Fossr

Free and Open-Source Software Recommender

“Recommendation is an indirect way of bringing programmers together”

Integrantes

- Alan Pierri Legajo: 50112
- Teresa di Tada Legajo: 52354
- Valeria Serber Legajo: 51021

Índice

Abstract

De una manera similar a la recomendación de YouTube para videos o de Spotify para canciones, se busca un sistema de sugerencia de proyectos Open-Source para un usuario de algún sistema de repositorios Git. En particular, analizamos GitHub que, a pesar de promover la comunidad abierta, no cuenta con ningún sistema de recomendaciones. BitBucket no se contempló ya que apunta más a los repositorios privados.

Para encontrar proyectos de interés tanto para utilizar como para aportar a los mismos, los programadores no poseen ninguna herramienta de recomendación personalizada. Github permite explorar sus repositorios agrupándolos en temas, integraciones y popularidad[1] pero no realiza recomendaciones personalizadas como Netflix o Spotify, por lo tanto rara vez los programadores utilizan esta forma de búsqueda. Basándonos en este problema y viendo una posibilidad de crecimiento en el área pensamos en suplir esta necesidad realizando recomendaciones a partir de la plataforma de GitHub donde se encuentran gran cantidad de proyectos públicos de todo el mundo.

Para recomendaciones personalizadas en las plataformas más populares lo más utilizado es filtros colaborativos que representa una idea y no una implementación particular. La premisa de un sistema de Filtros Colaborativos es que dos usuarios con gustos similares probablemente les interese compartir sus gustos entre sí. Para esto, se cuenta con los siguientes datos de los usuarios:

- Repositorios con los que se contribuyen
- Stars
- Seguidores
- Seguidos
- Contribuyentes de repositorios propios
- Cualquier otro dato accesible públicamente

Los mismos se pueden utilizar para realizar recomendaciones colaborativas. Un escenario esperado sería el siguiente:

Usuario A: Me gustó colaborar con los repositorios P, Q, R and S.
Usuario B: Ah! A mi me gustó colaborar con los repositorios Q, R, S and T.

[1] Encontrando proyectos Open-Source en Github https://help.github.com/articles/finding-open-source-projects-on-github/

Usuario A: Entonces deberías ver si te gusta el repositorio P!
Usuario B: Gracias! Por cierto, yo te recomiendo el repositorio T.

Para realizar estas recomendaciones se implementaron tres algoritmos que recorren un grafo social. El mismo está compuesto por nodos usuario y nodos repositorio y las conexiones entre ellos pueden ser de tipo colaboraciones, bifurcaciones y estrellas. Las colaboraciones pueden ser tanto en repositorios propios del usuario como de otros usuarios. Las bifurcaciones significan una intención de contribuir con ese proyecto, y las estrellas significan que el usuario está interesado en darle un seguimiento a ese repositorio, ya sea para usarlo o simplemente porque le parece interesante. Utilizando el grafo mencionado se implementó un modelo de filtros colaborativos basado en memoria, es decir, que necesita la totalidad del grafo donde se encuentran los datos para realizar las recomendaciones.

En las siguientes secciones de la documentación se comenzará explicando qué es un sistema de recomendación y qué tipos existen. Pasaremos luego a la explicación de filtros colaborativos junto a sus ventajas y desventajas frente a otros modelos. Luego entraremos en un nivel más bajo describiendo las implementaciones y comparando la propia con otras existentes. Presentaremos los datos que se encontraban disponibles y cuales decidimos utilizar. Una vez que se planteó el escenario se analizarán los tres algoritmos implementados con sus ventajas y desventajas, las encuestas realizadas para compararlos y el análisis de las mismas. Finalmente se llegará a una conclusión sobre el trabajo realizado y las soluciones implementadas.

Sistemas de recomendación

Los sistemas de recomendación tienen como objetivo presentarle a un usuario algún objeto que sea de su interés dentro un dominio específico. Se usan para sistemas donde se tiene una gran cantidad de datos. Son muy utilizados, por ejemplo, en aplicaciones populares como Netflix, Spotify, Amazon, etc.

Estos sistemas utilizan la actividad pasada de los usuarios para recomendarles objetos con los cuales todavía no han interactuado.

Existen distintos enfoques para recomendar. El enfoque **basado en contenido** utiliza las características de los objetos que se quieren recomendar, es decir, los objetos pertenecen a ciertas categorías, y son recomendados en base a la historia pasada del usuario con esa categoría. Generalmente en este enfoque se realizan análisis exhaustivos de los objetos, que dependen del dominio de dicho objeto. Por ejemplo, Spotify realiza un análisis de las ondas sonoras de cada canción. Otro enfoque, llamado **basado en objetos**, no realiza análisis de los mismos, sino que utiliza las relaciones entre ellos para establecer una similitud y formar patrones de recomendación. Por otro lado, el enfoque **basado en usuarios** utiliza la similitud entre los perfiles de los distintos usuarios para recomendar objetos que estén relacionados con ellos.

Los distintos objetos se pueden combinar ya que se complementan en la información que cubren para inferir repositorios candidatos. La elección del tipo de información a utilizar depende del dominio del problema que se esté analizando. Por ejemplo, si el análisis comparativo entre el contenido de los objetos es muy demandante y termina no aportando relaciones de cantidad, es preferible no considerar el enfoque basado en contenido.

Filtros colaborativos

Muchas veces cuando se busca elegir qué libro leer o qué serie ver, una persona consulta a sus amigos o conocidos. Si muchos de ellos tienen o escucharon buenas opiniones al respecto de algún objeto en particular, la persona lo considera mejor que otro del cual tenga menos recomendaciones. A su vez sabemos que si alguien ya ha coincidido en el pasado con nosotros en gran cantidad de películas, es más probable que nos recomiende películas nos gusten. La idea de los filtros colaborativos es automatizar estos métodos que se utilizan desde hace mucho tiempo teniendo acceso a una cantidad mucho mayor de datos.

Los filtros colaborativos se utilizan para filtrar información utilizando las opiniones de otras personas o la similitud relacional de los objetos del modelo, de esta manera los usuarios colaboran en el proceso de recomendación indirectamente.

Presentan un poderoso método que permite filtrar sobre la gran cantidad de información disponible en internet que posee reacciones de usuarios y ahorra a las personas gran cantidad de tiempo de búsqueda.

Los filtros colaborativos utilizan una mezcla de las recomendaciones basadas en usuarios y las basadas en objetos de manera de aprovechar tanto la similitud de las personas como la de los objetos.

Los algoritmos que implementan este sistema de recomendación pueden dividirse en dos categorías:

- **Basados en memoria:** Utilizan la base de datos completa para generar recomendaciones. Usando distintas técnicas, se busca generar un criterio para relacionar usuarios u objetos, que compartan alguna característica. Los ejemplos de este enfoque son algoritmos basados en vecinos, que intentan calcular la similitud entre usuarios u objetos, para dar recomendaciones en base a los datos que se tienen sobre los objetos similares.
- **Basados en el modelo:** Proveen recomendaciones de objetos desarrollando primero un modelo de opiniones de usuarios. Encaran el proceso de recomendación usando un enfoque probabilístico y computando el valor esperado de la opinión del usuario a partir de sus opiniones anteriores. Para construir el modelo se suelen usar distintos algoritmos de aprendizaje como redes Bayesianas, generación de grupos (clustering) o basadas en reglas.

Desventajas de los Filtros Colaborativos

Escasez

En la práctica, los usuarios pueden haber tenido pocas interacciones con los objetos del modelo. En sistemas con un set muy grande de objetos, los usuarios activos pueden haber interactuado con un 1% de los objetos, por lo tanto, las recomendaciones no tendrán mucha precisión.

Escalabilidad

Los algoritmos que implementan Filtros Colaborativos requieren un costo de cómputo que crece tanto cuando crece el número de usuarios como cuando crece el número de objetos. En el peor caso aumentará exponencialmente el costo de cómputo del algoritmo, por lo tanto es poco escalable.

Cold Start Problem

Nuevos usuarios

Cuando un nuevo usuario se registra a un sistema, al inicio no tiene historia en él, es decir no tiene interacciones con objetos, por lo tanto no se le puede hacer una recomendación personalizada. Tampoco se puede saber quiénes son los usuarios similares a él ni se puede calcular. Esto suele resolverse obligando al usuario a que califique algunos objetos al momento de registrarse en el sitio, o mostrándole al inicio recomendaciones no personalizadas, sino basadas en popularidad. También puede resolverse clasificando a los usuarios según su información demográfica y recomendarle objetos de esa población.

Nueva comunidad

Cuando se inicia un nuevo servicio de recomendaciones basado en filtros colaborativos, al inicio es costoso armar una base de datos que aporte verdadero valor. Al inicio las recomendaciones personalizadas puede que no sean muy precisas, y que mejoren a medida que el sistema gana más usuarios con más relaciones. Esto se soluciona utilizando el servicio primero con una comunidad reducida para que se generen ciertas relaciones y recomendaciones, y luego abrirlo a una comunidad más grande.

Ventajas de los Filtros Colaborativos

Independencia del contenido

Para realizar una recomendación con filtros colaborativos no es necesario analizar el contenido de los objetos del modelo. Esto permite ahorrar tiempo y análisis computacional además de ser útil en los casos donde no es tan útil o sencillo analizar el contenido.

Recomendaciones personalizadas

Las recomendaciones obtenidas varían de acuerdo a quién queramos recomendarle por lo cual tienen en cuenta la diferencia de personalidad y gusto entre las personas

Combinación de recomendaciones basadas en usuarios y en objetos del modelo

Los filtros colaborativos pueden aprovechar tanto la similitud entre usuarios, las conexiones de los objetos o ambas para recomendar, lo cual presenta una mejora frente a las situaciones donde se utiliza un solo tipo de datos.

Implementación de Filtros Colaborativos

No hay una implementación definida de filtros colaborativos ya que los mismos son una idea genérica de inteligencia artificial por lo que se pueden realizar de distintas formas.[2]

En el presente proyecto se decidió implementar filtros colaborativos pero de una manera diferente a la usual. El dominio se presenta como un grafo con nodos de tipo usuario o repositorio y conexiones únicamente entre dos nodos de distinto tipo. Cada nodo posee un identificador, una lista de conectores y un enlace de GItHub (si es un usuario hacia su perfil, si es un repositorio hacia la dirección del mismo). Los nodos usuarios tienen además el nombre y el enlace del avatar que los representa mientras que los repositorios tienen título, descripción, cantidad de estrellas y cantidad de forks. Por otra parte los conectores tienen un tipo (fork, push, star, que se corresponden con bifurcación del repositorio, contribución al mismo, y darle una estrella al mismo, respectivamente) y apuntan siempre a un nodo usuario y a un nodo repositorio.

En los algoritmos realizados se recorre el grafo de distintas maneras comenzando por el nodo del usuario a recomendarle repositorios y los proyectos a los cuales está conectado y atravesando más nodos de usuarios y repositorios en el camino.

Según cómo se lo mire, se puede considerar que el proceso es de caja blanca o caja negra. Es de caja blanca si se tiene en cuenta que, a diferencia de otros algoritmos de filtros colaborativos, éste explica las relaciones que determinaron la inferencia. Es de caja negra si se tiene en cuenta que nunca se observa el contenido de los objetos analizados, sólo las relaciones entre usuarios y repositorios.

De esta manera se buscó tener recomendaciones de caja blanca donde la información sobre por qué se le recomendó algo a un usuario se conozca.

[2] http://files.grouplens.org/papers/FnT%20CF%20Recsys%20Survey.pdf

Ventajas de nuestra implementación

Recomendaciones creíbles

Los usuarios valoran las recomendaciones que pueden ser explicadas de manera creíbles, por lo que es importante darle una razón a cada recomendación. En nuestro caso, utilizando una implementación con grafos se infiere una red social, permitiendo justificar las recomendaciones por medio de las conexiones del grafo, haciendo al método muy transparente. Existen otras implementaciones, por ejemplo las que utilizan regresiones lineales, en las cuales las recomendaciones salen de una "caja negra" ya que no puede decirse exactamente cómo se llegó a ellas.

Nuevos objetos

En nuestro sistema todo repositorio agregado está relacionado con un usuario (su dueño) por lo que siempre tiene alguna conexión. Esto nos permite una alta probabilidad de que el objeto recién agregado forme parte de la red social. Puede ocurrir en otras implementaciones que al momento de agregar un nuevo objeto al sistema se encuentren con el problema de cold start porque no habrá usuarios que hayan interactuado con él.

En el caso del primer repositorio de un usuario nuevo que todavía no haya interactuado con alguien no se podrían realizar recomendaciones, sin embargo lo solucionamos recomendando proyectos del ranking propio de github (que utiliza las estrellas de los repositorios para calcular probabilidad) aumentando la probabilidad de que el usuario se relacione con alguno de ellos.

Observaciones de comportamiento

Muchos sistemas de recomendación se basan en calificaciones que se les pide a los usuarios al momento de ingresar al sistema, para tener información inicial de sus gustos y poder recomendar en base a eso. Esto genera un comportamiento artificial y no siempre refleja la realidad del usuario. La ventaja de nuestra implementación es que se basa en observaciones del comportamiento del usuario a lo largo del tiempo, incluso antes de la existencia del sistema, entonces se puede inferir realmente sus intereses. Indagar en la historia del usuario hace que podamos independizarnos de las calificaciones artificiales.

Recomendaciones en proyectos de código abierto

Dentro del mundo del código abierto los repositorios que contienen los distintos proyectos se encuentran esparcidos por la web. En general se llega a ellos a través de búsquedas de Google. Pocos programadores aportan a los mismos pero muchos los utilizan en sus propios proyectos ya que poseen licencias abiertas.

Decidimos realizar una aplicación web donde a partir de su nombre en GitHub se le realizan al usuario recomendaciones de repositorios. Para esto se utiliza la API de la plataforma que otorga información útil de los usuarios: repositorios propios, repositorios a los que contribuyó y repositorios que le interesan. Con los datos generamos recomendaciones para que el usuario de nuestra aplicación utilice o contribuya a los proyectos de manera más sencilla y rápida.

Las recomendaciones son una manera indirecta de acercar a los programadores y mejorar el mundo del código abierto.

Datos disponibles para las recomendaciones

En el mundo del código abierto existen distintas interacciones que se pueden tener con un proyecto o usuario. En particular analizaremos las relaciones presentes en la plataforma de Github donde recolectamos datos.

- Interacciones entre usuarios:

  En esta categoría se encuentran la relación de que un usuario siga a otra. Es una relación unilateral al igual que en otras plataformas como Twitter o Instagram a diferencia de la amistad en Facebook donde no se puede ser amigo de alguien sin que éste lo sea de uno.

  A partir de las distintas relaciones se puede inferir la popularidad de un usuario o el interés que un usuario tiene sobre otros que programan en cierto lenguaje o sobre temas específicos.

- Interacciones entre usuario y repositorio:
  1. *Repositorios propios de un usuario:*

     Todos los repositorios se crean con un único dueño a pesar de que muchas personas puedan colaborar. Esta persona tiene una relación distinta con el proyecto ya que puede agregar colaboradores, cambiar la visibilidad del repositorio (pública o privada) y borrarlo[3]. Los permisos que otorga ser dueño no se pueden compartir.
  2. *Repositorios a los que un usuario contribuyó:*

     En esta categoría entran los forks que nos dicen que el usuario bifurcó un proyecto para trabajar sobre él, y los push que identifican a qué proyecto se aportó código.
  3. *Repositorios que un usuario marcó con una estrella[4]:*

     Las estrellas en Github sirven para que el usuario tenga un acceso más sencillo a un determinado repositorio y mostrar aprecio a los usuarios relacionados con el repositorio. Github tiene rankings de repositorios de acuerdo a la cantidad de estrellas que posee para poder hacer búsquedas.

[3] Niveles de permiso de una cuenta en GitHub https://help.github.com/articles/permission-levels-for-a-user-account-repository/

[4] Sobre marcar un repositorio con una estrella en Github: https://help.github.com/articles/about-stars/

Selección y peso de los datos

De las dos categorías de interacciones decidimos darle más importancia a las relaciones entre usuarios y repositorios. Las mismas nos permiten identificar la afinidad y el compromiso de las personas con los distintos proyectos y se representan en nuestro grafo como aristas.

Al momento de elegir los pesos de cada conexión se le da más importancia a las relaciones donde se escribió código en un proyecto (push) luego donde hubo intención de hacerlo (fork) y por último a la expresión de interés (star). Por lo tanto un camino de personas que colaboraron en proyectos va a ser recomendado antes que uno que tiene solo estrellas.

Por otro lado decidimos no utilizar las relaciones entre usuarios. La popularidad de un usuario tiene el problema de que no es un reflejo fiel de la calidad o interés de los repositorios con los que está relacionado. Pensemos en los usuarios sociables que tienen muchos compañeros de trabajo, de facultad o conocidos. Esto no refleja que sus repositorios sean interesantes o valga la pena recomendarlos ya que por ahí no tienen ninguna estrella y nadie mostró interés en ellos.

Algoritmos

Definiciones

Breadth-First Search (Búsqueda en ancho)

Al mapear los datos en un grafo la primera implementación trivial que se realizó fue una búsqueda de repositorios comenzando el recorrido en el usuario al cual recomendar y analizando por profundidad: se comienza con los repositorios directamente conectados, después con los que están un usuario más de distancia y así se van agregando recomendaciones.

El resultado son recomendaciones de repositorios según su cercanía prefiriendo los más cercanos al usuario. Dentro de los repositorios que se encuentran a la misma distancia se agregó una política adicional otorgándole más importancia a las conecciones que consideramos más relevantes frente a las menos relevantes como se explica en la sección Selección y peso de los datos.

Ventajas

- Bajo costo de programación
- Análisis acotable por distancia
- Analiza la calidad de los caminos con misma profundidad

Desventajas

- No analiza la cantidad de caminos hacia el mismo repositorio. El resultado es que una recomendación a la cual muchos usuarios relacionados le pusieron una estrella es igual a una a la cual uno solo lo hizo si están a la misma profundidad.
- Alto costo computacional

Path Analysis Algorithm

Este algoritmo recorre el grafo a partir del usuario y sus repositorios comenzando por los caminos de menor profundidad y aumentando al igual que el BFS. Su nombre (“paths analysis”) lo obtiene ya que tiene en cuenta la cantidad de caminos hacia cada repositorio y su tipo de conexión para realizar las recomendaciones. Es decir, recomienda aquellos repositorios que tengan una mayor cantidad de caminos desde el usuario hacia él, pero ponderando la importancia de las conexiones realizadas por contribuciones por sobre las conexiones hechas por estrellas. La ponderación se realiza dándole un peso a cada arista, y luego la suma de la cantidad de caminos tiene en cuenta el peso de cada arista del camino.

Además se está utilizando la información de la cantidad de estrellas y bifurcaciones que tienen los repositorios al momento de decidir el valor de la recomendación. Si el proyecto suma cinco o menos apreciaciones entre las estrellas y las bifurcaciones se le reduce su valor con una función de tangente hiperbólica.

Ventajas

- Utiliza en su ponderación la cantidad y calidad de los caminos
- Análisis acotable por distancia
- A medida que se aleja del usuario a recomendar cada interacción se desestima más

Desventajas

- Sobreestima repositorios lejanos cuando existen demasiados caminos al mismo
- Alto costo computacional

Flow Analysis Algorithm

Este algoritmo analiza el vínculo entre nodos pensando al grafo social como una red de cañerías de agua. Podemos pensar al nodo inicial (Usuario a recomendar) como una fuente de agua de caudal infinito y cada una de las conexiones limita la cantidad de caudal de agua que puede pasar a través de ellas. El caudal se determina por el tipo de conexión y la lejanía a la fuente. Se establece el caudal máximo entre fuente y sumidero (cualquier repositorio a analizar) cómo la fuerza de conexión entre ellos. Se recomiendan los sumideros con mayor caudal de agua.

También utiliza la cantidad de estrellas y bifurcaciones para darle valor a las recomendaciones al igual que en el “Path Analysis Algorithm”.

Ventajas

- Menor costo computacional
- Los usuarios fuertemente conectados entre sí no generan que cualquier recomendación que pase por ellos tenga importancia excesiva solo por la cantidad de caminos.
- Análisis acotable por conexiones irrelevantes

Desventajas

- Alto costo de programación
- Las conexiones están limitadas por el algoritmo más pesimista entre User-Based e Item-Based

Resultados

Se realizaron encuestas a algunos posibles usuarios del sistema, para tener opiniones de las recomendaciones que les brindamos. En una primer versión de la encuesta obtuvimos opiniones que nos dieron la posibilidad de mejorar el sistema de recomendación.

Las opiniones más populares fueron que los repositorios ya los conocían, que estaban muy relacionados con su trabajo o sus amigos de la facultad. Por otro lado, la red de usuarios estuvo condicionada por el hecho de que muchos de los usuarios que encuestamos son alumnos de la facultad y por lo tanto los repositorios en los que ellos contribuyen son en general trabajos prácticos de la facultad, y no resultan interesantes para otros usuarios.

Otra de las críticas fue que muchos repositorios que se recomendaban estaban vacíos o no eran populares (no tenían ninguna estrella ni ninguna bifurcación) y entonces no resultaban atractivos.

En esta primer versión de la encuesta, además, no se encontraron diferencias significativas entre los tres algoritmos que desarrollamos. Realizando un promedio de las calificaciones obtuvimos casi los mismos resultados para las tres versiones.

Análisis de Resultados

A partir de los resultados anteriores, decidimos incorporar algunas mejoras al sistema. La primera, fue como resultado de la poca variedad de usuarios que tenía nuestro sistema. Lo ideal sería que la red de usuarios crezca y sea muy variada, para que las recomendaciones sean más exactas, y no estén sesgadas por los usuarios que la frecuentan. Por lo tanto, agregamos una mejora para expandir la red. Lo que hicimos fue que cada vez que se agrega un usuario a la red, se agreguen también los usuarios con los que él está relacionado. De esta manera generamos más conexiones nuevas para los usuarios que ya se encuentran en el sistema.

La segunda mejora fue subir la vara de los repositorios que recomendamos, para que resulten interesantes. Esta mejora sólo la implementamos en los algoritmos de Paths y Flow, ya que no lo vimos relevante para BFS pues es un caso muy trivial. Para esto, aplicamos una penalidad inversamente proporcional a la cantidad de estrellas para los proyectos con menos de 5 estrellas. Con esto logramos ponderar correctamente repositorios “fantasmas”, que no tienen contenido y no son populares. Decidimos hacer esto en base a la respuesta que tuvimos de los usuarios, que buscaban algo interesante y popular en vez de algo nuevo y con poca repercusión.

Luego de estos cambios realizamos una segunda tanda de encuestas para obtener nuevos resultados. El nivel de satisfacción general de esta nueva tanda subió en un mínimo porcentaje que no consideramos tan relevante, más que nada por un tema de subjetividad que tiene el hecho de preguntar si un repositorio es de interés o no.

El cambio que pudimos notar es que el sistema comenzó a recomendar repositorios más diversos, que el usuario no esperaba encontrar porque no los conocía. El cambio nos resultó muy positivo debido a que ayudó a eliminar los repositorios fantasmas, y de esta manera pasamos a recomendar repositorios más interesantes y con algo de popularidad.

Extensiones

Hay varias posibles extensiones de este proyecto que pueden aportar mucho más valor a las recomendaciones dadas.

Actualización de recomendaciones

Para actualizar los repositorios relacionados con un usuario en particular, se deberá hacer periódicamente una query a GitHub y agregar las nuevas conexiones al sistema. Para eso se puede implementar un background job que corra cada cierto tiempo y actualice los usuarios. También se puede correr una actualización para un usuario en particular cuando éste ingresa al sitio. Esto es necesario ya que nos da la posibilidad de actualizar las recomendaciones con repositorios nuevos, pero utilizando la misma relación entre usuarios.

Recomendaciones basadas en contenido

Una posible mejora del sistema de recomendación es complementar los filtros colaborativos con las recomendaciones basadas en contenido. Para esto, se deberá analizar la información pública de cada repositorio y clasificarlos en distintos grupos. Por ejemplo, se pueden clasificar por lenguaje predominante del repositorio, por cantidad de contribuyentes, por cantidad de estrellas que poseen, etc. Estos parámetros se utilizarían para dar respuestas que se parezcan aún más a los repositorios en los que está interesado el usuario inicialmente, y por lo tanto se le darían recomendaciones aún más personalizadas. Esta técnica complementa a filtros colaborativos y brinda mayor precisión en las recomendaciones, ya que genera respuestas en base a similitud con usuarios u objetos, pero agregando la posibilidad de personalizar más las características de las respuestas buscadas.

Además, se podría dejar que el usuario elija cuáles de estos parámetros de las distintas clasificaciones valora más y darle recomendaciones en base su valoración.

Recomendaciones de repositorios para utilizar o para contribuir

Durante la selección de los pesos surgió la idea de diferenciar a los usuarios que buscan contribuir a proyectos de los que buscan simplemente repositorios interesantes para utilizar. Se podría consultar junto al nombre de usuario qué es lo que él mismo busca y a partir de allí variar los pesos de las conexiones: el push es lo más relevante si se quiere contribuir en cambio la estrella es lo más importante si se buscan repositorios interesantes para utilizar.

Configuraciones de usuario

Además del usuario ser capaz de elegir si desea recomendaciones para colaborar o para utilizar como se sugirió anteriormente se podría generar una configuración donde el usuario selecciones otras variables como por ejemplo si prefiere ver repositorios de gente cercana a él o repositorios de gente muy popular e interesante.

Recomendaciones de forks

Muchos repositorios parten de la bifurcación de otro pero no agregan nada nuevo. Hoy en día al momento de recomendar este tipo de proyectos no tenemos en cuenta esta situación por lo que se podría mejorar la recomendación listando en su lugar el repositorio original o al menos un enlace al mismo.

Utilizar los timestamp para calcular los pesos en las aristas

Otro avance de los algoritmos podría surgir al incluir las fechas de las relaciones de commit, fork o star en los pesos de las aristas dando más importancia a lo más nuevo y menor importancia a lo más antiguo. Esto permitiría reflejar en las recomendaciones un cambio de gustos de un usuario y no recomendar temas en los cuales ya no muestra interés.

Creación de grupos de desarrollo con tutor

Fomentaría mucho la participación en FOSS si se pudiera generar grupos de gente que tenga interés en un repositorio para que generen un grupo de desarrollo. En nuestra implementación, el usuario podría ponerse en “cola de

espera” para un grupo para un repositorio y el mismo enviar un mail cuando de esa cola de espera se pueda formar un grupo. Incluso se puede determinar (o generar un bot automatizado que proponga) un tutor del grupo, basándose en información de la interacción de los usuarios (por ejemplo, quienes hicieron commit y/o quiénes hicieron más commits en el repositorio.

Alertas de repositorios interesantes

La información de nuestro grafo no es tan dinámica como una red social, pero no es para nada estática. Podemos generar workers sobre el grafo para ir buscando nuevas actualizaciones de los fragmentos del grafo y propagar nuevos flujos posibles. De esa manera, podríamos saber las recomendaciones que cambiaron y, si hay un cambio muy importante, alertar al usuario (por ejemplo, por mail).

Feedback del usuario

Sería ideal que el usuario pueda ir marcando proyectos que ya conoce para ir filtrándolos y complementar los datos obtenidos de las diferentes fuentes. De esta forma se resolvería la recomendación de proyectos muy populares que el usuario ya conoce.

Conclusiones

Un sistema de filtros colaborativos no necesariamente tiene éxito en relacionar contenido al gusto del usuario. A menos que la plataforma alcance una diversidad e independencia de opiniones impecable, siempre existe una corriente de pensamiento que domine el resto en una comunidad específica, como es el caso de la comunidad Open-Source.

No existe técnica de recomendaciones personalizada que pueda tener éxito con usuarios nuevos, ya que no conoce sus preferencias. Lo mismo sucede con nuevos objetos si no se logra analizar el contenido del mismo, a menos que en la naturaleza del objeto exista una relación con un usuario, como en nuestro caso la de un repositorio y su creador. Más aún, los sistemas de Filtros Colaborativos típicamente requieren que una cantidad de usuarios significante interactúen con un objeto para poder recomendarlo. Afortunadamente, contamos con el acceso a todos los datos que requerimos, ya que las interacciones analizadas son públicas.

Para mitigar las dificultades de los sistemas de recomendación, muchos sistemas de recomendaciones (con énfasis en el sector comercial) son híbridos entre un sistema basado en memoria y en el modelo. De esta forma, logran ser más performantes y escapar de las limitaciones de los filtros colaborativos ordinarios como los datos dispersos y la pérdida de información. La contraparte es que dichos sistemas son complejos, por lo que poseen un alto costo de implementación y acarrean el problema de la complejidad de código.

Un desafío importante para los sistemas de recomendaciones es poder adaptarse a las preferencias reales a partir de un dominio acotado, con información incompleta. En nuestro caso, era muy relevante poder recomendar repositorios que el usuario desconozca, pero las interacciones de los usuarios no siempre se refleja en los datos.

Finalmente, con todo esto en cuenta, creemos que en el proyecto actual se logró eficiencia y efectividad en las recomendaciones, y que al mismo tiempo se cuenta con una buena estructura de datos que permite la explicación y por lo tanto la credibilidad de las recomendaciones. Se encontraron también muchas oportunidades de mejora y extensión del proyecto. Creemos que esta rama de la Inteligencia Artificial va a ganar más relevancia a medida que la cantidad de datos digitales siga creciendo exponencialmente, y ésto va a poder ser aprovechado en el futuro.