IMPACTO NEGATIVO DE BOTS EN ESTADÍSTICAS DE USO DE REPOSITORIOS DIGITALES:

Análisis de un caso y estrategia aplicada

Rafael Bértoli
PREBI-SEDICI; Universidad Nacional de La Plata, Argentina | bertoli.rafa@sedici.unlp.edu.ar
https://orcid.org/0009-0001-7707-9998

Ariel Jorge Lira
PREBI-SEDICI; Universidad Nacional de La Plata, Argentina | alira@sedici.unlp.edu.ar
https://orcid.org/0000-0003-3647-3101

DOI: *10.22477/xiv.biredial.410*
EJE TEMÁICO: *Evaluación y métricas alternativas*

RESUMEN

La detección de bots en servicios web es una necesidad presente desde la masificación de internet. Estos agentes recorren la información disponible en la web para diversos fines como ser desarrollo de motores de búsqueda, análisis SEO o, recientemente, para entrenamiento de modelos de inteligencia artificial. Los repositorios son particularmente interesantes para estos agentes, gracias a que ofrecen información de calidad, controlada y descrita por metadatos curados. Independientemente de su objetivo, el repositorio es continuamente cosechado por múltiples bots, lo que produce caídas ocasionales y alteraciones en los registros de uso utilizados para la generación de reportes estadísticos, que sirven para medir el impacto real de las obras preservadas y publicadas. En este trabajo se presenta la experiencia del repositorio Servicio de Difusión de la Creación Intelectual (SEDICI) en el análisis y depuración de registros recolectados durante 13 años. La detección se realiza mediante el análisis del comportamiento de los bots, como uso excesivo, patrones de uso anómalos, escaneos y ataques. Luego, se utiliza un modelo de IA para marcar bots con comportamiento poco evidente no identificados como tales. Aplicadas estas estrategias, se logró eliminar más de 50 millones de registros de uso provenientes de bots atípicos que acceden de forma sistemática y recurrente al repositorio.

Palabras-clave: Robots,aprendizaje automático, repositorios, estadísticas de uso.

ABSTRACT

Bot detection in web services has been a necessity since the mass adoption of the internet. These agents traverse the information available on the web for various purposes such as developing search engines, SEO analysis, or, recently, for training artificial intelligence models. Repositories are particularly interesting to these agents because they offer high-quality, controlled information described by curated metadata. Regardless of their objective, the repository is continuously harvested by multiple bots, which causes occasional downtime and alterations in the

usage records used for generating statistical reports, which serve to measure the real impact of the preserved and published works. This work presents the experience of the Servicio de Difusión de la Creación Intelectual (SEDICI) repository in the analysis and purging of records collected over 13 years. Detection is performed by analyzing the bots' behavior, such as excessive use, anomalous usage patterns, scanning, and attacks. Subsequently, an AI model is used to flag bots with less evident behavior that were not identified as such. Applying these strategies, it was possible to eliminate more than 50 million usage records originating from atypical bots that systematically and recurrently access the repository.)

Keywords: Bots, machine learning, repositories, usage stats.

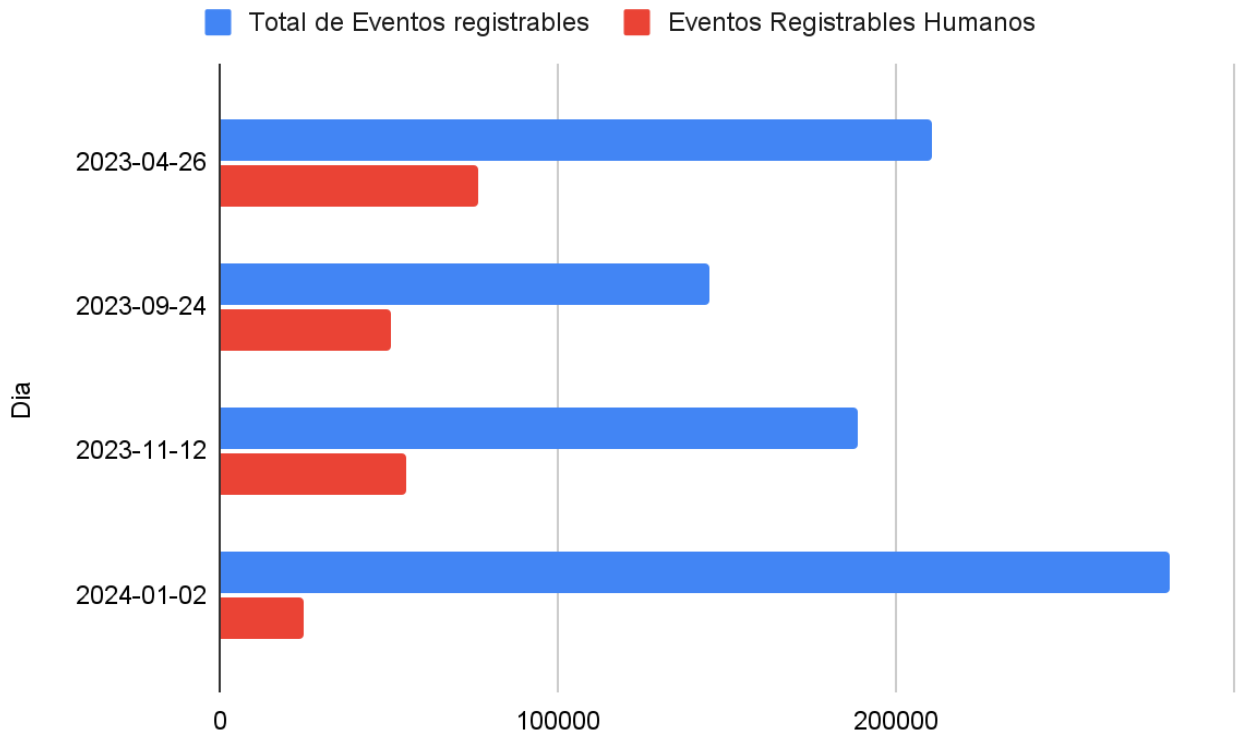
INTRODUCCIÓN

La presencia de *bots* en la web ha aumentado con el paso de los años y todo indica que este crecimiento seguirá aumentando en el futuro. En el reporte de Imperva de 2023 (Imperva, 2024) se afirma que el 49,8% del tráfico de internet es generado de manera automática; en 2022 la cifra era 47,7% (Imperva, 2023), mientras que en 2020 (Imperva, 2020), la cifra sólo era 37,2%. Los *bots* más clásicos, como *crawlers* o *spiders,* buscan acceder todo tipo de información dentro de un sitio para explotar posteriormente los datos. Además de estos *bots*, en un principio "buenos", existen también otros que recorren el sitio buscando vulnerabilidades a explotar y en algunos casos realizar daños.

Esta situación se vio complicada en los años recientes con el crecimiento de tecnologías de inteligencia artificial, en particular por los modelos de grandes lenguajes (LLMs) como GPT, Gemini, Llama, Bard, entre muchos otros. Estos modelos requieren para su entrenamiento grandes volúmenes de datos de múltiples disciplinas y fuentes. Sus datos también se usan para el desarrollo de servicios de detección de plagios, que rastrean todo tipo de publicaciones científicas, tanto sus metadatos como documentos completos.

En síntesis, los sitios son accedidos a todas horas tanto por *bots* como por humanos y, en general, es muy difícil diferenciarlos.

Figura 1 - Cantidad de accesos por origen en el repositorio https://sedici.unlp.edu.ar tomando como muestra 4 días aleatorios dentro del periodo 2023-2024



Fuente: Elaboración propia (2025).

Clasificar correctamente entre quien accede como “bot bueno”, “bot malo” y “usuario real” es una temática ampliamente estudiada que afecta a todo tipo de sitios web. Se pueden encontrar soluciones específicas para distintas aplicaciones en particular, como Wei & Nguyen (2019), que exploran una solución a este problema dentro del contexto de Twitter. También son exploradas múltiples tecnologías para realizar esta tarea, desde análisis de grafos, como en Collins & Reiter (2007), datos biométricos (Acien *et al.*, 2021), el uso de machine learning (Saleem *et al.*, 2020) o la aplicación de honeypots (McKenna, 2016).

Los repositorios no son una excepción y hasta se podría decir que para algunos tipos de *bots* los repositorios representan el néctar de la web, ya que brindan cúmulos curados de información con publicaciones, datos y metadatos bien formados. Ante esta realidad, un repositorio debería tener alguna manera de poder diferenciar este tipo de accesos independientemente de la tarea que se desee cumplir. Para dar un ejemplo, las estadísticas que se pueden generar basándose en los registros de uso pueden verse comprometidas debido a que normalmente se contabilizan eventos generados tanto por usuarios humanos como *bots*, por no disponer de una buena herramienta que permita diferenciar eventos generados de manera automática.

Particularmente, con la finalidad de la generación de reportes estadísticos se requiere una fuente de información confiable, y para lograr esta confianza se debe tener una estrategia

para mantenerla de la manera más correcta posible. Una primera aproximación al problema, aunque un poco ingenua, sería aplicar una política en la que se confíe en que todo bot debe identificarse a sí mismo como tal en el encabezado *userAgent*[1]. De esta manera, todo usuario que no se identifique como *bot*, se lo considerará un usuario humano, y como tal, se lo tendrá en cuenta a la hora de generar reportes.

Profundizando en esta idea, en los últimos años se han desarrollado redes federadas que generan reportes de uso, como *UsageCounts* de OpenAire (Pierrakos, 2020), a partir de eventos de uso de múltiples repositorios aportantes. En este contexto, cada repositorio realiza una limpieza interna de datos a partir de una prácticas estandarizadas, como el caso del "*COUNTER Code of Practice rules on usage events*"[2]. El problema de estas prácticas es que se basan en listas que son limitadas y normalmente no se actualizan con la cantidad creciente de *bots* que aparecen todos los días, ni sus direcciones IP, ni sus agentes de usuario. Para complejizar aún más la labor de detección, cuando los datos se envían a muchos de estos servicios centralizados, se anonimiza parte de sus datos como la terminación de la IP para evitar atentar contra las personas. Esta acción, positiva desde el punto de vista de uso de datos personales, restringe aún más la capacidad de detección de accesos espurios en estos datos centralizados. *UsageCounts* compara sus datos con una lista mantenida por Matomo donde añaden spamers reconocidos de manera regular, comparación que no resulta una estrategia lo suficientemente robusta como para generar reportes estadísticos sobre la agregación de eventos de uso de múltiples fuentes. Por eso resulta vital depurar los eventos en origen, donde se cuenta con la mayor cantidad de información disponible de quien accede.

Es relevante destacar que esta problemática está presente en otros ámbitos fuera de los repositorios, y asimismo perjudica la calidad de los servicios que se brindan. Las soluciones aplicadas en general son limitaciones por tasa de acceso (*rateLimit*) o servicios externos pagos que actúan como punto de acceso inicial y filtran las solicitudes en función de múltiples configuraciones, por ejemplo Cloudflare.

En este trabajo se presenta, de manera general, la aplicación de una estrategia que, a partir de múltiples etapas consecutivas, permite depurar los registros de uso, eliminando el tráfico proveniente de *bots*. Usando esta estrategia, se presenta la implementación particular para SEDICI, repositorio institucional de la Universidad Nacional de La Plata (UNLP), y se expone cómo su aplicación mejora la confianza de los reportes estadísticos generados.

[1] userAgent. Disponível em: https://developer.mozilla.org/es/docs/Web/HTTP/Reference/Headers/User-Agent Acesso em: 2 set. 2025.
[2] COUNTER Code of Practice rules on usage events. Disponível em: https://openaire.github.io/usage-statistics-guidelines/service-specification/service-spec/#application-of-counter-code-of-practice-rules-on-usage-events Acesso em: 2 set. 2025.

CARACTERIZACIÓN DE *BOTS*

Distintos autores han caracterizado a los *bots*que acceden a los sitios web según su comportamiento u objetivo. En Li *et al.* (2021) los autores proponen el concepto de *bots* "benignos", *bots* "maliciosos" y "grises". Los *bots* benignos son aquellos que agregan un valor o utilidad al sitio web visitado; los *bots* maliciosos son aquellos que intentan vulnerar el sitio y los grises, aquellos que no tienen una intención clara. En la experiencia de los autores y a los fines de la detección, se pueden dividir a los *bots* en dos grandes grupos:

Aceptables:

- C*rawlers* buenos: aquellos que cumplen las normas expresadas en el robot.txt, se identifican a sí mismos como *bots,* ya sea mediante el uso de alguna palabra como *Bot* en su nombre, o proporcionando una lista de las direcciones IP desde las cuales se realizarán sus procesos. Ejemplo de estos casos es GoogleBot que, al indexar un sitio, aparecerá en las búsquedas de Google, se identifica a sí mismo como tal, además de publicar sus direcciones de origen en una página web[3]. Otro caso puede ser el de Anthropic y su *chatbot* Claude, que se identifica como tal en el encabezado del agente de usuario.
- *Bots* de control: dentro de este subgrupo se incluyen herramientas que acceden de manera automática al sitio para su mejor funcionamiento. Un ejemplo es un monitor que acceda regularmente al sitio y avise cuando el sitio no se encuentre accesible (Zabbix, Nagios, etc). Otro tipo de *bots* de control son aquellos que realizan pruebas de intrusión o *pentest* en búsqueda de vulnerabilidades, para detectarlas y resolverlas antes de que un usuario con malas intenciones se aproveche de ellas.
- Otros *bots*: no aportan ninguna utilidad al sitio y generalmente se desconoce su finalidad. Sin embargo, que no sean esperados no implica que sean inválidos, ya que suelen respetar las indicaciones dadas en el robots.txt[4], se identifican como *bots*, respetan las buenas prácticas y no representan una amenaza. Algunos casos son los *bots* de análisis SEO, de detección de plagios, y muchos otros que rastrillan los repositorios con fines académicos.

No aceptables:

- *bots* con malas prácticas: típicamente son crawlers que acceden a gran cantidad de datos del sitio, pero a diferencia de los "buenos", estos no suelen tener buenas prácticas ni respetar las recomendaciones dadas en el robots.txt. Ocasionalmente, este tipo de *bots* tampoco se identifican a sí mismos como tales, y presentan en el agente de usuario cosas como "Mozilla" para sortear cualquier regla básica que pueda tener el sitio.

[3] Disponível em: https://developers.google.com/search/apis/ipranges/googlebot.json Acesso em: 2 set. 2025.
[4] Disponível em: https://www.rfc-editor.org/rfc/rfc9309.html Acesso em: 2 set. 2025.

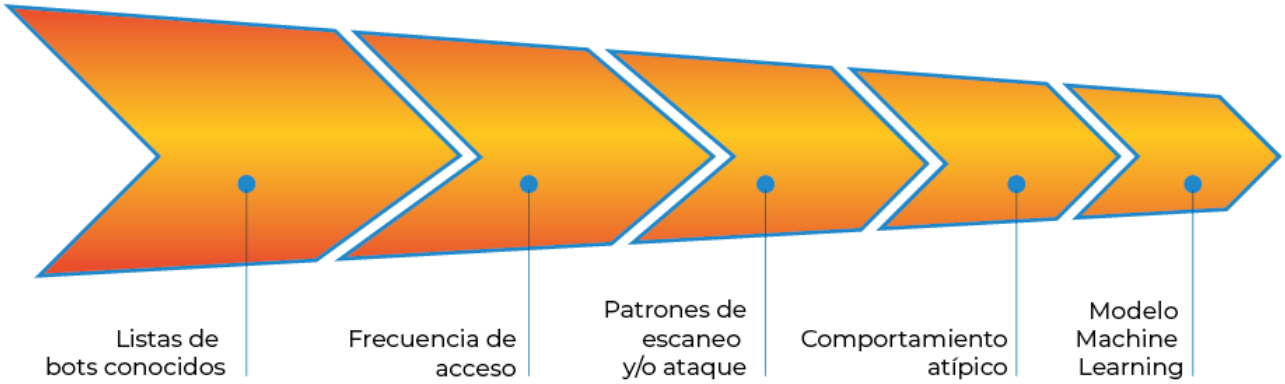
- *Bots* maliciosos: son agentes que realizan escaneos de aplicaciones, búsqueda de vulnerabilidades conocidas o frecuentes e incluso suelen realizar ataques. Las acciones son mayormente realizadas de manera automática. Este comportamiento es visible principalmente en los parámetros de las solicitudes. Los ataques pueden variar desde simples intentos de inyección SQL (*SQL Injection*), intentos de ejecución remota de código (*RCE*), XSS u otro tipo de vulnerabilidades comunes. La diferencia principal entre este tipo de bots y aquellos orientados a la seguridad es que en estos casos buscan vulnerabilidades para poder explotarlas y no se identifican de manera alguna .
- Redes de *bots*: en este caso no se ve un comportamiento excesivo desde una IP sino que se observan múltiples direcciones IP que participan de una acción colectiva como un ataque en masa, scrapping o descargas de grandes cantidades de archivos. A la hora de analizar los accesos de un rango de direcciones, se observan tasas mayores a las esperables, que tienen un comportamiento muy similar. De igual manera que en otros *bots*, estos no serán aceptables cuando no respeten las buenas prácticas y especialmente cuando no se identifiquen como tales.

Si se caracterizan los *bots* de manera sencilla, se puede observar que tienen un comportamiento muy disímil al de los humanos, como el acceso por ráfagas y una gran velocidad entre accesos, efectuar demasiadas descargas de recursos similares, enviar solicitudes con parámetros inválidos, pedir páginas inexistentes, entre otras.

ETAPAS DEL PROCESO DE CLASIFICACIÓN

Una vez entendidos los principales tipos de *bots* que se pueden encontrar en el contexto de los repositorios, es preciso plantear una estrategia para reducir el impacto de su presencia en los registros de uso. Para lograr esto, se presenta una estrategia de 5 pasos que, aplicados cada uno de manera independiente, permiten mejorar los registros de uso y las estadísticas que se generan.

Figura 2 - Etapas del proceso de depuración de *bots*.



Fuente: Elaboración propia (2025).

La primera etapa busca mantener una lista de aquellos agentes de usuario (*userAgent*) para los que se conoce su naturaleza, como es el caso de GoogleBot, un bot que se sabe que va a acceder desde cierto rango de direcciones IP y se va a identificar a sí mismo como tal. Esto permite identificar en la primera etapa a todos aquellos *bots* que respetan buenas prácticas y que representan una gran cantidad de los accesos recibidos. Tanto la lista publicada por Matomo mencionada anteriormente, como las listas de userAgent de DSpace, pueden servir como insumo para esta etapa.

La segunda etapa busca detectar aquellas IP que presenten una tasa de acceso superior a la esperable por un usuario humano. Normalmente, un bot presenta ráfagas de acceso mucho mayores a las esperadas por un humano, por la cantidad de información que puede procesar un sistema automático en comparación a lo que puede realizar un humano. Esta aproximación se realiza sobre los datos históricos y suele detectar una gran cantidad de eventos, ya que quienes tienen mayor tasa de acceso son aquellos con mayor cantidad de accesos totales. Esta tarea puede ser compuesta del análisis de múltiples subtareas, donde cada una de estas busquen analizar rangos de tiempo distintos, para lograr un evaluación más granular del comportamiento y mejores resultados. Los rangos de tiempo que se busque analizar dependen de la implementación particular de la estrategia, pero realizar múltiples iteraciones de este análisis, por años, meses, semanas y/o días, resulta útil para lograr una estrategia más robusta.

En la tercera etapa se busca reconocer qué usuarios realizaron peticiones en las que alguno de sus campos fue modificado con una carga indebida en búsqueda de vulnerabilidades. Para lograr esta tarea, se suelen identificar los parámetros de aplicación para luego buscar dentro de ellos ciertos patrones que representen ataques al sitio. Para ejemplificar este tipo de patrones, en caso de buscar ataques de *SQL injection*, se puede buscar patrones comunes como *SELECT*, *UNION* o *OR*. En casos de buscar indicios de *path traversal*, se puede buscar pa-

trones como "../" o directorios que pueden ser objetivo de ataques como "/etc/passwd". Existen numerosos tipos de ataques, muchos de ellos son de alcance general mientras que otros son específicos a ciertas aplicaciones populares como Wordpress.

En la cuarta etapa se pretende realizar un análisis más detallado del comportamiento del *usuario* en el sitio. Es normal para algunos *bots* realizar sólo un tipo de acción como descargar archivos o solicitar resultados de búsqueda. Este comportamiento, cuando se repite, no es asimilable a usuarios humanos que, en caso de realizar múltiples accesos dentro del sitio, suelen hacer acciones variadas recorriendo el sitio a través de enlaces que apuntan a diferentes rutas. Para ejemplificar un caso, un humano puede buscar un item, visitarlo y después descargarlo. Distintos *bots* solo descargan cientos de pdf sin siquiera realizar una única búsqueda ni realizar una sola visita a los registros de las publicaciones que descargan. Al determinar qué patrones representan acciones humanas o de *bots* y aplicándolos a la información de la que se disponga, se puede diferenciar otro gran número de usuarios.

Por último, como este entorno es altamente dinámico, con nuevos patrones de comportamiento de *bots* que surgen día a día, no es suficiente basarse en instrumentos rígidos como las listas de IP, listas de *userAgent*, tasas de acceso o patrones de comportamiento anómalo. Por ello, se incorpora una última etapa del análisis que delega la clasificación de los accesos restantes a un modelo de inteligencia artificial entrenado en esta tarea. Si bien puede ser costoso generar un modelo de IA que permita realizar esta tarea de clasificación, este tipo de soluciones son más dinámicas por su naturaleza y resultan atractivas para usar en aquellos casos donde no se pueda determinar solo con la experiencia previa la naturaleza de los usuarios.

APLICACIÓN EN SEDICI

SEDICI es el repositorio institucional central de la UNLP y como tal alberga toda la producción científica generada en esta institución. El repositorio, creado en 2003 y migrado a dspace en 2012, un software libre ampliamente adoptado en todo el mundo, funciona actualmente sobre la versión 5.10 de DSpace. Particularmente, DSpace tiene un núcleo o *core* de Solr llamado *statistic* en el que se registran todos los eventos de uso generados en el sitio. Allí se encuentran las vistas, descargas y búsquedas realizadas en el sitio. Este índice funciona como registro histórico y mantiene los eventos realizados en SEDICI desde el año 2012 hasta la actualidad, y en él se encuentran más de 195 millones de registros. Como norma general, estos registros de uso en Solr son la fuente de análisis para determinar acciones, con excepción de la estrategia de machine learning que usa una fuente más completa (access.log). Luego, las acciones de limpieza siempre se hacen sobre el core de Solr, eliminando los registros asimilables a *bots*.

Cabe destacar que, a pesar de tener un número muy elevado de eventos de uso - casi 200 millones- estos son el resultado de sucesivas tareas de depuración de *bots* conocidos. Se

estima que previo al inicio de este trabajo, ya se habían eliminado alrededor de 100 millones de registros correspondientes a IPs de *bots* conocidos.

Dentro de los registros almacenados en Solr, se dispone de la dirección IP del usuario que generó el evento, el agente de usuario utilizado y el *referer* en algunos casos, entre otros. DSpace, como parte de su implementación, tiene una copia del repositorio COUNTER-Robots[5] con direcciones IP y agentes de usuario pertenecientes a *bots*, y al dispararse los eventos, los compara con estas listas para saber si marcarlos con un flag dentro de los eventos llamada *isBot*. En los casos donde esta comparación sea cierta, el flag se marcará como *true* indicando que fue realizado por un bot. Para este trabajo, se tomó la decisión de eliminar estos casos para mantener el *core* de SOLR lo más reducido posible, utilizando la opción de configuración *logBots* en falso.

Debe destacarse, como complemento de la primera etapa, la eliminación de casos no detectados dentro del registro histórico, ya sea por no estar implementado el mecanismo de las listas o registros previos a añadir el caso dentro de las listas. Realizar un primer borrado para estos agentes conocidos, como aquellos que tengan la palabra *Bot*, puede resultar una buena opción para eliminar residuos de casos antiguos.

De forma empírica, se determinaron las tasas de acceso esperables para un usuario humano y, en base a estas, se calcularon umbrales muy altos para determinar que aquellos *usuarios* que superen este límite, se consideran *bots*. Los límites se eligieron elevados, de forma tal que permitan diferenciar humanos de *bots* y estar suficientemente seguros de que los registros a eliminar hayan sido realmente generados por *bots* y minimizar la cantidad de falsos positivos. Determinado el límite/umbral, se evalúan las direcciones IP que realizaron mayor cantidad de accesos que el límite esperado dentro del rango de evaluación. En el caso de aplicación, se determinó realizar un análisis sobre cada año (2012, 2013, 2014, ...), cada mes y, por último, por intervalos de 3 días consecutivos. Al utilizar este enfoque y en el orden expresado, se logró atacar primero a aquellos casos de mayor impacto y limpiar una gran cantidad de registros de *bots*.

En la etapa 3, para eliminar registros que indiquen intentos de ataque, se determinaron los caminos típicos, como encabezados HTTP y parámetros. En este caso, los campos más usados eran el agente de usuario (*userAgent*), el de origen (*referrer*) y los parámetros de búsqueda recibidos en la parte Query de la solicitud GET. Luego, se realizaron búsquedas sobre cada uno de estos campos donde se buscan patrones comunes en este tipo de comportamiento dependiendo del tipo de ataque como *SELECT* o *exec*.

En la cuarta etapa de la estrategia, se determinó una serie de lineamientos para entender el comportamiento común de usuarios humanos dentro del sitio. Normalmente, son reglas de sentido común las que se aplican al comportamiento en el sitio, por ejemplo: “se es-

[5] Disponível em: https://github.com/atmire/COUNTER-Robots Acesso em: 2 set. 2025.

pera que el usuario realice cierto número de vistas de ítems para realizar descargas". En el caso de SEDICI, se espera una relación menor al 10%, es decir que para descargar 100 archivos del sitio, se espera que el usuario realice 10 visitas a ítems. Es claro que es un límite muy alto, no obstante permite detectar con bastante seguridad los casos positivos, es decir, los *bots*.

Por último, agotadas las etapas descritas, se aplica una herramienta que se apoya en un modelo de *machine learning* entrenado para que, dado un access-log de un día del repositorio, identifique qué accesos fueron realizados por *bots*. Dentro del access-log se registran todos los accesos al sitio, no solo los eventos registrables, manteniendo el recurso solicitado, método HTTP utilizado, hora y demás. El modelo fue entrenado con esta fuente de datos para reconocer el comportamiento normal del sitio, no sólo lo registrable en SOLR. Esta herramienta se ejecuta de manera diaria, eliminando los registros que escapen a todas las reglas expresadas anteriormente, lo que da por resultado el *core* estadístico de SOLR más correcto posible hasta el momento. Para poder realizar una limpieza del registro histórico, se precisa tener un access log para cada uno de los días que se desee hacer la limpieza, por lo tanto esta etapa está limitada a los datos en posesión, que en el caso particular de SEDICI, es desde 29 de octubre del 2022, permitiendo ejecutar esta etapa desde dicha fecha hasta la actualidad.

En Bértoli *et al.* (2024) se investigó cómo desarrollar este modelo, mediante múltiples pruebas sobre gran parte de los algoritmos utilizados en el área, y qué procesos realizar sobre esta fuente de datos para su entrenamiento. En la implementación de esta estrategia se optó por utilizar el algoritmo de random forest (Ho, 1995), que tiene un valor de 97% de casos bien clasificados. Esta etapa no sólo se apoya en la clasificación del modelo, del día, sino que también utiliza las clasificaciones previas para determinar si un usuario es *bot.* Para ser clasificado *bot* por primera vez, el modelo debe estar seguro, pero una vez clasificado, se tiende a creer que el usuario es bot nuevamente.

RESULTADOS OBTENIDOS

Tabla 1 - Comparación de rendimiento entre modelos de clasificación para detección de *bots* en *logs web*.

Model	Accuracy	Precision		Recall		F1-Score		AUC	
		No Bot	Bot	No Bot	Bot	No Bot	Bot	No Bot	Bot
Random Forest	0.9712	0.9696	0.9780	0.9947	0.8836	0.9820	0.9284	0.9854	0.9690
XGBoost	0.9709	0.9691	0.9783	0.9947	0.8818	0.9818	0.9275	0.9854	0.9689
Gradient Boosting	0.9703	0.9694	0.9740	0.9937	0.8830	0.9814	0.9263	0.9851	0.9671
AdaBoost	0.9622	0.9649	0.9509	0.9880	0.8661	0.9763	0.9065	0.9778	0.9557

Multiperceptron	0.9687	0.9679	0.9722	0.9933	0.8773	0.9804	0.9223	0.9771	0.9536
K-NN	0.9577	0.9692	0.9129	0.9774	0.8843	0.9733	0.8984	0.9701	0.9433
Regresión Logística	0.9623	0.9578	0.9831	0.9961	0.8362	0.9766	0.9037	0.9660	0.9330
LDA	0.8323	0.8294	0.8817	0.9942	0.2294	0.9039	0.3390	0.9572	0.8653
SVM (RBF)	0.8823	0.9030	0.7964	0.9616	0.5867	0.9293	0.6199	0.9494	0.8505
SVM (Linear)	0.8461	0.8560	0.7840	0.9749	0.3654	0.9101	0.4494	0.9396	0.8005
SVM (Polly)	0.7709	0.8043	0.8133	0.9477	0.1108	0.8440	0.1312	0.8012	0.5160
Gaussian N Bayes	0.7869	0.7935	0.4603	0.9864	0.0430	0.8795	0.0786	0.7532	0.3514

Fuente: Bértoli *et al.* (2024).

La primera etapa se realiza en su mayoría desde DSpace, excluyendo de los logs los accesos de *bots* conocidos (por user agent o por IP). Por este motivo, es difícil determinar cuántos registros son descartados en esta etapa. A modo orientativo, utilizando el *log* de accesos de un día y comparando con la lista de direcciones IP y agentes de usuario reconocidos como bot, podemos determinar que cerca de 60% de registros son evitados durante esta etapa. Posteriormente, de los 195 millones de registros almacenados, se eliminaron 3 millones de registros, analizando el agente de usuario, buscando elementos como *Bot*, *Crawler* o *Spider*.

En la segunda etapa se eliminaron los registros generados por IP con una tasa de acceso fuera de lo esperado. Primero se analizó aquellas con más accesos de lo esperado por año y se eliminaron cerca de 20 millones de registros. Analizando por mes se eliminaron poco más de 9 millones, y por día más de 8 millones. Al atacar esta problemática en este orden, se consigue, previo a realizar un análisis más minucioso y costoso en el tiempo, realizar el borrado de gran parte de los registros y, por lo tanto, mejorar la ejecución de las etapas posteriores.

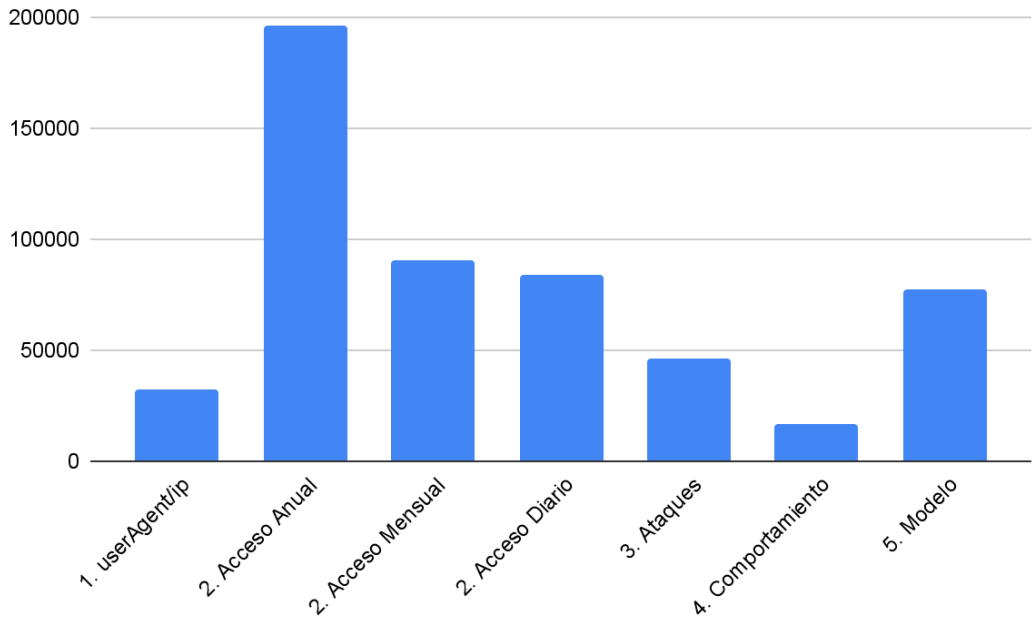
En la tercera etapa se detectaron unos 4 millones de registros que tienen la forma de ataques o de escaneos de vulnerabilidades. A partir del análisis del agente de usuario sólo se detectaron poco más de 13 mil registros (posiblemente porque ya se hizo una depuración por userAgent en la etapa 1), por referer se logró eliminar casi 4 millones de registros y por búsqueda poco más de 60 mil.

La cuarta etapa analiza el comportamiento dentro del sitio, y basándose en este comportamiento se logró eliminar más de 1,5 millones de registros que no tenían un comportamiento asimilable al de un humano. En la implementación de esta etapa se expresaron límites altos para no eliminar registros de usuarios que quizá no fuesen realmente *bots* y, por lo tanto,

es una de las etapas que menos registros pudo eliminar. Es posible que también se deba a ser una de las últimas etapas y que gran parte de este comportamiento ya fuese eliminado en etapas anteriores.

En la última etapa, el modelo detectó más de 7 millones de registros generados por agentes automáticos entre el 29 de octubre del 2022 y 19 de febrero del 2024. Este proceso no es aplicable a toda la historia del repositorio ya que necesita un access-log completo para poder realizar el análisis, y al momento de la escritura de esta ponencia solo se disponen de las fechas mencionadas.

Figura 3 - Total de registros eliminados divididos por causa.



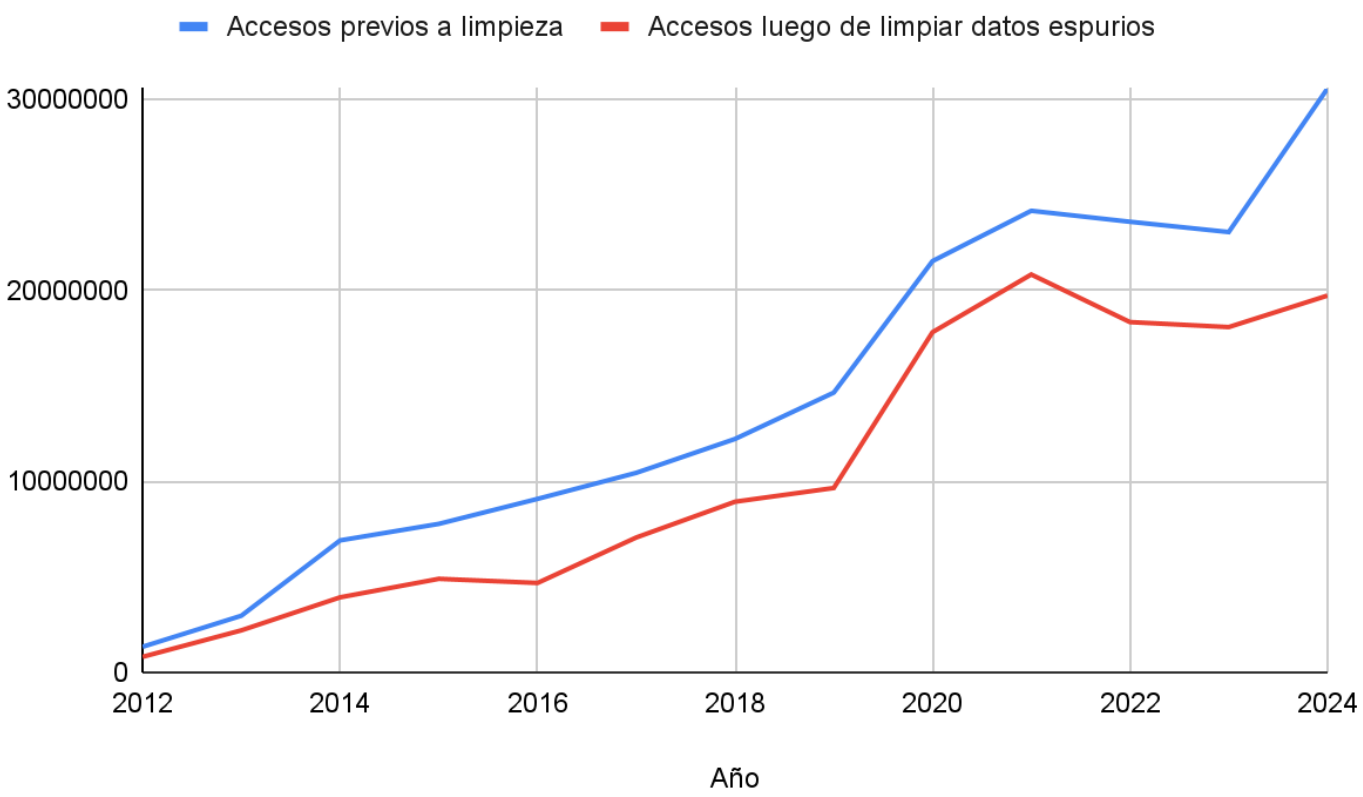
Fuente: Elaboración propia (2025).

Si se analiza el orden de ejecución de cada etapa, visible en la Figura 3, se puede observar que en cada nueva etapa los registros eliminados son cada vez menores, lo cual es explicable a que ciertos patrones no sean exclusivos a una única etapa. Normalmente los ataques pueden realizarse en rafagas de tiempo que pueden llegar a ser detectadas en análisis diarios, o comportamiento claramente bot como solo realizar descargas, ser detectado por un rate de accesos mayor al esperado en un año. De esta manera, cada etapa trabaja con un conjunto menor de posibles registros a ser eliminados, y un porcentaje de los registros detectables en la etapa fueron eliminados en etapas anteriores.

El modelo de machine learning, no es tan afectado en este aspecto, probablemente debido a que la fuente de datos utilizada por el modelo de machine learning es distinta a la usada en las etapas previas. Al usar el access-log, nueva información es tenida en cuenta para clasificar comportamiento de bot y de esta manera se logra generar buenos resultados pese a

haber ejecutado múltiples etapas de borrado previamente.

Figura 4 - Número de registros almacenados antes y después de la limpieza.



Fuente: Elaboración propia (2025).

En la Figura 4 se puede ver cómo esta combinación de estrategias mantiene un nivel de borrado estable a lo largo del tiempo. No obstante, en los últimos años los registros almacenados se disparan de una manera que no fue vista en el pasado del repositorio, posiblemente por la masificación de las nuevas tecnologías de IA. Pese a este aumento, luego del borrado de los registros realizados por bot, no se nota tanto aumento en los registros almacenados. También es notable cómo desde 2022, etapa desde la cual se puede ejecutar el modelo de inteligencia artificial, se logra eliminar una mayor proporción de registros espurios.

Finalmente, después de ejecutar todas estas etapas, se logró eliminar más de 54 millones de registros del periodo 2012-2025, poco más del 25% de los registros originales, sin contar las eliminaciones realizadas antes de la implementación de esta estrategia.

CONCLUSIONES

La estrategia planteada permitió clasificar y eliminar gran cantidad de eventos de uso provenientes de *bots*. Se puede afirmar que gracias a la depuración realizada, el registro de eventos de uso del repositorio refleja de mejor manera el uso real del sitio por parte de usua-

rios humanos y por tanto permitirá generar estadísticas de uso más confiables.

La cantidad de falsos positivos, es decir, usuarios detectados como *bots*, es extremadamente baja gracias a las diferentes etapas propuestas, el orden de aplicación entre ellas, la cautelosa selección de límites en tasas de acceso, los datos validados de agente de usuario e IP de *bots*, y la elección de un modelo de machine learning que sólo clasifica cuando posee una elevada tasa de certeza. Lógicamente, pueden quedar falsos negativos sin eliminar, es decir, *bots* que siguen considerándose como usuarios y que no se detectaron como *bots*. Sin embargo, como se mencionó, esta es una primera aproximación al problema, que lo resuelve en parte y sobre la que está previsto seguir realizando mejoras por medio de ajustes en el modelo y quizás la incorporación de nuevas etapas.

Cabe aclarar que estas estrategias son aplicables en otros repositorios o portales de publicación académicos, tanto los que funcionan sobre DSpace como los que usan otros softwares con registros internos de eventos de uso, como OJS. En la implementación pueden variar detalles específicos como los límites de accesos diarios, los patrones esperables o incluso se puede reentrenar el modelo con patrones de acceso propios, pero las etapas deberían ser aplicables de igual manera.

En SEDICI, la aplicación de esta estrategia resultó ser útil y se cree que se pueden lograr resultados similares en la aplicación de esta estrategia en otros sitios con una problemática similar. Con la reducción de un cuarto del total de registros almacenados, se puede decir que la implementación fue correcta para el sitio y en todas las etapas se obtuvieron buenos resultados.

La virtud de esta estrategia consiste en separar la solución en múltiples etapas independientes entre sí, de manera que el buen funcionamiento de cada una es independiente de las otras. De esta manera, si una de las etapas debe modificarse para afrontar nuevas dificultades en el futuro, las etapas previas y posteriores pueden funcionar igualmente y solo se debe actualizar el funcionamiento de la etapa correspondiente. Más aún, las decisiones tomadas en cada etapa pueden basarse en fuentes de datos diferentes, como sucede con la última etapa que utiliza los access log de apache, y todas las otras etapas son ejecutadas directamente con los registros en SOLR.

BIBLIOGRAFÍA

Acien, A., Morales, A., Fierrez, J., Vera-Rodriguez, R., & Delgado-Mohatar, O. (2021). BeCAPTCHA: Behavioral bot detection using touchscreen and mobile sensors benchmarked on HuMIdb. Engineering Applications of Artificial Intelligence, 98, 104058. https://doi.org/10.1016/j.engappai.2020.104058

Bértoli, R., Estrebou, C. A., & Lira, A. J. (2024). Aprendizaje automático para la detección de

bots en repositorios digitales. In Anais do XXX Congreso Argentino de Ciencias de la Computación (CACIC). (pp. 12-26). Facultad de Informática (UNLP), La Plata. http://sedici.unlp.edu.ar/handle/10915/172755

Collins, M. P., & Reiter, M. K. (2007). Hit-list worm detection and bot identification in large networks using protocol graphs. In C. Kruegel, R. Lippmann, & A. Clark (Eds.). *Proceedings of the 10rd symposium recent advances in intrusion detection (RAID),*(pp. 276–295). Springer, Gold Coast, Australia. https://doi.org/10.1007/978-3-540-74320-0_15

Pierrakos, D., Czerniak, A., & Schirrwagen, J. (2020). OpenAIRE usage counts: The analytics service of OpenAIRE Research Graph. In *Proceedings of the 16rd Research Data Alliance Plenary Meeting*, Virtual. Zenodo. https://doi.org/10.5281/zenodo.4268144

Ho, T. K. (1995). Random decision forests. In *Proceedings of the 3rd International Conference on Document Analysis and Recognition* (pp. 278–282). IEEE, Montreal, QC, Canada. https://doi.org/10.1109/ICDAR.1995.598994

Imperva Inc. (2020). *2020 bad bot report — Resource library.* https://www.imperva.com/resources/resource-library/reports/2020-bad-bot-report/I

Imperva Inc. (2023). *2023 bad bot report — Resource library.* https://www.imperva.com/resources/resource-library/reports/2023-imperva-bad-bot-report/

Imperva Inc. (2024). *2024 bad bot report — Resource library.* https://www.imperva.com/resources/resource-library/reports/2024-bad-bot-report/

Li, X., Azad, B. A., Rahmati, A., & Nikiforakis, N. (2021). Good bot, bad bot: Characterizing automated browsing activity. In *Proceedings of the IEEE Symposium on Security and Privacy* (pp. 1589–1605). IEEE, São Paulo. https://doi.org/10.1109/SP40001.2021.00079

McKenna, S. F. (2016). *Detection and classification of web robots with honeypots* [Doctoral dissertation, Naval Postgraduate School]. https://apps.dtic.mil/sti/tr/pdf/AD1027491.pdf

Saleem, S., Sheeraz, M., Hanif, M., & Farooq, U. (2020). Web server attack detection using machine learning. In *Proceedings of the International Conference on Cyber Warfare and Security (ICCWS)* (pp. 1–7). IEEE, Islamabad, Pakistan. https://doi.org/10.1109/ICCWS48432.2020.9292393

Wei, F., & Nguyen, U. T. (2019). Twitter bot detection using bidirectional long short-term memory neural networks and word embeddings. In *Proceedings of the IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)* (pp. 101–109). IEEE, Los Angeles, CA, USA. https://doi.org/10.1109/TPS-ISA48467.2019.00021

ANEXO 1

RESUMEN BIOGRÁFICO DE LOS AUTORES

Rafael Bertoli

Estudiante avanzado en Informática por la Universidad Nacional de La Plata (UNLP). Integra el equipo de desarrollo de PREBI-SEDICI, donde se desempeña en tareas vinculadas a la administración y personalización del repositorio institucional SEDICI, basado en la plataforma DSpace. Su trabajo se centra en el desarrollo de soluciones con tecnologías de software libre para la gestión, preservación y difusión del conocimiento científico, con una perspectiva orientada a la soberanía tecnológica y la interoperabilidad entre sistemas académicos.

Ariel Lira

Licenciado en Informático por la Universidad Nacional de La Plata, desde 2006 forma parte del equipo de PREBI-SEDICI. Se especializa en repositorios digitales de publicaciones y datos, ciencia abierta y preservación digital. Participa en el desarrollo de herramientas y servicios para la gestión y difusión de la producción académica. Su trabajo contribuye a fortalecer el ecosistema de comunicación científica en acceso abierto.

ANEXO 2

REQUERIMIENTOS DE EQUIPO TÉCNICO PARA LA PRESENTACIÓN DE LA PONENCIA

Ninguno.