAOstack.

El programa que obtiene los datos de The Graph, los empaqueta en formato Arrow o Valores Separados por Comas (CSV, Comma-separated values) y los sube a Zenodo y Kaggle se llama dao-scripts y el código en Python está disponible en GitHub[2]. Los datos son obtenidos diariamente con un GitHub action y actualizados tanto en Zenodo como en Kaggle, en un conjunto conocido como *DAO Analyzer Dataset* y al que habría que añadir las organizaciones de Snapshot, como se explicará en la próxima sección. Los ficheros CSV que forman el conjunto de datos ocupan 130MB, de los cuales aproximadamente 100MB son la información textual de las propuestas.

6.1.2. Snapshot

En el caso de Snapshot, al ser una plataforma *off-chain*, es decir, no desplegada en la blockchain, no podemos utilizar The Graph. Sin embargo, Snapshot utiliza su propia API GraphQL[3], por lo que puede reutilizarse parte del código y de los algoritmos desarrollados en los dao-scripts. Sin embargo, debido al diferente esquema de la API [87], además de problemas de escalabilidad inherentes a su popularidad y la gran actividad con la que cuenta, Andrew Schwartz hizo estos scripts casi de cero en Jupyter Notebooks. Posteriormente, realicé un *fork* en GitHub[4] de dichos scripts, añadiendo algunas utilidades y, sobre todo, obteniendo la información textual de las organizaciones desplegadas en Snapshot.

[2] https://github.com/Grasia/dao-scripts
[3] https://hub.snapshot.org/graphql
[4] https://github.com/daviddavo/daos-verano

Finalmente, estos notebooks combinan los datos del *DAO Analyzer dataset* con los obtenidos de Snapshot para crear un nuevo conjunto de datos aumentado, denominado *DAOs Census TFM* y disponible en Kaggle [86]. Este nuevo dataset está formado por ficheros en formato Parquet debido a su escala, y ocupa 3.63 GB. En este caso, el gran espacio ocupado es debido a la información sobre los votos, que ocupa aproximadamente 3.55 GB. La información textual de las propuestas ocupa tan solo 45 MB, y el resto de datos de las propuestas y organizaciones ocupa 32 MB.

La sección 4.2 abordó la exploración de este conjunto de datos.

6.2. Preparación de datos

Para garantizar la reutilización del código de preparación de datos en todos los notebooks, se ha creado el módulo `src.datasets`, cuyo código está disponible en el GitHub del proyecto [68].

La extracción de los conjuntos de datos se realiza mediante la API de Kaggle, empleando el formato Apache Parquet para su almacenamiento. Dado el tamaño considerable de estos conjuntos, se ha optado por la utilización de DuckDB [88] para realizar consultas de filtrado, permitiendo así una manipulación eficiente de los datos. Posteriormente, se transforman los resultados en estructuras de datos manejables mediante Pandas DataFrame [89].

Este proceso de preparación de datos se encapsula en el método `get` del módulo `src.datasets.daocensus`. Dicho método, al ser invocado con el nombre específico de la DAO, devuelve dos DataFrames distintos: uno que contiene todas las propuestas y otro que contiene los votos relacionados con dichas propuestas. Además, esta función es capaz de realizar la descarga automática de los conjuntos de datos desde Kaggle si estos aún no se encuentran disponibles, y permite la aplicación de filtros según la plataforma y el número mínimo de votos por propuesta.

Además, se ha implementado un proceso de estandarización de los datos para adecuarlos al formato comúnmente utilizado por los modelos de la librería `microsoft/recommenders` [50]. Para ello, se ha desarrollado el método `to_microsoft`, el cual convierte el DataFrame de votos en un DataFrame de interacciones con tres columnas: *userID*, *itemID* y *rating* (siendo este último siempre igual a 1 debido a que se trata de un *feedback* implícito).

También se ha facilitado la carga del conjunto de datos en forma de grafo utilizando un formato basado en pares de tensores, para su uso en PyTorch Geometric [49]. A pesar de no haberse utilizado dicha librería en la implementación final, esta opción se mantiene disponible para futuras investigaciones o análisis que puedan requerir este tipo de representación de los datos.

6.3. Especificaciones de los sistemas recomendadores

Se han desarrollado dos sistemas recomendadores siguiendo dos enfoques distintos: uno basado en contenido y otro basado en la relación entre los usuarios y las propuestas, también conocido como Filtrado Colaborativo (CF, Collaborative Filtering).

Además, se ha explorado la posibilidad de crear un sistema recomendador híbrido que combine las recomendaciones generadas por estos dos modelos.

A continuación, se detallan en profundidad estos tres sistemas desarrollados para ofrecer una comprensión exhaustiva de sus características, funcionamiento y resultados.

Para el desarrollo de los sistemas se han utilizado únicamente los datos de la organización Decentraland, aunque en el capítulo 7 se presentan los resultados de utilizar el sistema en otras organizaciones.

6.3.1. Sistema basado en contenido

El enfoque del sistema basado en contenido se fundamenta en el análisis de la información textual asociada a las propuestas. En este contexto, la disponibilidad de datos se reduce al título y la descripción de cada propuesta. Es importante tener en cuenta que no todas las propuestas contienen este tipo de información. Para abordar este problema, se ha decidido un enfoque pre-entrenado que permita aprovechar modelos de PLN previamente entrenados en grandes conjuntos de datos textuales.

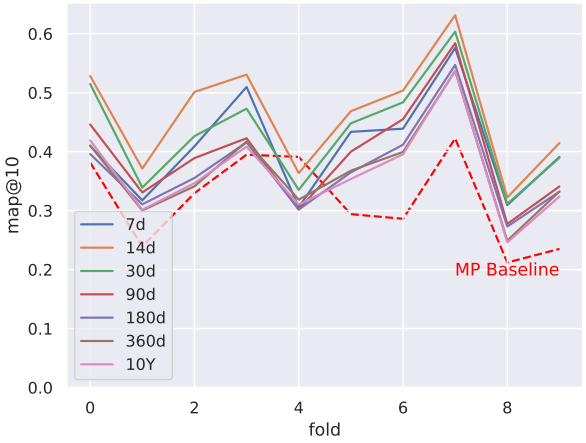
La selección de un modelo pre-entrenado se justifica por la moderada cantidad de datos disponibles y la naturaleza semi-supervisada del aprendizaje. Este enfoque ofrece la ventaja de utilizar representaciones de texto de alta calidad aprendidas en corpus de datos extensos, lo que puede mejorar significativamente el rendimiento del modelo.

Un aspecto clave en el procesamiento de la información textual de las propuestas es la naturaleza específica del lenguaje utilizado. En muchas ocasiones, las propuestas abordan temas técnicos con terminología especializada o incluyen jerga propia de la DAO. Esto plantea desafíos adicionales para el modelado del contenido, ya que un enfoque tradicional de bolsa de palabras (*bag-of-words*) podría resultar inadecuado. En su lugar, se requiere un enfoque que capture la semántica subyacente de las palabras y sea capaz de manejar términos derivados y neologismos específicos del dominio. Por ejemplo, aunque la palabra «Decentraland» no se encuentre en el corpus, debería tener una representación similar a la de «Descentralización».

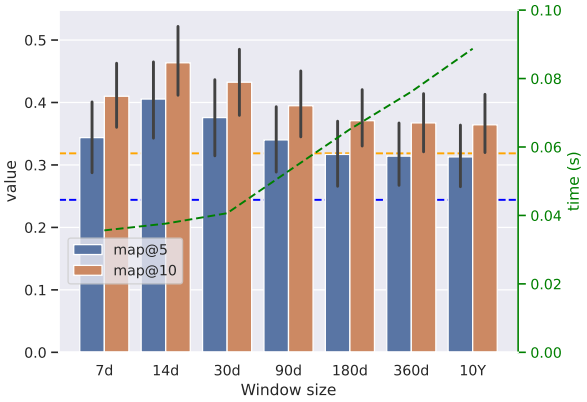
Comparado con el enfoque de filtrado colaborativo, el sistema basado en contenido presenta la ventaja de evitar el problema del arranque en frío (*cold start*) para las propuestas. Desde el momento de su creación, el sistema puede generar recomendaciones basadas en el contenido textual disponible. Sin embargo, se ha seguido un enfoque de evaluación similar al de los otros sistemas para garantizar la comparabilidad de los resultados y la robustez de la evaluación.

En la generación del espacio latente de las propuestas, se concatena el título y la descripción de cada propuesta para generar un *embedding* de texto representativo. Es importante destacar que este *embedding* permanece constante durante toda la vida útil de la propuesta, lo que permite generar recomendaciones desde el momento de su creación y mantener la consistencia en las recomendaciones a lo largo del tiempo.

El impacto del tiempo se refleja en la evolución del modelo de los usuarios, que puede ser influenciado por su interacción con nuevas propuestas. Sin embargo, los *embeddings* de las propuestas permanecen inalterados. En las secciones siguientes



(a) Precisión obtenida en cada fold.



(b) Media entre todos los folds. Las barras de error representan el IC.

Figura 6.2: Resultados del sistema recomendador basado en similitud con el usuario entre propuestas y usuarios dependiendo del tamaño de la ventana de propuestas en Decentraland. En rojo la línea base del clasificador más votado.

de este capítulo, se exploran dos modelos que hemos considerado para aplicar a cada usuario y realizar recomendaciones: el basado en similitud del coseno y el enfoque kNN.

Para la generación de los *embeddings* de las propuestas, se ha utilizado el modelo `all-mpnet-base-v2` de la librería de Python Sentence Transformers [90], que utiliza redes de tipo Representación de Codificador Bidireccional de Transformers (BERT, Bidirectional Encoder Representations from Transformers).

El código utilizado para desarrollar y evaluar estos modelos se encuentra disponible en el notebook `11_pln_tune` del repositorio GitHub del proyecto [68]. A partir de las pruebas realizadas en el notebook, se ha creado una clase del modelo de similitud de coseno, la cual está disponible en la clase `src.models.NLPSimilarity` del repositorio.

6.3.1.1. Similitud del usuario

Cada usuario está representado por un *embedding*, el cual se obtiene como la suma normalizada de los *embeddings* de las propuestas en las que ha emitido un voto, ya sea a favor o en contra. La recomendación está formada por las k propuestas con mayor similitud del coseno con el *embedding* del usuario.

Este enfoque se distingue por su rapidez tanto en el entrenamiento como en la ejecución. Dado que no requiere ajustar hiperparámetros, el proceso de entrenamiento es ágil y no implica iteraciones para encontrar configuraciones óptimas. Aunque en la implementación actual no se utiliza una ventana temporal, la operación de suma sobre una ventana temporal tiene un costo constante $\mathcal{O}(1)$ [91], lo que garantizaría eficiencia y escalabilidad, independientemente del tamaño del conjunto de datos.

El rendimiento medio del MAP@10 de este recomendador en todos los folds es de 0.36 ± 0.08. Se realizó un experimento para evaluar el impacto de reducir el número de propuestas utilizadas para realizar las recomendaciones y se observó que limitar las propuestas a aquellas en las que el usuario ha votado durante los últimos 30

días mejoraba significativamente la calidad de las recomendaciones en comparación con el uso de todo el conjunto de datos. Además, este enfoque reduce el tiempo de cálculo de los *embeddings* del usuario debido a la menor cantidad de propuestas a considerar.

Se llevó a cabo una búsqueda en rejilla (*grid search*) con diferentes tamaños de ventana, variando desde 7 días hasta 1 año. Los resultados se presentan en la figura 6.2. Se observó que el uso de una ventana de 14 días resulta óptimo, ya que el $map@10$ aumenta de 0.36 ± 0.08 a 0.46 ± 0.09, en comparación con el uso de todo el conjunto de propuestas. Además, el tiempo de ejecución se reduce de 86ms a 36ms. Hay que tener en cuenta que el tiempo de ejecución no incluye el cálculo de los *embeddings*, que está cacheado, aunque el servidor (véase A) tarda a penas 5 segundos en calcular los *embeddings* de las 1950 propuestas, utilizando el modelo `all-mpnet-base-v2`.

6.3.1.2. K vecinos más cercanos

En el enfoque de k vecinos más cercanos (kNN, k-nearest neighbors), cada usuario posee su propio modelo donde los datos son las propuestas en las que ha votado o no. Se intenta clasificar una nueva propuesta en dos clases (votará/no votará) mediante la agregación de los k vecinos más cercanos. El *score* de una predicción será el porcentaje de vecinos de la nueva propuesta en los que el usuario ha votado. Para esto, se ha empleado la clase `KNeighborsClassifier` de scikit-learn [80].

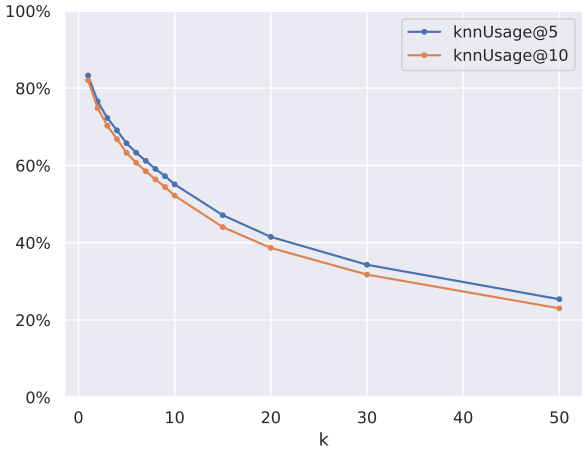
Dado el bajo número de muestras positivas, se realiza un muestreo donde el número de muestras negativas es igual al número de muestras positivas. Es importante destacar que, dado que el feedback es implícito, una muestra negativa no indica necesariamente que al usuario no le habría interesado la propuesta. Es probable que el usuario no haya votado en ella porque no estaba disponible en ese momento.

Debido a que muchos usuarios han emitido pocos votos, se realiza un *fallback* al modelo línea base Más Popular (véase 5.3) en el caso de que el valor de k en el kNN sea mayor que el número de propuestas en las que ha votado el usuario. Esto implica que a medida que k aumenta, se reduce el uso del kNN y se incrementa el del *fallback*, como puede observarse en la figura 6.3a.

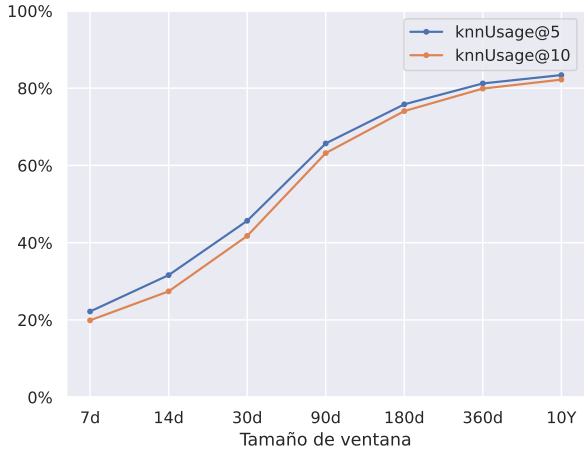
Al igual que en el modelo anterior, se ha decidido probar distintas ventanas temporales de entrenamiento para comprobar el rendimiento del modelo. El tamaño de esta ventana impacta en la cantidad de propuestas incluidas en el entrenamiento, afectando así el porcentaje de uso del kNN. Este efecto se observa en la figura 6.3b. También se ha probado el modelo con distintas métricas de distancia entre las muestras. Sin embargo, cambiar la métrica de distancia no afectará al porcentaje de uso del kNN.

Para utilizar kNN, es necesario llevar a cabo una búsqueda del hiperparámetro k. Además, usamos distintos tamaños de ventana de propuestas a considerar como en el modelo anterior, y probamos tanto con la distancia del coseno como la Euclídea (Minkowski con $p = 2$). Se ha realizado la búsqueda con una rejilla de 196 puntos, definida por el espacio en la tabla 6.1.

El resumen de los resultados de la búsqueda se muestra en la figura 6.4. En general, la elección de la medida de distancia apenas influye en los resultados, y el mejor valor de k parece ser 1 (*Nearest neighbor*) independientemente del tamaño de la ventana.



(a) Porcentaje medio de uso de knn para los distintos k probados.



(b) Porcentaje medio de uso de knn en cada tamaño de ventana.

Figura 6.3: Porcentaje de uso de kNN vs el *fallback* variando dos hiperparámetros.

Hiperparámetro	**Valores**
Window size	$\{7d, 14d, 30d, 90d, 180d, 360d, 10Y\}$
k vecinos	$[1, 10] \cup \{15, 20, 30, 50\}$
Métrica	{ Euclídea, Similitud del coseno }

Tabla 6.1: Espacio de búsqueda de hiperparámetros para la optimización del modelo basado en PLN y kNN.

Además, cuando k es demasiado alto, el modelo acaba recurriendo únicamente al *fallback*, lo que no ofrece recomendaciones personalizadas.

En cuanto al tiempo de ejecución (véase figura 6.5), este modelo es 300 veces más lento que el basado en similitud con el usuario, con tiempos de ejecución de alrededor de 15 segundos para los hiperparámetros que maximizan el MAP. A diferencia del otro modelo, que realiza una simple suma y calcula la distancia del usuario a todas las propuestas, este necesita calcular la distancia entre todas las propuestas entre sí. Se ha realizado un muestreo para eliminar ejemplos negativos, lo que sugiere que el tiempo de ejecución podría ser aún mayor sin este proceso.

6.3.2. Sistema basado en filtrado colaborativo

Esta estrategia se basa en aprovechar el grafo de relaciones entre los usuarios como parte del SR. Esta aproximación permite capturar conexiones sociales y patrones emergentes en la interacción entre usuarios, incluso cuando la información sobre las propuestas en sí es limitada. Sin embargo, estas propuestas se desenvuelven en comunidades, donde surgen dinámicas específicas y existen estructuras sociales latentes, al igual que en el caso del Software Libre [92].

En términos más específicos, se ha desarrollado un sistema de filtrado colaborativo basado en aprendizaje profundo, utilizando en particular una Graph Neural Network (GNN). La intención es poder expandir el grafo en el futuro incorporando otras interacciones de los usuarios, incluso aquellas que ocurren fuera de la organización

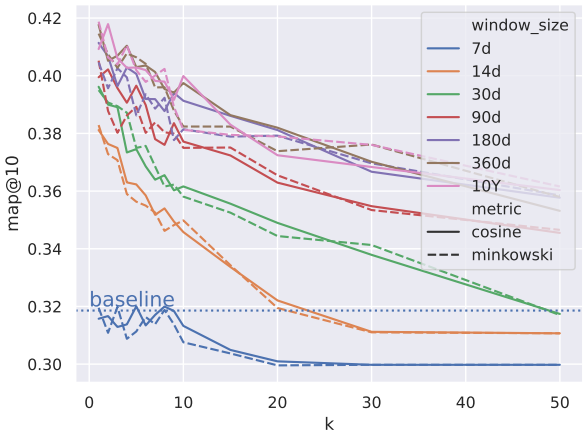


Figura 6.4: Resultados de la búsqueda de hiperparámetros para el modelo kNN

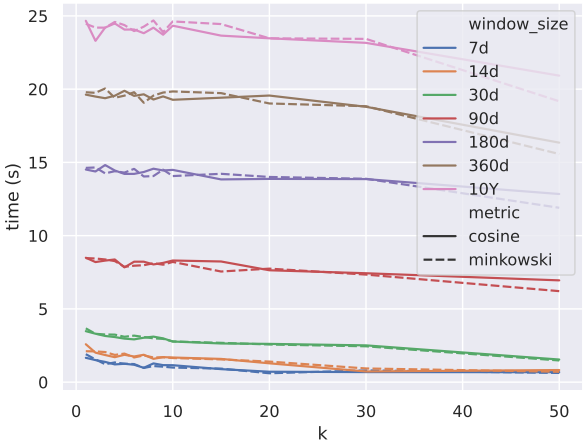


Figura 6.5: Tiempo de ejecución con varios hiperparámetros con el modelo kNN

o de otras dApps, como las transferencias de dinero. Para esta implementación, se ha optado por utilizar el modelo LightGCN [66], disponible en la librería microsoft/-recommenders [50]. Este modelo se ha seleccionado debido a su actualidad, rendimiento sólido, y su eficiencia computacional (similar a una factorización de matrices), además de haber sido probado en diversos conjuntos de datos y *benchmarks*.

Es importante subrayar que el propósito principal de este proyecto no radica en la creación ni en el desarrollo de nuevos modelos, sino en la aplicación de los Sistemas Recomendadores a las DAOs. El funcionamiento del modelo se explica en profundidad en la subsubsección 3.3.1.2.

En cuanto al entrenamiento y ajuste de hiperparámetros del modelo, se ha utilizado la librería Ray Tune [93]. Ha sido necesario modificar ligeramente las siguientes partes de la implementación del modelo:

- Se incorporó un post-filtrado que evita recomendar propuestas que no están abiertas. Para ello, se ajustó la función `recommend_k_items`, la cual ahora recibe una lista de propuestas *recomendables* (es decir, actualmente abiertas). Aquellas recomendaciones que no estén en esta lista se les asigna un *score* de $-\infty$, de ese modo al hacer el *ranking* de propuestas aparecerán las últimas y no serán recomendadas.
- Se introdujo el método `fit_epoch` para realizar el fit de una sola época, permitiendo la detención temprana y permitiendo reportar las métricas con mayor granularidad en Ray Tune.

Este modelo modificado está disponible en la clase `LightGCNCustom`, ubicada en el módulo `src.models.lightgcn` del repositorio del TFM en GitHub [68].

El espacio de búsqueda de hiperparámetros es continuo, por lo que se realiza una búsqueda metaheurística en este espacio usando HyperOpt [94], pues es el único algoritmo de búsqueda de Ray Tune que permite la búsqueda en un espacio que mezcle hiperparámetros con valores continuos y discretos. En la tabla 6.2 se muestran los valores que definen el espacio de búsqueda utilizado.

Asumimos que el modelo se reentrena cada vez que se requiere actualizar las reco-

Hiperparámetro	Valores	Muestreo
Batch size	$bs \in \{64, 128, 256, 512, 1024\}$	Uniforme
Embedding dim	$e \in \mathbb{N}, 1 \leq e \leq 1024$	Loguniforme
Capas de convolución	$c \in \{1, 2, 3, 4, 5, 6\}$	Uniforme
Learning rate	$lr \in \mathbb{R}, 10^{-4} \leq lr \leq 1$	Loguniforme
Regularización L2	$l2 \in \mathbb{R}, 10^{-7} \leq l2 \leq 10^{-2}$	Loguniforme

Tabla 6.2: Espacio de búsqueda de hiperparámetros para la optimización del modelo LightGCN.

mendaciones. En consecuencia, no tiene sentido utilizar el mismo conjunto de hiperparámetros para el modelo en diferentes folds; se considera posible ajustar los hiperparámetros para generar nuevas recomendaciones. Por lo tanto, se optimiza la búsqueda de hiperparámetros en cada uno de los folds.

Sobre la generación de estos folds, se proporciona una explicación detallada en la sección 5.1. Para cada uno de los 10 folds, se realizan 100 muestras. Las primeras 20 muestras son el punto de partida de HyperOpt. Aunque normalmente se utilizan muestras aleatorias, uno de los puntos de inicio es la mejor muestra del fold anterior.

La búsqueda intenta maximizar la métrica objetivo MAP@10. Se observó que utilizar la función de pérdida Bayesian Personalized Ranking (BPR) generaba sobreaprendizaje, ya que resultaba en un valor de pérdida muy bajo, pero la precisión no alcanzaba niveles aceptables por encima de la línea base. Además, intentar optimizar la precisión o exhaustividad tampoco daba buenos resultados al ignorar por completo el orden de las recomendaciones. Sin embargo, un mejor MAP sí se corresponde con una mejor precisión, como puede verse en la figura 6.6. También podría haberse usado el nDCG, pues son métricas similares y que están correlacionadas, como puede observarse en la figura 6.7.

Todo el trabajo realizado se encuentra documentado en dos notebooks disponibles en el repositorio de GitHub del proyecto [68]. El primero, denominado `07_microsoft_tuning.ipynb`, aborda la definición del proceso de entrenamiento del modelo. El segundo, `09_analyze_results`, presenta los resultados de la ejecución y genera las métricas y gráficas expuestas en las siguientes secciones.

La decisión de utilizar dos notebooks se tomó para permitir la ejecución en paralelo del segundo y monitorizar el estado de ejecución del primero, ya que el experimento en su totalidad requiere varias horas para completarse.

6.3.2.1. Tiempo de ejecución

El experimento se configura para realizar 100 muestras en 10 folds, con cada ejecución limitada a 5 minutos (300 segundos). Por ende, el tiempo de ejecución teórico asciende a 5,000 minutos, aproximadamente 3 días y medio. No obstante, en la práctica, la suma de los tiempos de ejecución de los experimentos es de 3 días y 20 horas. Esto se debe a que, en lugar de interrumpir abruptamente el experimento, se permite que no continúe tras concluir el *epoch*, y el tiempo medio de ejecución se sitúa en 330 segundos.

El tiempo real de ejecución del experimento es de aproximadamente 5 horas y 45

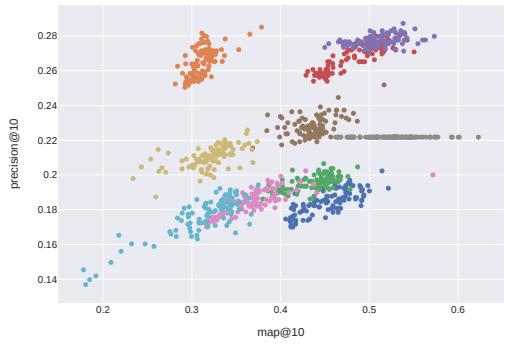


Figura 6.6: Gráfico de dispersión entre el MAP y la precisión en las 1000 muestras tomadas para la optimización del recomendador basado en GNN para la DAO Decentraland. Dentro de cada fold (color), aumentar el MAP hace aumentar la precisión.

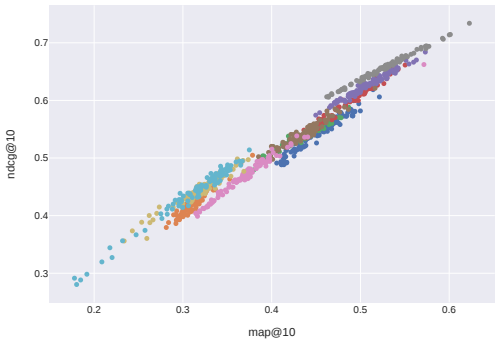


Figura 6.7: Gráfico de dispersión entre el MAP y el nDCG en las 1000 muestras tomadas para la optimización del recomendador basado en GNN para la DAO Decentraland. El color es distinto para cada uno de los folds, mostrando que cada fold tiene unas métricas máximas alcanzables distintas.

minutos. Esto se debe a que el servidor utilizado (consultar el apéndice A) permitía la ejecución de 16 entrenamientos en paralelo sin afectar el tiempo de ejecución de otros entrenamientos.

6.3.2.2. Emisiones del experimento

Según la información proporcionada por el programa nvidia-smi, la tarjeta gráfica consumía 250W durante la ejecución de los experimentos. Además, de acuerdo con las especificaciones del procesador [95], este tiene una potencia base de 150W, aunque no se estaba utilizando al 100 %.

Por lo tanto, la ejecución del experimento consumió 2.3KWh, equivalente a 0.23KWh por cada *fold* como cota superior. En términos de emisiones, actualizar las recomendaciones resultaría en una emisión de 630g de CO_2 al utilizar una comercializadora en España sin Garantías de Origen renovable (GdO) [96]. En caso de actualizar las recomendaciones semanalmente, las emisiones ascenderían a aproximadamente 33kg de CO_2. Cabe destacar que, al ubicarse el servidor en la Facultad de Informática de la Universidad Complutense de Madrid, que ha adoptado electricidad de origen verde desde 2019, no se generan emisiones [97].

6.3.2.3. Evaluación del modelo

No podemos emplear una búsqueda de hiperparámetros calculando el promedio de los resultados de todos los folds, ya que implicaría asumir dos supuestos cruciales: en primer lugar, que todos los modelos de cada fold deben tener los mismos hiperparámetros; y en segundo lugar, que los resultados de cada fold estarían disponibles para evaluar los anteriores.

Por lo tanto, optamos por realizar la optimización de hiperparámetros de manera

independiente en cada fold. De las 100 muestras obtenidas durante la ejecución de la optimización de hiperparámetros de un fold, seleccionamos la mejor y la evaluamos utilizando dichos hiperparámetros en el siguiente fold.

Esta metodología busca simular el comportamiento de la optimización de hiperparámetros en un entorno real, donde se dispone de los datos hasta el momento actual pero se desconocen los datos futuros. Por ende, optimizamos los hiperparámetros utilizando únicamente los datos disponibles hasta el momento.

Es crucial resaltar que, para calcular la métrica objetivo de la optimización de hiperparámetros, utilizamos un conjunto de validación que, supuestamente, no está disponible en el momento del entrenamiento. Por esta razón, al modelo optimizado le llamamos «Oráculo». Al modelo que emplea los hiperparámetros que obtienen la mejor métrica en el *Oráculo* del fold anterior, lo denominamos «Realista».

En la figura 6.8 se exponen los resultados de estas operaciones en los últimos 10 folds, comparando estos modelos con una línea base (recomendar las propuestas más votadas hasta el momento, véase 5.3). La línea base arroja una MAP@10 media de 0.32 ± 0.08, mientras que el modelo Oráculo está altamente optimizado con un valor medio de 0.47 ± 0.08. El modelo Realista se posiciona entre ambos, con un valor de 0.37 ± 0.08. Aunque siempre es preferible utilizar el modelo Oráculo, el modelo Realista demuestra un rendimiento notable al superar en general a la línea base. En el capítulo 7 se lleva a cabo una comparación entre los resultados obtenidos por el modelo Realista y el basado en PLN y el enfoque Híbrido.

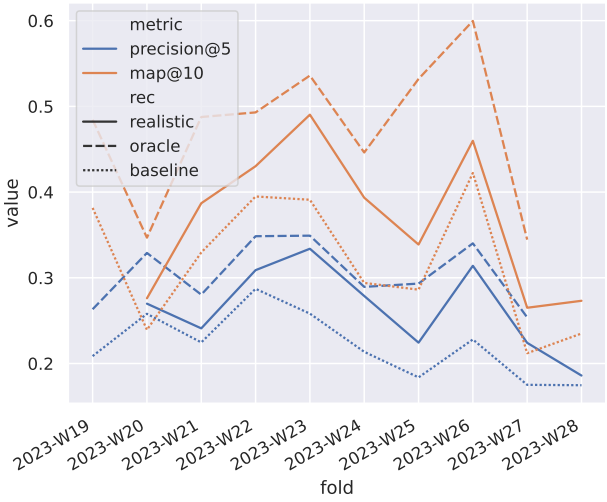


Figura 6.8: Resultados del entrenamiento *realista* del SR basado en GNN.

6.3.3. Sistema híbrido

El sistema híbrido propuesto combina las recomendaciones generadas por los dos sistemas presentados en los apartados anteriores. En lugar del modelo kNN utilizado en la recomendación basada en contenido, se optó por emplear el modelo basado en la similitud del coseno con el *embedding* del usuario, debido a su mejor desempeño y rapidez. Se emplearon los hiperparámetros optimizados para cada sistema individual, Sin realizar una búsqueda adicional de hiperparámetros para el sistema híbrido. El código de este modelo se encuentra en la clase `src.models.hybrid.HybridRecommendation`, y se ha evaluado su funcionamiento en el notebook `12_hybrid.ipynb`.

Para combinar los resultados de ambos sistemas, se solicitan k recomendaciones a cada uno y se intercalan para formar un conjunto de k recomendaciones. En caso de que existan recomendaciones duplicadas, es decir, aquellas en las que coinciden ambos sistemas, se han definido tres métodos de fusión, que se detallarán en la

siguiente subsección. En la figura 6.9 ofrece una visualización de estos tres métodos.

6.3.3.1. Métodos de fusión

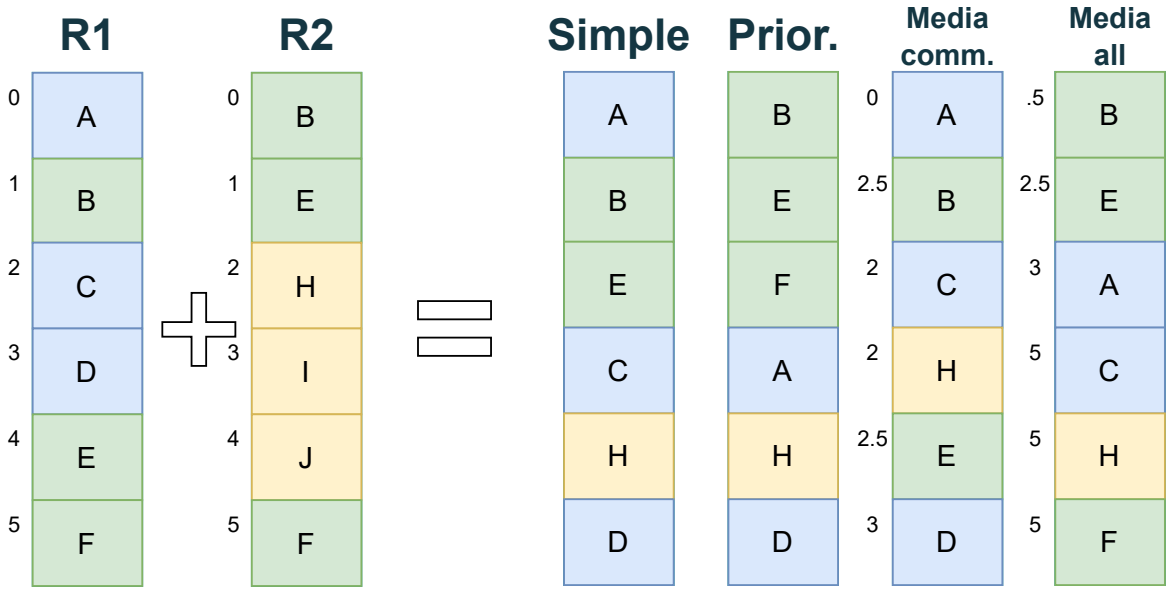


Figura 6.9: Visualización de los distintos métodos de fusión implementados de dos grupos de recomendaciones. En azul las recomendaciones exclusivas al primer recomendador, en amarillo las recomendaciones exclusivas del segundo, y en verde las recomendaciones comunes. Una recomendación superior será una recomendación igual o mejor que la inferior, aunque no son comparables entre recomendadores. El número de la izquierda indica la posición, o el *score* del sistema basado en la posición media.

La fusión de los dos sistemas recomendadores se basa en el entrelazamiento de las recomendaciones de ambos, es decir, se selecciona una recomendación de cada sistema hasta alcanzar k. Sin embargo, la manera en que tratamos las recomendaciones que aparecen en ambos sistemas puede variar, y a continuación se describen cuatro métodos de tratar estos duplicados:

Entrelazado simple Se realiza el entrelazado de las recomendaciones y luego se eliminan los elementos duplicados. Es decir, la prioridad asignada a un elemento común será la máxima de las prioridades generadas por los dos recomendadores.

Priorizar los comunes Se priorizan primero los elementos comunes, asumiendo que si ambos recomendadores coinciden en una recomendación, es porque se considera una buena opción. Los elementos no comunes, son entrelazados.

Posición media de los comunes Se calcula la posición media de los elementos comunes en ambos sistemas, y luego se seleccionan los elementos según su *ranking* en esta posición media. Los elementos que no son comunes no varían su *puntuación*. Un elemento duplicado que aparece en la primera posición de un recomendador y en la tercera del otro, tendrá una puntuación de 1, por lo que debería aparecer entre el primero y el segundo.

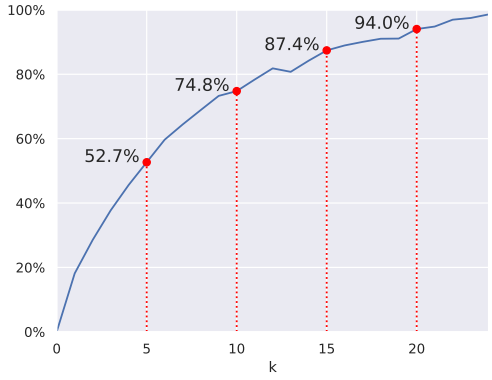


Figura 6.10: Porcentaje de propuestas en común entre el recomendador basado en contenido y el de filtrado colaborativo aumentando el número de recomendaciones.

	precision@k		recall@k		MAP@k		nDCG@k	
Model	k=5	k=10	k=5	k=10	k=5	k=10	k=5	k=10
OpenPop	0.221	0.196	0.351	0.626	0.236	0.310	0.317	0.416
avg. all	0.292	0.222	0.526	0.751	0.347	0.398	0.458	0.515
avg.	0.287	0.224	0.514	0.759	0.341	0.399	0.449	0.518
prioritize	0.291	0.224	0.525	0.755	0.341	0.399	0.452	0.520
naive	0.289	0.224	0.521	0.757	0.353	0.426	0.460	0.540
GNN	0.271	0.216	0.485	0.736	0.334	0.409	0.434	0.522
NLP	0.295	0.227	0.513	0.765	0.357	0.434	0.464	0.549

Tabla 6.3: Resultados media de los distintos métodos de fusión del sistema recomendador híbrido y comparación con línea base y los otros dos sistemas recomendadores.

Posición media Al igual que en el método anterior, se calcula la posición media de los elementos comunes entre ambos sistemas. Sin embargo, como asumimos que las recomendaciones que no aparecen en ambos sistemas puede que no sean tan buenas, también se calcula su posición media. Como no se conoce la posición real en el otro recomendador, se asume que estarían en la posición $i+k$, donde i es la posición en el recomendador en el que aparecen, y k es el número de elementos comunes. Es decir, se penaliza la posición de los elementos no comunes agregándoles $k/2$ a su posición.

6.3.3.2. Evaluación del recomendador híbrido

El enfoque híbrido propuesto combina los resultados de los SRs basados en contenido y en filtrado colaborativo, buscando alcanzar un equilibrio entre ambos enfoques. Este método podría proporcionar un resultado intermedio entre los resultados individuales de cada sistema, aumentando la estabilidad de las recomendaciones.

Al incrementar el número de recomendaciones generadas (k), es probable que aumente la cantidad de recomendaciones duplicadas entre los sistemas. A medida que se generan más recomendaciones, disminuye la relevancia del método elegido para fusionar los dos sistemas. Con tan sólo 5 recomendaciones, el 50 % son comunes entre ambos sistemas, y con 10 este número aumenta al 74 %, como se puede observar en la figura 6.10. De hecho, la precisión solo varía en una centésima según el método elegido (tabla 6.3). Sin embargo, sí que será relevante la manera en la que se ordenan estas recomendaciones comunes para el MAP y el nDCG.

Todos los métodos benefician ligeramente al sistema basado en PLN, debido a que es seleccionado primero durante el entrelazado en el proceso de fusión, como puede

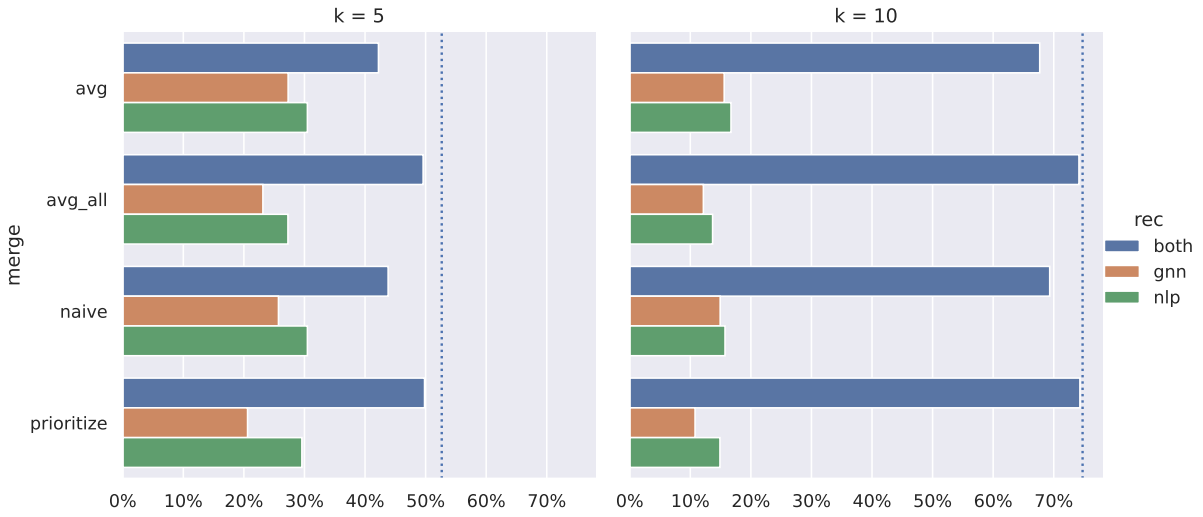


Figura 6.11: Porcentaje de propuestas de cada recomendador elegidas por cada uno de los métodos de fusión del recomendador híbrido (media de todos los folds). En las 5 recomendaciones originales, el porcentaje de propuestas duplicadas es 50.6%, y con 10 recomendaciones hay un 74% de propuestas duplicadas.

verse en la figura 6.11. Debido a que este es el mejor sistema de los dos, el método que más priorice las propuestas del PLN sobre las otras dará mejores resultados. El método que más elige las propuesas comunes es, obviamente, *prioritize*, seguido de avg_all, los dos métodos pesimistas sobre las recomendaciones realizadas por uno solo de los sistemas. Parece que el mejor de todos es *naive*, que podríamos considerar que en lugar de agregar utilizando la operación *media* utiliza la operación *máximo*. Puede que este sea el mejor método (tabla 6.3) porque tiende un poco más a escoger propuestas del recomendador PLN. Como puede verse en la figura, al aumentar el número de recomendaciones de 5 a 10, el método de fusión es menos relevante, pues siempre hay más de un 65% de recomendaciones en común.

Finalmente, como se puede observar en la tabla 6.3 el recomendador híbrido tiene unos resultados entre medias de los dos recomendadores que lo forman, aunque en el map@5 supera al pln por una centésima.

Sin embargo, como se puede ver en la figura 6.12 en el fold W19 el método *proritize* supera al naive en map@10, y en los folds W22 y W23 es el *avg* el que lo supera. Por lo tanto, el sistema recomendador híbrido podría superar a ambos de elegir de mejor manera cual de los dos sistemas base priorizar. La figura 6.13 muestra los resultados de el mejor de los híbridos y los compara con los otros sistemas. El sistema *naive*, que obtiene los mejores resultados de media, se encuentra entre los resultados del modelo GNN y el NLP excepto en el fold W23. En esta figura también se muestra con un sombreado azul los mejores y peores resultados obtenido por cualquier método de fusión, y en este caso sí que se supera en varios folds al modelo NLP. De hecho, de escoger en cada caso el mejor método de fusión, se conseguiría un MAP@10 medio de 0.437, superando en 3 centésimas al modelo NLP y demostrando que una mejor hibridación podría llegar a superar los resultados de ambos modelos base desarrollados.

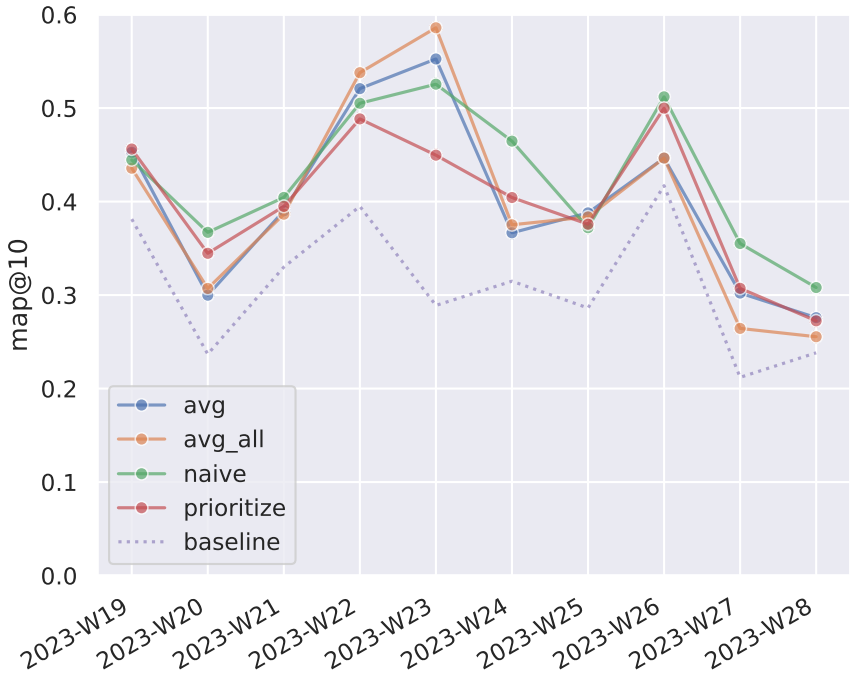


Figura 6.12: Resultados del sistema recomendador híbrido que usa los distintos métodos de fusión en cada fold, comparado con la línea base OpenPop.

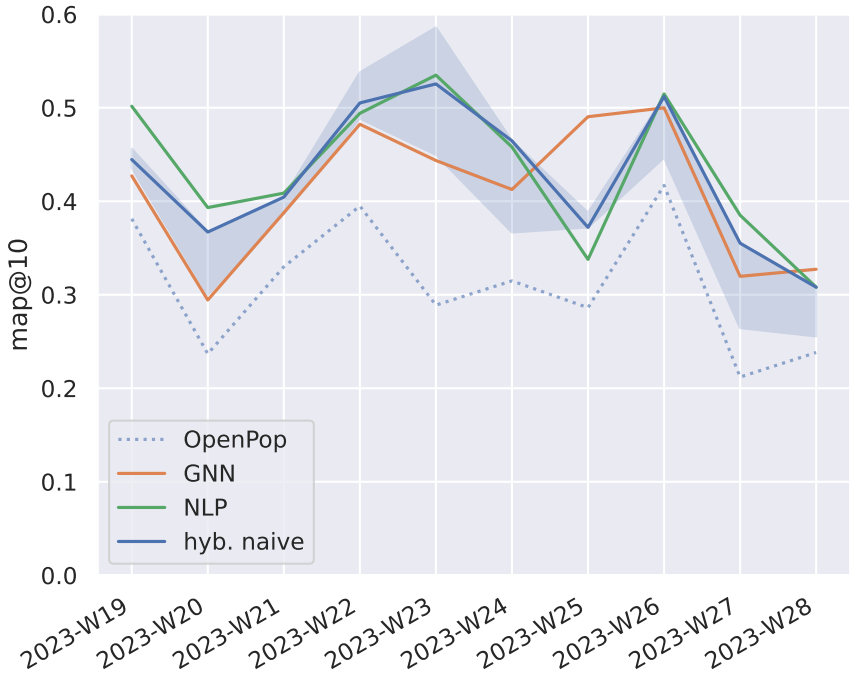


Figura 6.13: Resultados del sistema recomendador híbrido *naïve* comparado con los otros sistemas recomendadores desarrollados. El área azul representa el mejor y peor resultado obtenido en ese fold por cualquiera de los otros métodos de fusión.

Capítulo 7

Resultados y discusión

Para desarrollar el modelo se han utilizado las últimas 10 semanas del conjunto de datos de la organización *Decentraland*, que cuenta con 117 mil votos emitidos por 7300 votantes en 1942 propuestas. El modelo se entrena con una división en folds temporales incrementales, por lo que cada semana el número de propuestas y votos en el conjunto de entrenamiento aumenta, aunque solo se utilizan para evaluación las propuestas que están abiertas en el momento de hacer el split, como se detalla en la sección 5.1.

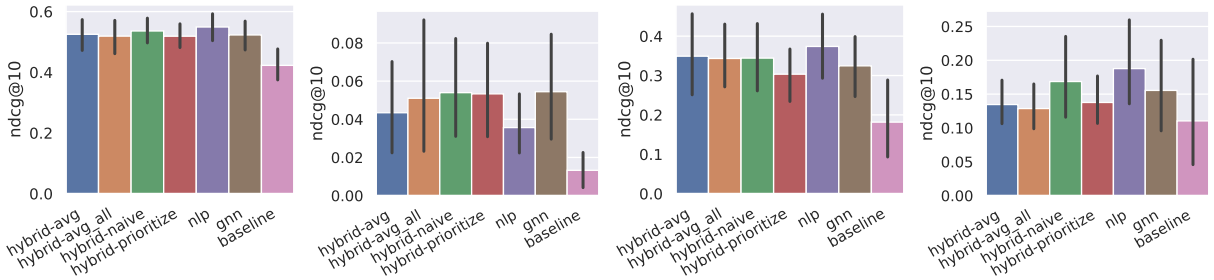
Los resultados fold a fold del entrenamiento de los modelos pueden consultarse en el capítulo 6. Tras desarrollar el modelo usando Decentraland como objetivo, se ha seguido el proceso para DAOs seleccionadas de la tabla 4.2. Para seleccionar las organizaciones, se han eliminado las organizaciones marcadas como SPAM en su plataforma. Además, se ha explorado el conjunto de datos para observar cual es la distribución de la duración de las propuestas, y se han realizado las recomendaciones de manera acorde.

En el caso de Decentraland, se utilizan los últimos 10 folds disponibles. Sin embargo, esto no es posible en todas las organizaciones, por lo que se ha buscado un rango de fechas tal que los folds tengan cierta cantidad de propuestas abiertas. Algunas organizaciones, aún teniendo muchas propuestas, no cumplen estas condiciones, por lo que tampoco se ha ejecutado el sistema recomendador en ellas. Finalmente, las DAOs con las que se ha probado el modelo quedan reflejadas en la tabla 7.1. En cuanto a Aave, no ha sido posible realizar el experimento debido a que cuenta con demasiadas propuestas. Para probar cada una de las organizaciones se ha creado el Jupyter

Nombre	**Entrenamiento**	**Fecha últ. fold**
Decentraland	Semanal (Jueves)	2023-07-28[†]
DEAD Foundations DAO	Cada 2 días	2021-11-28
MetaCartel / MetaCartel Ventures	Semanal (Jueves)	2022-07-22
PancakeSwap	Cada 3 días	2023-07-01
Aave / Aavegotchi	Cada 5 días	2023-07-28[†]

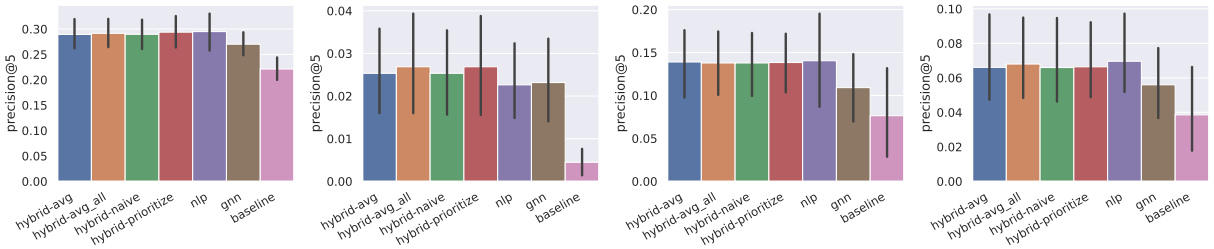
[†] Es el último fold disponible en el conjunto de datos.

Tabla 7.1: Organizaciones sobre las que se han entrenado los modelos.



(a) Decentraland (b) DEAD Foundations (c) Metacartel (d) PancakeSwap

Figura 7.1: Resultados de la métrica ndcg@10 en las organizaciones probadas.



(a) Decentraland (b) DEAD Foundations (c) Metacartel (d) PancakeSwap

Figura 7.2: Resultados de la métrica precision@5 en las organizaciones probadas.

Notebook `31_run_all.ipynb`, que ejecuta los notebooks necesarios para cada una de las DAOs seleccionadas. Los resultados de ejecución de dichos notebooks están disponibles en el GitHub del proyecto [68], bajo la carpeta `nbout`.

Nótese que debido al bajo número de propuestas *relevantes*, algunas métricas de recuperación de información nunca llegarán a tener un valor de 1 si el valor de la k no es muy bajo. Para solventar este problema, los clasificadores se comparan no solo entre sí, si no también con un clasificador perfecto que nunca falla (y reporta, por lo tanto, la métrica máxima alcanzable), y un clasificador línea base llamado *OpenPop* que recomienda siempre la propuesta abierta más votada en el momento de realizar la recomendación (véase la sección 5.3).

Tras ejecutar el experimento en las 4 organizaciones seleccionadas, en todas ellas los modelos desarrollados superan la linea base establecida, tanto para el ndcg@10 (figura 7.1) como para la precisión@5 (figura 7.2). En general, el modelo basado en PLN logra mejores resultados que el basado en GNN o el híbrido.

El modelo GNN, basado en filtrado colaborativo, demuestra el desafío de arranque en frío (*cold-start*) que sufren este tipo de modelos para cada una de las propuestas que se crean continuamente, y que cuentan con pocas interacciones. Como se detalla en la subsección 3.3.2, los sistemas basados en contenido sufren menos dicho problema. Aunque presentan el problema de *cold-start* con usuarios, en este tipo de organizaciones el número de usuarios es relativamente estable.

En cuanto al híbrido, en general logra unos resultados que se sitúan entre el modelo basado en contenido y el basado en filtrado colaborativo. En cuanto a los distintos métodos de fusión de las recomendaciones, las otras organizaciones tienen unas conclusiones similares al caso de Decentraland (véase la subsección 6.3.3): el método de

entrelazado simple parece lograr mejores resultados de *ranking* porque prioriza el PLN, que es el mejor modelo.

La única excepción de entre las organizaciones usadas es DEAD Foundations, en el que el sistema híbrido supera al basado en GNN, que a su vez supera al basado en contenido. Esta DAO, además, cuenta con una línea base especialmente baja, y es de hecho la organización con menor densidad ($\simeq 2‰$), como se muestra en la tabla 4.2. Este hecho podría indicar que un enfoque híbrido es el camino a seguir para realizar mejores recomendaciones en organizaciones dispersas, que sí que pueden contar con ambos problemas de *cold-start*.

Aunque en todas las organizaciones probadas se supere la linea base, es necesario considerar el contexto específico de cada organización para seleccionar o desarrollar el mejor enfoque de recomendación, analizando su actividad y las características de los usuarios y las propuestas.

Capítulo 8

Conclusiones y trabajo futuro

8.1. Conclusiones

Para realizar este trabajo, se ha definido el problema de la recomendación de propuestas en DAOs, y se han creado distintos elementos para evaluar este problema de manera agnóstica al modelo, usando datos históricos (evaluación offline): una línea base robusta inspirada en MostPop acorde a este caso de uso llamada OpenPop, y una manera de realizar validación cruzada con distintos subconjuntos de datos del conjunto de datos, basada en el tiempo.

Para comprobar dicha metodología, se han desarrollado tres sistemas recomendadores: un primer sistema, basado en el contenido, utilizando *embeddings* de la información textual de las propuestas; un segundo de filtrado colaborativo basado en LightGCN; y un tercer recomendador, híbrido, que fusiona los otros dos. Los tres sistemas han superado satisfactoriamente la linea base establecida, siendo el basado en contenido el que mejores resultados ha obtenido, y obteniendo el híbrido resultados entre medias de los otros dos.

Finalmente, se ha comprobado la metodología con otras 3 organizaciones, superando en todas ellas la línea base establecida.

8.1.1. Limitaciones

Debido a que se ha utilizado evaluación offline, desconocemos cómo funcionaría el sistema en un entorno real, y si este afectaría al comportamiento de los usuarios. Aunque en otros tipos de comunidades en línea haya aumentado la participación, no tiene por que ser así en este caso. Además, se han utilizado pocos datos de los presentes en el conjunto de datos.

En el caso del modelo basado en GNN (y, por lo tanto, el híbrido también), deben ser reentrenados para realizar recomendaciones, con sus debidos costes. No se ha comprobado el comportamiento de utilizar en el siguiente fold temporal un modelo preentrenado.

Dentro de la evaluación offline, no se han comprobado métricas que pueden proporcionar información más interesante para explicar las recomendaciones realizadas, como la diversidad, la novedad, serendipia, o la *explained variance*. Además, la diferencia entre la linea base OpenPop, sin personalización, y el resto de modelos proba-

dos es muy baja, pues los pocos elementos más votados sesgan las métricas basadas en *top-k recommendations* [98].

8.1.2. Trabajo publicado

El código realizado está disponible en GitHub [68], y el conjunto de datos en Kaggle [86]. Los resultados de la búsqueda y optimización de hiperparámetros de los modelos (`ray_results`) están disponibles en Zenodo [99].

Finalmente, se ha escrito y enviado un artículo para el *18th ACM Conference on Recommender Systems*, aún pendiente de revisión [100].

8.2. Trabajo futuro

Una vez comprobado que ambos enfoques proporcionan recomendaciones similares, debería explorarse mejor el uso de otros modelos, especialmente híbridos, como pueden ser las Factorization Machines como xDeepFM [101], o algoritmos de GNN para grafos etiquetados. Como la línea base logra muy buenos resultados a bajo coste, podría explorarse el uso de modificaciones con personalización como TimePop [102]. Dado que la información textual parece ser altamente relevante, pueden realizarse recomendaciones utilizando Grandes Modelos de Lenguaje (LLMs, Large Language Models) [103]. Como se ha podido ver, las distribución temporal de votos en propuestas no es uniforme, ni según el tiempo absoluto (día de la semana), ni según el tiempo relativo (horas desde que se abrió la propuesta), por lo que debería añadirse esa información contextual al sistema recomendador. El grafo utilizado para el filtrado colaborativo también puede expandirse utilizando interacciones entre los usuarios fuera de la DAO, pues esos datos son accesibles en el blockchain.

Recientemente se ha añadido soporte para sistemas recomendadores en la librería PyTorch Geometric [49], y en su hoja de ruta está el añadir soporte para grafos temporales, por lo que sería buena idea intentar utilizar sus modelos, pero siguiendo utilizando el *framework* de evaluación de Microsoft Recommenders [50].

Para mejorar la escalabilidad del modelo, deberían poderse realizar recomendaciones sin la necesidad de reentrenarlo completamente, o bien cambiando el tipo de modelo, o bien comprobando el comportamiento del modelo actual a la reanudación del entrenamiento con nuevos datos.

Finalmente, el sistema podría desplegarse de manera descentralizada, en alguna plataforma como Golem Network [104].

Bibliografía

[1] Samer Hassan y Primavera De Filippi. «Decentralized Autonomous Organization». En: *Internet Policy Review* 10.2 (20 de abr. de 2021). 122 citations (CrossRef 2024/5/19). ISSN: 2197-6775. DOI: 10.14763/2021.2.1556. URL: https://policyreview.info/glossary/DAO (visitado 07-07-2023).

[2] Javier Arroyo et al. «DAO-Analyzer: Exploring Activity and Participation in Blockchain Organizations». En: *Companion Publication of the 2022 Conference on Computer Supported Cooperative Work and Social Computing*. CSCW'22 Companion. 2 citations (CrossRef 2024/5/19). New York, NY, USA: Association for Computing Machinery, 8 de nov. de 2022, págs. 193-196. ISBN: 978-1-4503-9190-0. DOI: 10.1145/3500868.3559707. URL: https://dl.acm.org/doi/10.1145/3500868.3559707 (visitado 14-11-2023).

[3] Jakob Nielsen. *Participation Inequality: The 90-9-1 Rule for Social Features*. Nielsen Norman Group. 8 de oct. de 2006. URL: https://www.nngroup.com/articles/participation-inequality/ (visitado 11-03-2024).

[4] Benny Geys. «Explaining voter turnout: A review of aggregate-level research». En: *Electoral Studies* 25.4 (1 de dic. de 2006). 514 citations (CrossRef 2024/5/19), págs. 637-663. ISSN: 0261-3794. DOI: 10.1016/j.electstud.2005.09.002. URL: https://www.sciencedirect.com/science/article/pii/S0261379405000910 (visitado 10-03-2024).

[5] Hanna Halaburda y Christoph Mueller-Bloch. «Will We Realize Blockchain's Promise of Decentralization?» En: *Harvard Business Review* (4 de sep. de 2019). Section: Technology and analytics. ISSN: 0017-8012. URL: https://hbr.org/2019/09/will-we-realize-blockchains-promise-of-decentralization (visitado 11-12-2023).

[6] Joshua Z. Tan et al. *Open Problems in DAOs*. 29 de oct. de 2023. DOI: 10.48550/arXiv.2310.19201. URL: http://arxiv.org/abs/2310.19201 (visitado 14-11-2023).

[7] Vitalik Buterin. *Notes on Blockchain Governance*. 17 de dic. de 2017. URL: https://vitalik.ca/general/2017/12/17/voting.html (visitado 04-07-2023).

[8] Isaac Patka. «Exploiting Inattention & Optimism in DAOs». Devcon 6. Oct. de 2022. URL: https://archive.devcon.org/archive/watch/6/exploiting-inattention-and-optimism-in-daos (visitado 26-02-2024).

[9] Stephan Tual. *Ethereum Launches*. Ethereum Foundation Blog. 30 de jun. de 2015. URL: https://blog.ethereum.org/2015/07/30/ethereum-launches (visitado 11-04-2024).

[10] Kaidong Wu et al. «A first look at blockchain-based decentralized applications». En: *Software: Practice and Experience* 51.10 (2021). 49 citations (CrossRef

2024/5/19), págs. 2033-2050. ISSN: 1097-024X. DOI: 10.1002/spe.2751. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2751 (visitado 11-04-2024).

[11] Vitalik Buterin. *DAOs, DACs, DAs and More: An Incomplete Terminology Guide*. Ethereum Foundation Blog. 6 de mayo de 2014. URL: https://blog.ethereum.org/2014/05/06/daos-dacs-das-and-more-an-incomplete-terminology-guide (visitado 14-11-2023).

[12] Jacob Kastrenakes. *Crypto collective raises $27 million to bid for rare copy of US Constitution*. The Verge. 17 de nov. de 2021. URL: https://www.theverge.com/2021/11/17/22787993/constitutiondao-crypto-buy-us-constitution-copy-sothebys-ethereum (visitado 11-04-2024).

[13] Andrea Peña-Calvin et al. «A Categorization of Decentralized Autonomous Organizations: The Case of the Aragon Platform». En: *IEEE Transactions on Computational Social Systems* (2023), págs. 1-. ISSN: 2329-924X. DOI: 10.1109/TCSS.2023.3299254. URL: https://ieeexplore.ieee.org/abstract/document/10217072 (visitado 30-01-2024).

[14] Youssef Faqir-Rhazoui, Javier Arroyo y Samer Hassan. «A comparative analysis of the platforms for decentralized autonomous organizations in the Ethereum blockchain». En: *Journal of Internet Services and Applications* 12.1 (1 de oct. de 2021). 43 citations (CrossRef 2024/5/19), pág. 9. ISSN: 1869-0238. DOI: 10.1186/s13174-021-00139-6. URL: https://doi.org/10.1186/s13174-021-00139-6 (visitado 25-04-2024).

[15] DeepDAO Ventures Ltd. *DeepDAO*. DeepDAO - Discovery Engine for the DAO Ecosystem. 2023. URL: https://deepdao.io (visitado 20-12-2023).

[16] Vitalik Buterin. *Governance, Part 2: Plutocracy Is Still Bad*. 28 de mar. de 2018. URL: https://vitalik.ca/general/2018/03/28/plutocracy.html (visitado 04-07-2023).

[17] Vitalik Buterin. *Moving beyond coin voting governance*. 16 de ago. de 2021. URL: https://vitalik.ca/general/2021/08/16/voting3.html (visitado 04-07-2023).

[18] Yixuan Fan et al. «Insight into Voting in DAOs: Conceptual Analysis and A Proposal for Evaluation Framework». En: *IEEE Network* (2023). 2 citations (CrossRef 2024/5/19), págs. 1-8. ISSN: 1558-156X. DOI: 10.1109/MNET.137.2200561.

[19] Youssef Faqir-Rhazoui et al. «Effect of the Gas Price Surges on User Activity in the DAOs of the Ethereum Blockchain». En: *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI EA '21. 15 citations (CrossRef 2024/5/19). New York, NY, USA: Association for Computing Machinery, 8 de mayo de 2021, págs. 1-7. ISBN: 978-1-4503-8095-9. DOI: 10.1145/3411763.3451755. URL: https://doi.org/10.1145/3411763.3451755 (visitado 19-06-2023).

[20] Andrea Peña-Calvin et al. *Census of the Ecosystem of Decentralized Autonomous Organizations*. Ver. 1.0.0. 11 de mar. de 2024. DOI: 10.5281/ZENODO.10794915. URL: https://zenodo.org/doi/10.5281/zenodo.10794915 (visitado 11-04-2024).

[21] Dan Cosley et al. «SuggestBot: using intelligent task routing to help people find work in wikipedia». En: *Proceedings of the 12th international conference on Intelligent user interfaces*. IUI '07. 153 citations (CrossRef 2024/5/19). New York, NY, USA: Association for Computing Machinery, 2007, págs. 32-41.

ISBN: 978-1-59593-481-9. DOI: 10.1145/1216295.1216309. URL: https://dl.acm.org/doi/10.1145/1216295.1216309 (visitado 31-01-2024).

[22] Ellery Wulczyn et al. «Growing Wikipedia Across Languages via Recommendation». En: *Proceedings of the 25th International Conference on World Wide Web*. WWW '16. 24 citations (CrossRef 2024/5/19). Republic y Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2016, págs. 975-985. ISBN: 978-1-4503-4143-1. DOI: 10.1145/2872427.2883077. URL: https://dl.acm.org/doi/10.1145/2872427.2883077 (visitado 12-07-2023).

[23] Oleksii Moskalenko, Denis Parra y Diego Saez-Trumper. *Scalable Recommendation of Wikipedia Articles to Editors Using Representation Learning*. 24 de sep. de 2020. DOI: 10.48550/arXiv.2009.11771. URL: http://arxiv.org/abs/2009.11771 (visitado 12-07-2023).

[24] Davor Cubranic y Gail C Murphy. «Automatic bug triage using text categorization». En: *Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering*. Software Engineering and Knowledge Engineering. Banff, Alberta, Canada: Frank Maurer y Günther Ruhe, 2004, págs. 92-97. ISBN: 1-891706-14-4.

[25] Huoliang He y ShunKun Yang. «Automatic Bug Triage Using Hierarchical Attention Networks». En: *2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. 2021 IEEE 21st International Conference on Software Quality, Reliability and Security Companion (QRS-C). 2 citations (CrossRef 2024/5/19) ISSN: 2693-9371. Dic. de 2021, págs. 1043-1049. DOI: 10.1109/QRS-C55045.2021.00158. URL: https://ieeexplore.ieee.org/document/9742064 (visitado 31-01-2024).

[26] Anton Strand et al. «Using a context-aware approach to recommend code reviewers: findings from an industrial case study». En: *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Practice*. ICSE-SEIP '20. 11 citations (CrossRef 2024/5/19). New York, NY, USA: Association for Computing Machinery, 18 de sep. de 2020, págs. 1-10. ISBN: 978-1-4503-7123-0. DOI: 10.1145/3377813.3381365. URL: https://dl.acm.org/doi/10.1145/3377813.3381365 (visitado 01-02-2024).

[27] Aleksandr Chueshev et al. «Expanding the Number of Reviewers in Open-Source Projects by Recommending Appropriate Developers». En: *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME). 5 citations (CrossRef 2024/5/19) ISSN: 2576-3148. Adelaide, SA, Australia: IEEE, sep. de 2020, págs. 499-510. DOI: 10.1109/ICSME46990.2020.00054.

[28] Kattiana Constantino, Fabiano Belém y Eduardo Figueiredo. «Dual analysis for helping developers to find collaborators based on co-changed files: An empirical study». En: *Software: Practice and Experience* 53.6 (2023). 1 citations (CrossRef 2024/5/19), págs. 1438-1464. ISSN: 1097-024X. DOI: 10.1002/spe.3194. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.3194 (visitado 12-07-2023).

[29] Yunyi Xie et al. «Time-Series Snapshot Network for Partner Recommendation: A Case Study on OSS». En: *IEEE Transactions on Computational Social Systems* 9.4 (ago. de 2022). 1 citations (CrossRef 2024/5/19) Conference Name:

IEEE Transactions on Computational Social Systems, págs. 1048-1059. ISSN: 2329-924X. DOI: 10.1109/TCSS.2021.3070914.

[30] Ali Sajedi-Badashian y Eleni Stroulia. «Vocabulary and time based bug-assignment: A recommender system for open-source projects». En: *Software: Practice and Experience* 50.8 (2020). 10 citations (CrossRef 2024/5/19), págs. 1539-1564. ISSN: 1097-024X. DOI: 10.1002/spe.2830. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2830 (visitado 12-07-2023).

[31] Themistoklis Diamantopoulos, Nikolaos Saoulidis y Andreas Symeonidis. «Automated issue assignment using topic modelling on Jira issue tracking data». En: *IET Software* 17.3 (2023), págs. 333-344. ISSN: 1751-8814. DOI: 10.1049/sfw2.12129. URL: https://onlinelibrary.wiley.com/doi/abs/10.1049/sfw2.12129 (visitado 12-07-2023).

[32] Diego Garzia y Stefan Marschall. «Research on Voting Advice Applications: State of the Art and Future Directions». En: *Policy & Internet* 8.4 (1 de dic. de 2016). 20 citations (CrossRef 2024/5/19) Publisher: John Wiley & Sons, Ltd, págs. 376-390. ISSN: 1944-2866. DOI: 10.1002/poi3.140. URL: https://onlinelibrary.wiley.com/doi/10.1002/poi3.140 (visitado 07-02-2024).

[33] Diego Garzia y Stefan Marschall. «Voting Advice Applications». En: *Oxford Research Encyclopedia of Politics*. Oxford University Press, 26 de mar. de 2019. ISBN: 978-0-19-022863-7. DOI: 10.1093/acrefore/9780190228637.013.620. (Visitado 26-02-2024).

[34] Luis Terán y Andreas Meier. «A Fuzzy Recommender System for eElections». En: *Electronic Government and the Information Systems Perspective*. Ed. por Kim Normann Andersen et al. Lecture Notes in Computer Science. 25 citations (CrossRef 2024/5/19). Berlin, Heidelberg: Springer, 2010, págs. 62-76. ISBN: 978-3-642-15172-9. DOI: 10.1007/978-3-642-15172-9_6.

[35] Daniil Buryakov et al. «Using Open Government Data to Facilitate the Design of Voting Advice Applications». En: *Electronic Participation*. Ed. por Robert Krimmer et al. Lecture Notes in Computer Science. 1 citations (CrossRef 2024/5/19). Cham: Springer Nature Switzerland, 2022, págs. 19-34. ISBN: 978-3-031-23213-8. DOI: 10.1007/978-3-031-23213-8_2.

[36] Iván Cantador et al. «Personalized recommendations in e-participation: offline experiments for the 'Decide Madrid' platform». En: *Proceedings of the International Workshop on Recommender Systems for Citizens*. CitRec '17. 15 citations (CrossRef 2024/5/19). New York, NY, USA: Association for Computing Machinery, 2017, págs. 1-6. ISBN: 978-1-4503-5370-0. DOI: 10.1145/3127325.3127330. URL: https://dl.acm.org/doi/10.1145/3127325.3127330 (visitado 11-03-2024).

[37] Iván Cantador et al. «What's going on in my city? recommender systems and electronic participatory budgeting». En: *Proceedings of the 12th ACM Conference on Recommender Systems*. RecSys '18. 6 citations (CrossRef 2024/5/19). New York, NY, USA: Association for Computing Machinery, 27 de sep. de 2018, págs. 219-223. ISBN: 978-1-4503-5901-6. DOI: 10.1145/3240323.3240391. URL: https://dl.acm.org/doi/10.1145/3240323.3240391 (visitado 11-03-2024).

[38] Wei Zhang y Jianyong Wang. «A Collective Bayesian Poisson Factorization Model for Cold-start Local Event Recommendation». En: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mi-*

ning. KDD '15. 72 citations (CrossRef 2024/5/19). New York, NY, USA: Association for Computing Machinery, 2015, págs. 1455-1464. ISBN: 978-1-4503-3664-2. DOI: 10.1145/2783258.2783336. URL: https://dl.acm.org/doi/10.1145/2783258.2783336 (visitado 02-11-2023).

[39] Tuan-Anh Nguyen Pham et al. «A general graph-based model for recommendation in event-based social networks». En: *2015 IEEE 31st International Conference on Data Engineering*. 2015 IEEE 31st International Conference on Data Engineering. 77 citations (CrossRef 2024/5/19) ISSN: 2375-026X. Abr. de 2015, págs. 567-578. DOI: 10.1109/ICDE.2015.7113315. URL: https://ieeexplore.ieee.org/abstract/document/7113315 (visitado 19-04-2024).

[40] Einat Minkov et al. «Collaborative future event recommendation». En: *Proceedings of the 19th ACM international conference on Information and knowledge management*. CIKM '10. 56 citations (CrossRef 2024/5/19). New York, NY, USA: Association for Computing Machinery, 26 de oct. de 2010, págs. 819-828. ISBN: 978-1-4503-0099-5. DOI: 10.1145/1871437.1871542. URL: https://doi.org/10.1145/1871437.1871542 (visitado 22-04-2024).

[41] Augusto Q. Macedo, Leandro B. Marinho y Rodrygo L.T. Santos. «Context-Aware Event Recommendation in Event-based Social Networks». En: *Proceedings of the 9th ACM Conference on Recommender Systems*. RecSys '15. 134 citations (CrossRef 2024/5/19). New York, NY, USA: Association for Computing Machinery, 16 de sep. de 2015, págs. 123-130. ISBN: 978-1-4503-3692-5. DOI: 10.1145/2792838.2800187. URL: https://dl.acm.org/doi/10.1145/2792838.2800187 (visitado 30-10-2023).

[42] Dalila Ressi et al. «AI-enhanced blockchain technology: A review of advancements and opportunities». En: *Journal of Network and Computer Applications* (1 de mar. de 2024), pág. 103858. ISSN: 1084-8045. DOI: 10.1016/j.jnca.2024.103858. URL: https://www.sciencedirect.com/science/article/pii/S1084804524000353 (visitado 04-03-2024).

[43] Maciej Kula. *Spotlight*. 2017. URL: https://github.com/maciejkula/spotlight.

[44] Adam Paszke et al. «PyTorch: An Imperative Style, High-Performance Deep Learning Library». En: (3 de dic. de 2019). 222 citations (INSPIRE 2024/5/19) 221 citations w/o self (INSPIRE 2024/5/19). DOI: 10.48550/arXiv.1912.01703. arXiv: 1912.01703[cs,stat]. URL: http://arxiv.org/abs/1912.01703 (visitado 04-03-2024).

[45] Nicolas Hug. «Surprise: A Python library for recommender systems». En: *Journal of Open Source Software* 5.52 (5 de ago. de 2020). 114 citations (CrossRef 2024/5/19), pág. 2174. ISSN: 2475-9066. DOI: 10.21105/joss.02174. URL: https://joss.theoj.org/papers/10.21105/joss.02174 (visitado 04-03-2024).

[46] Pauli Virtanen et al. «SciPy 1.0: fundamental algorithms for scientific computing in Python». En: *Nature Methods* 17.3 (2 de mar. de 2020). 3401 citations (INSPIRE 2024/5/19) 3344 citations w/o self (INSPIRE 2024/5/19), págs. 261-272. ISSN: 1548-7091, 1548-7105. DOI: 10.1038/s41592-019-0686-2. URL: https://www.nature.com/articles/s41592-019-0686-2 (visitado 04-03-2024).

[47] Ben Frederickson. *Implicit: Fast Python Collaborative Filtering for Implicit Feedback Datasets*. 2018. URL: https://github.com/benfred/implicit.

[48] Marcel Caraciolo et al. *Crab: A Recommendation Engine Framework for Python*. Austin, Texas, 2010. URL: https://github.com/muricoca/crab.

[49] Matthias Fey y Jan Eric Lenssen. «Fast Graph Representation Learning with PyTorch Geometric». En: (25 de abr. de 2019). 42 citations (INSPIRE 2024/5/19) 42 citations w/o self (INSPIRE 2024/5/19). DOI: 10.48550/arXiv.1903.02428. arXiv: 1903.02428[cs,stat]. URL: http://arxiv.org/abs/1903.02428 (visitado 06-02-2024).

[50] Andreas Argyriou, Miguel González-Fierro y Le Zhang. «Microsoft Recommenders: Best Practices for Production-Ready Recommendation Systems». En: *Companion Proceedings of the Web Conference 2020*. WWW '20. 12 citations (CrossRef 2024/5/19). New York, NY, USA: Association for Computing Machinery, 2020, págs. 50-51. ISBN: 978-1-4503-7024-0. DOI: 10.1145/3366424.3382692. URL: https://dl.acm.org/doi/10.1145/3366424.3382692 (visitado 05-12-2023).

[51] massquantity. *LibRecommender*. Jul. de 2020. URL: https://github.com/massquantity/LibRecommender.

[52] Charu C. Aggarwal. *Recommender Systems*. Cham: Springer International Publishing, 2016. ISBN: 978-3-319-29659-3. DOI: 10.1007/978-3-319-29659-3. URL: http://link.springer.com/10.1007/978-3-319-29659-3 (visitado 04-09-2023).

[53] Gawesh Jawaheer, Martin Szomszor y Patty Kostkova. «Comparison of implicit and explicit feedback from an online music recommendation service». En: *Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems*. HetRec '10. New York, NY, USA: Association for Computing Machinery, 26 de sep. de 2010, págs. 47-51. ISBN: 978-1-4503-0407-8. DOI: 10.1145/1869446.1869453. URL: https://dl.acm.org/doi/10.1145/1869446.1869453 (visitado 19-05-2024).

[54] Yehuda Koren, Robert Bell y Chris Volinsky. «Matrix Factorization Techniques for Recommender Systems». En: *Computer* 42.8 (ago. de 2009). 7038 citations (CrossRef 2024/5/19) Conference Name: Computer, págs. 30-37. ISSN: 1558-0814. DOI: 10.1109/MC.2009.263. URL: https://ieeexplore.ieee.org/document/5197422 (visitado 15-05-2024).

[55] Simon Funk. *Netflix Update: Try This at Home*. 11 de dic. de 2006. URL: https://sifter.org/~simon/journal/20061211.html (visitado 16-05-2024).

[56] Yehuda Koren. «Factor in the neighbors: Scalable and accurate collaborative filtering». En: *ACM Transactions on Knowledge Discovery from Data* 4.1 (2010). 494 citations (CrossRef 2024/5/19), 1:1-1:24. ISSN: 1556-4681. DOI: 10.1145/1644873.1644874. URL: https://doi.org/10.1145/1644873.1644874 (visitado 16-05-2024).

[57] Xiangnan He et al. «Neural Collaborative Filtering». En: *Proceedings of the 26th International Conference on World Wide Web*. WWW '17. 3384 citations (CrossRef 2024/5/19). Republic y Canton of Geneva, CHE: International World Wide Web Conferences Steering Committee, 2017, págs. 173-182. ISBN: 978-1-4503-4913-0. DOI: 10.1145/3038912.3052569. URL: https://doi.org/10.1145/3038912.3052569 (visitado 16-05-2024).

[58] Shiwen Wu et al. «Graph Neural Networks in Recommender Systems: A Survey». En: *ACM Computing Surveys* 55.5 (2022). 239 citations (CrossRef 2024/5/19), 97:1-97:37. ISSN: 0360-0300. DOI: 10.1145/3535101. URL: https://dl.acm.org/doi/10.1145/3535101 (visitado 16-05-2024).

[59] Shoujin Wang et al. *Graph Learning Approaches to Recommender Systems: A Review*. 22 de abr. de 2020. DOI: 10.48550/arXiv.2004.11718. arXiv: 2004.11718[cs]. URL: http://arxiv.org/abs/2004.11718 (visitado 16-05-2024).

[60] Chen Gao et al. «A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions». En: *ACM Transactions on Recommender Systems* 1.1 (3 de mar. de 2023). 103 citations (CrossRef 2024/5/19), 3:1-3:51. DOI: 10.1145/3568022. URL: https://dl.acm.org/doi/10.1145/3568022 (visitado 17-07-2023).

[61] Quinn DuPont. *New Online Communities: Graph Deep Learning on Anonymous Voting Networks to Identify Sybils in Polycentric Governance*. 2 de feb. de 2024. URL: http://arxiv.org/abs/2311.17929 (visitado 10-02-2024).

[62] Bryan Perozzi, Rami Al-Rfou y Steven Skiena. «DeepWalk: online learning of social representations». En: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. KDD '14. 5637 citations (CrossRef 2024/5/19). New York, NY, USA: Association for Computing Machinery, 2014, págs. 701-710. ISBN: 978-1-4503-2956-9. DOI: 10.1145/2623330.2623732. URL: https://doi.org/10.1145/2623330.2623732 (visitado 16-05-2024).

[63] David K Duvenaud et al. «Convolutional Networks on Graphs for Learning Molecular Fingerprints». En: *Advances in Neural Information Processing Systems*. Vol. 28. Curran Associates, Inc., 2015. URL: https://papers.nips.cc/paper_files/paper/2015/hash/f9be311e65d81a9ad8150a60844bb94c-Abstract.html (visitado 16-05-2024).

[64] Will Hamilton, Zhitao Ying y Jure Leskovec. «Inductive Representation Learning on Large Graphs». En: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017 (visitado 16-05-2024).

[65] Xiang Wang et al. «Neural Graph Collaborative Filtering». En: *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR'19. 1657 citations (CrossRef 2024/5/19). New York, NY, USA: Association for Computing Machinery, 18 de jul. de 2019, págs. 165-174. ISBN: 978-1-4503-6172-9. DOI: 10.1145/3331184.3331267. URL: https://dl.acm.org/doi/10.1145/3331184.3331267 (visitado 23-04-2024).

[66] Xiangnan He et al. «LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation». En: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '20. 1532 citations (CrossRef 2024/5/19). New York, NY, USA: Association for Computing Machinery, 25 de jul. de 2020, págs. 639-648. ISBN: 978-1-4503-8016-4. DOI: 10.1145/3397271.3401063. URL: https://doi.org/10.1145/3397271.3401063 (visitado 15-05-2024).

[67] Steffen Rendle et al. *BPR: Bayesian Personalized Ranking from Implicit Feedback*. 9 de mayo de 2012. DOI: 10.48550/arXiv.1205.2618. URL: http://arxiv.org/abs/1205.2618 (visitado 27-07-2023).

[68] David Davó. *daviddavo/upm-tfm-notebooks: Notebooks to explore the creation of a RecSys for DAOs*. 2024. URL: https://github.com/daviddavo/upm-tfm-notebooks (visitado 01-02-2024).

[69] Steffen Rendle. «Factorization Machines». En: *2010 IEEE International Conference on Data Mining*. 2010 IEEE International Conference on Data Mi-

ning. 1669 citations (CrossRef 2024/5/19) ISSN: 2374-8486. Dic. de 2010, págs. 995-1000. DOI: 10.1109/ICDM.2010.127. URL: https://ieeexplore.ieee.org/document/5694074 (visitado 16-05-2024).

[70] Yuefeng Zhang. *An Introduction to Matrix factorization and Factorization Machines in Recommendation System, and Beyond*. 12 de mar. de 2022. DOI: 10.48550/arXiv.2203.11026. arXiv: 2203.11026[cs]. URL: http://arxiv.org/abs/2203.11026 (visitado 16-05-2024).

[71] Douglas W Oard y Jinmook Kim. «Implicit Feedback for Recommender Systems». En: *Proceedings of the AAAI workshop on recommender systems*. AAAI workshop on recommender systems. Vol. 83. University of Maryland, College Park, MD 20742: College of Library e Information Service, 1998, págs. 81-83.

[72] Emanuele Rossi et al. *Temporal Graph Networks for Deep Learning on Dynamic Graphs*. 9 de oct. de 2020. URL: http://arxiv.org/abs/2006.10637 (visitado 13-02-2024).

[73] Javier Arroyo, David Davó y Youssef Faqir-Rhazoui. *DAO Analyzer dataset*. Ver. 2024-01-17. 17 de ene. de 2024. DOI: 10.5281/ZENODO.7669709. URL: https://zenodo.org/doi/10.5281/zenodo.7669709 (visitado 17-01-2024).

[74] DappRadar. *Decentraland - Project Overview, Analytics, and Data*. DappRadar. 2024. URL: https://dappradar.com/dapp/decentraland (visitado 04-04-2024).

[75] Stan Higgins. *$26 Million: Blockchain VR Project Decentraland Raises New Funding in ICO*. CoinDesk. Section: Markets. 18 de ago. de 2017. URL: https://www.coindesk.com/markets/2017/08/18/26-million-blockchain-vr-project-decentraland-raises-new-funding-in-ico/ (visitado 15-02-2024).

[76] Victor Tangermann. *$1.2 Billion Metaverse Horrified by Report It Only Had 38 Active Users*. The Byte. 10 de dic. de 2022. URL: https://futurism.com/the-byte/metaverse-decentraland-report-active-users (visitado 15-02-2024).

[77] Decentraland. *Decentraland Places Overview*. 2024. URL: https://decentraland.org/places/ (visitado 15-02-2024).

[78] Ewa Maslowska, Bas van den Putte y Edith G. Smit. «The Effectiveness of Personalized E-mail Newsletters and the Role of Personal Characteristics». En: *Cyberpsychology, Behavior, and Social Networking* 14.12 (dic. de 2011). 24 citations (CrossRef 2024/5/19) Publisher: Mary Ann Liebert, Inc., publishers, págs. 765-770. ISSN: 2152-2715. DOI: 10.1089/cyber.2011.0050. URL: https://www.liebertpub.com/doi/abs/10.1089/cyber.2011.0050 (visitado 16-11-2023).

[79] Yitong Ji et al. «A Critical Study on Data Leakage in Recommender System Offline Evaluation». En: *ACM Transactions on Information Systems* 41.3 (31 de jul. de 2023). 6 citations (CrossRef 2024/5/19), págs. 1-27. ISSN: 1046-8188, 1558-2868. DOI: 10.1145/3569930. URL: https://dl.acm.org/doi/10.1145/3569930 (visitado 29-04-2024).

[80] F. Pedregosa et al. «Scikit-learn: Machine Learning in Python». En: *Journal of Machine Learning Research* 12 (2011), págs. 2825-2830.

[81] Leonard J. Tashman. «Out-of-sample tests of forecasting accuracy: an analysis and review». En: *International Journal of Forecasting*. The M3- Competition 16.4 (1 de oct. de 2000). 512 citations (CrossRef 2024/5/19), págs. 437-450.

ISSN: 0169-2070. DOI: 10.1016/S0169-2070(00)00065-0. URL: https://www.sciencedirect.com/science/article/pii/S0169207000000650 (visitado 21-03-2024).

[82] Jonathan L. Herlocker et al. «Evaluating collaborative filtering recommender systems». En: *ACM Transactions on Information Systems* 22.1 (2004). 3835 citations (CrossRef 2024/5/19), págs. 5-53. ISSN: 1046-8188. DOI: 10.1145/963770.963772. URL: https://dl.acm.org/doi/10.1145/963770.963772 (visitado 23-02-2024).

[83] Christopher D. Manning, Prabhakar Raghavan e Hinrich Schütze. «Chapter 8. Evaluation in information retrieval». En: *Introduction to information retrieval*. OCLC: ocn190786122. New York: Cambridge University Press, 2008, págs. 151-175. ISBN: 978-0-521-86571-5.

[84] Steffen Rendle, Li Zhang y Yehuda Koren. *On the Difficulty of Evaluating Baselines: A Study on Recommender Systems*. 3 de mayo de 2019. DOI: 10.48550/arXiv.1905.01395. URL: http://arxiv.org/abs/1905.01395 (visitado 25-10-2023).

[85] Yitong Ji et al. «A Re-visit of the Popularity Baseline in Recommender Systems». En: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '20. 20 citations (CrossRef 2024/5/19). New York, NY, USA: Association for Computing Machinery, 25 de jul. de 2020, págs. 1749-1752. ISBN: 978-1-4503-8016-4. DOI: 10.1145/3397271.3401233. URL: https://dl.acm.org/doi/10.1145/3397271.3401233 (visitado 21-02-2024).

[86] Andrew Schwartz y David Davó Laviña. *DAOs Census TFM*. 2023. DOI: 10.34740/kaggle/dsv/8413149. URL: https://www.kaggle.com/dsv/8413149 (visitado 19-01-2024).

[87] Snapshot. *Snapshot docs*. Welcome to Snapshot docs! URL: https://docs.snapshot.org/ (visitado 01-02-2024).

[88] Mark Raasveldt y Hannes Muehleisen. *DuckDB*. Ver. 0.9.2. 2023. URL: https://github.com/duckdb/duckdb (visitado 06-02-2024).

[89] Wes McKinney. «Data Structures for Statistical Computing in Python». En: Python in Science Conference. 198 citations (INSPIRE 2024/5/19) 198 citations w/o self (INSPIRE 2024/5/19). Austin, Texas, 2010, págs. 56-61. DOI: 10.25080/Majora-92bf1922-00a. URL: https://conference.scipy.org/proceedings/scipy2010/mckinney.html (visitado 06-02-2024).

[90] Nils Reimers e Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 27 de ago. de 2019. DOI: 10.48550/arXiv.1908.10084. arXiv: 1908.10084[cs]. URL: http://arxiv.org/abs/1908.10084 (visitado 08-02-2024).

[91] Martin Hirzel, Scott Schneider y Kanat Tangwongsan. «Sliding-Window Aggregation Algorithms: Tutorial». En: *Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*. DEBS '17. 16 citations (CrossRef 2024/5/19). New York, NY, USA: Association for Computing Machinery, 8 de jun. de 2017, págs. 11-14. ISBN: 978-1-4503-5065-5. DOI: 10.1145/3093742.3095107. URL: https://dl.acm.org/doi/10.1145/3093742.3095107 (visitado 25-01-2024).

[92] Christian Bird et al. «Latent social structure in open source projects». En: *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of software engineering*. SIGSOFT '08/FSE-16. 182 citations (CrossRef

2024/5/19). New York, NY, USA: Association for Computing Machinery, 9 de nov. de 2008, págs. 24-35. ISBN: 978-1-59593-995-1. DOI: 10.1145/1453101.1453107. URL: https://dl.acm.org/doi/10.1145/1453101.1453107 (visitado 01-02-2024).

[93] Richard Liaw et al. «Tune: A Research Platform for Distributed Model Selection and Training». En: (13 de jul. de 2018). 31 citations (INSPIRE 2024/5/19) 31 citations w/o self (INSPIRE 2024/5/19). DOI: 10.48550/arXiv.1807.05118. URL: http://arxiv.org/abs/1807.05118 (visitado 29-01-2024).

[94] James Bergstra, Daniel Yamins y David Cox. «Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures». En: *Proceedings of the 30th International Conference on Machine Learning*. International Conference on Machine Learning. ISSN: 1938-7228. PMLR, 13 de feb. de 2013, págs. 115-123. URL: https://proceedings.mlr.press/v28/bergstra13.html (visitado 30-04-2024).

[95] Intel. *Procesador Intel® Core™ i9-12900KS (caché de 30 MB, hasta 5,50 GHz) - Especificaciones de productos*. Intel. 2022. URL: https://www.intel.la/content/www/xl/es/products/sku/225916/intel-core-i912900ks-processor-30m-cache-up-to-5-50-ghz/specifications.html (visitado 02-02-2024).

[96] *Anexo I. RESULTADOS DEL ETIQUETADO DE ELECTRICIDAD DE LAS EMPRESAS COMERCIALIZADORAS RELATIVOS A LA ENERGÍA PRODUCIDA EN EL AÑO 2022*. GDO/DE/001/23. Comisión Nacional de los Mercados y la Competencia, 2023. URL: https://gdo.cnmc.es/CNMC/accesoEtiquetado.do.

[97] Vicerrectorado de Tecnología y Sostenibilidad. *Informe Huella de Carbono UCM*. Madrid: Universidad Complutense de Madrid, 2021. URL: https://www.ucm.es/sostenibilidad/huellaucm.

[98] Paolo Cremonesi, Yehuda Koren y Roberto Turrin. «Performance of recommender algorithms on top-n recommendation tasks». En: *Proceedings of the fourth ACM conference on Recommender systems*. RecSys '10. 851 citations (CrossRef 2024/5/19). New York, NY, USA: Association for Computing Machinery, 26 de sep. de 2010, págs. 39-46. ISBN: 978-1-60558-906-0. DOI: 10.1145/1864708.1864721. URL: https://dl.acm.org/doi/10.1145/1864708.1864721 (visitado 25-02-2024).

[99] David Davó. *Ray results of «Exploración de Sistemas Recomendadores para la Recomendación de Propuestas en Organizaciones Autónomas Descentralizadas»*. 26 de abr. de 2024. DOI: 10.5281/zenodo.11072578. URL: https://zenodo.org/doi/10.5281/zenodo.11072578 (visitado 15-05-2024).

[100] David Davó y Javier Arroyo. «Enhancing Voter Engagement in Decentralized Governance: A Recommender System for DAOs using Graph Neural Networks». Submitted. Submitted. 2024.

[101] Jianxun Lian et al. «xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems». En: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 601 citations (CrossRef 2024/5/19). 19 de jul. de 2018, págs. 1754-1763. DOI: 10.1145/3219819.3220023. arXiv: 1803.05170[cs]. URL: http://arxiv.org/abs/1803.05170 (visitado 15-11-2023).

[102] Vito Walter Anelli et al. «Local Popularity and Time in top-N Recommendation». En: *Advances in Information Retrieval*. European Conference on Information Retrieval. Ed. por Leif Azzopardi et al. Vol. 11437. Lecture Notes in Com-

puter Science. Cham: Springer International Publishing, 7 de abr. de 2019, págs. 861-868. ISBN: 978-3-030-15711-1. DOI: 10.1007/978-3-030-15712-8_63. URL: http://link.springer.com/10.1007/978-3-030-15712-8_63 (visitado 25-02-2024).

[103] Zelong Li et al. *PAP-REC: Personalized Automatic Prompt for Recommendation Language Model*. version: 1. 31 de ene. de 2024. DOI: 10.48550/arXiv.2402.00284. arXiv: 2402.00284[cs]. URL: http://arxiv.org/abs/2402.00284 (visitado 13-02-2024).

[104] Rafael Brundo Uriarte y Rocco DeNicola. «Blockchain-Based Decentralized Cloud/Fog Solutions: Challenges, Opportunities, and Standards». En: *IEEE Communications Standards Magazine* 2.3 (sep. de 2018). 60 citations (CrossRef 2024/5/19) Conference Name: IEEE Communications Standards Magazine, págs. 22-28. ISSN: 2471-2833. DOI: 10.1109/MCOMSTD.2018.1800020. URL: https://ieeexplore.ieee.org/abstract/document/8515145 (visitado 06-02-2024).

Siglas

BERT *Bidirectional Encoder Representations from Transformers* (Representación de Codificador Bidireccional de Transformers). 37

BPR *Bayesian Personalized Ranking*. 12, 41

CF *Collaborative Filtering* (Filtrado Colaborativo). 10, 20, 35

CSV *Comma-separated values* (Valores Separados por Comas). 34

CTDG *Continuous-time dynamic graphs*. 16

DAO *Decentralized Autonomous Organization* (Organización Autónoma Descentralizada). 1, 2, 4, 7, 15–18, 21, 33, 35, 36, 40, 42, 49

dApp *Decentralized Application* (Aplicación Descentralizada). 33, 34, 40

DCG *Discounted Cumulative Gain* (Ganacia Acumulativa Descontada). 28

DeFi *Decentralized Finance* (Finanzas Descentralizadas). 4

DTDG *Discrete-time dynamic graphs* (Grafos dinámicos de tiempo discreto). 16

EBSN *Event-Based Social Networks* (Redes Sociales Basadas en Eventos). 8

EVM *Ethereum Virtual Machine* (Máquina Virtual de Ethereum). 4, 5

FM *Factorization Machine*. 13

GCN *Graph Convolutional Network*. 7, 11

GdO Garantías de Origen renovable. 42

GNN *Graph Neural Network*. 7–11, 20, 39, 42, 43

IC Intervalo de Confianza. 37

ICO *Initial Coin Offering* (Oferta inicial de criptomonedas). 18

IPFS *InterPlanetary File System*. 5

ItemCF *item-based collaborative filtering*. 11

kNN *k-nearest neighbors* (k vecinos más cercanos). ix, 13, 37–40, 43

LGC *Light Graph Convolution*. 12

LLM *Large Language Model* (Gran Modelo de Lenguaje). 54

MAP *Mean Average Precission* (Precisión Media Promedio). 23, 27, 28, 39, 41, 42, 45

MF *Matrix Factorization* (Factorización de Matrices). 11–13

NCF *Neural Collaborative Filtering*. 11

nDCG *Normalized Discounted Cumulative Gain* (Ganacia Acumulativa Descontada Normalizada). 9, 23, 28, 31, 41, 42, 45

NFT *Non-Fungible Token* (Token no fungible). 18

NGCF *Neural Graph Collaborative Filtering*. 11, 12

PLN Procesamiento del Lenguaje Natural. 20, 36, 39, 43, 45, 46, 50

SR Sistema Recomendador. 20, 28, 29, 39, 40, 43, 45

SVD *Single Value Decomposition*. 11

SVM *Support Vector Machine*. 13

UserCF *user-based collaborative filtering*. 11

VP *voting power* (poder de votación). 5

VPP Votos por Propuesta. 17, 18

VPV Votos por Votante. 18, 26

Apéndice A

Características del servidor dedicado

Todas las ejecuciones se han realizado en un único servidor dedicado asegurándose de que no había ninguna otra tarea intensiva ejecutándose en el momento. Las epecificaciones de dicho servidor, proporcionado por el Instituto de Tecnología del Conocimiento de la Universidad Complutense de Madrid y administrado por el Departamento de Arquitectura de Computadores y Automática, son las siguientes:

OS Ubuntu 22.04 LTS

Kernel Linux 5.15

Placa base MPG Z690

CPU 12th Gen Intel i9-12900KS (24) @ 5.200GHz

RAM 128 GB

GPU NVIDIA AD102 (GeForce RTX 4090) 24 GiB VRAM

A este equipo se ha conectado en remoto mediante SSH para ejecutar Jupyter Notebooks usando Jupyter Lab. Se ha utilizado Python 3.9 debido a la dependencia de la librería de Microsoft Recommenders.