Evolución estructural de la red de dependencias de Python: Evidencia de una década de crecimiento del ecosistema

Juan Ignacio Catania[1], Mateo Guerrero Schmidt[1], Sofia Gutierrez[1], Martina Rosario Pérez[1], and Carlos Sarraute[2,3]

[1] Universidad de Buenos Aires
[2] Disruptive Research Institute
[3] Instituto Tecnológico de Buenos Aires (ITBA)
csarraute@itba.edu.ar

Resumen. Presentamos un análisis longitudinal de la red de dependencias del ecosistema de librerías de Python, evidenciando transformaciones estructurales sistemáticas a lo largo de casi una década. Los ecosistemas de paquetes de software pueden modelarse como redes complejas cuya topología refleja prácticas de desarrollo y tendencias tecnológicas. Para caracterizar su evolución, construimos grafos dirigidos de dependencias correspondientes a dos cortes temporales (2016 y 2025) y calculamos métricas estructurales, medidas de centralidad y organización comunitaria. Los resultados muestran una expansión masiva del ecosistema acompañada por cambios topológicos significativos. El número de paquetes crece más de un orden de magnitud, mientras que el número de dependencias aumenta aún más rápidamente, lo que indica un proceso de densificación. Asimismo, observamos un incremento del grado promedio, una disminución de la modularidad y una reconfiguración de los nodos centrales. Mientras que en etapas tempranas predominaban librerías asociadas al desarrollo web y la compatibilidad, en el período más reciente emergen como hubs estructurales paquetes vinculados a ciencia de datos, computación científica y testing automatizado.
Estos hallazgos sugieren que el ecosistema Python evoluciona hacia una arquitectura más interconectada y que su crecimiento sigue patrones consistentes con leyes conocidas de redes complejas, tales como la densificación y la reorganización de hubs. En términos más generales, el estudio aporta evidencia empírica sobre la evolución estructural de ecosistemas tecnológicos a gran escala y destaca el valor de los enfoques de ciencia de redes para su análisis.

Keywords: redes complejas· redes de dependencias· ecosistemas de software· evolución de redes· análisis de centralidad

Structural Evolution of the Python Dependency Network: Evidence from a Decade of Ecosystem Growth

Abstract. We present a longitudinal network analysis of the Python package dependency ecosystem, revealing systematic structural transformations over nearly a decade of evolution. Software package ecosystems can be modeled as complex networks whose topology reflects development practices and technological trends. To characterize this evolution, we construct directed dependency graphs for two large-scale snapshots (2016 and 2025) and compute structural metrics, centrality measures, and community organization.

Our results show a massive expansion of the ecosystem accompanied by clear topological shifts. The number of packages increases by more than an order of magnitude, while dependencies grow even faster, indicating strong densification. We also observe an increase in average degree, a decrease in modularity, and a marked reconfiguration of central nodes. Whereas earlier stages were dominated by web-development and compatibility libraries, recent snapshots show data science, scientific computing, and testing packages emerging as structural hubs.

These findings indicate that the Python ecosystem evolves toward a more interconnected architecture and follows growth patterns consistent with established laws of complex networks, including densification and hub reorganization. More broadly, this study provides empirical evidence that software dependency ecosystems exhibit systematic structural evolution and demonstrates the value of network science methods for understanding the dynamics, organization, and robustness of large technological systems.

Keywords: complex networks· dependency networks· software ecosystems· network evolution· centrality analysis

1. Introducción

El ecosistema de librerías de Python ha experimentado un crecimiento extraordinario durante la última década, impulsado por la expansión de disciplinas como ciencia de datos, aprendizaje automático, computación científica e ingeniería de software. A medida que aumenta la cantidad de paquetes disponibles, también crece la complejidad de las relaciones de dependencia entre ellos, dando lugar a estructuras que pueden analizarse como redes dirigidas de gran escala.

Desde la perspectiva de la ciencia de redes, los ecosistemas de software constituyen sistemas tecnológicos complejos en los que la arquitectura global emerge

de la interacción local entre componentes. El estudio de estas redes permite identificar patrones estructurales, roles funcionales de nodos y regularidades evolutivas que no son evidentes a nivel individual. En particular, analizar la evolución temporal de redes de dependencias resulta fundamental para comprender cómo cambian la centralidad, modularidad y conectividad de estos sistemas a medida que crecen.

Si bien existen estudios que comparan la evolución de múltiples ecosistemas de paquetes (Decan et al., 2019; Kikas et al., 2017), todavía es limitado el conocimiento empírico sobre cómo evoluciona longitudinalmente la arquitectura de un mismo ecosistema a gran escala desde la perspectiva de ciencia de redes. Esta falta de análisis temporales dificulta evaluar si dichos sistemas siguen leyes estructurales conocidas en redes complejas, tales como densificación progresiva o reducción del diámetro efectivo (Leskovec et al., 2007), reorganización de hubs o cambios en modularidad.

En este trabajo analizamos la red de dependencias de librerías de Python en dos cortes temporales separados por casi una década (2016 y 2025). Para cada año construimos un grafo dirigido en el que los nodos representan paquetes y las aristas representan relaciones de dependencia, y calculamos métricas estructurales y de centralidad, analizamos subgrafos relevantes y detectamos comunidades mediante algoritmos de particionado.

Este trabajo realiza las siguientes contribuciones principales: (i) Analiza dos grafos a gran escala del ecosistema PyPI separados por 9 años; (ii) Caracteriza cuantitativamente su evolución estructural; (iii) Identifica cambios en nodos centrales y organización comunitaria; (iv) Proporciona evidencia empírica sobre cómo evolucionan ecosistemas de software reales.

Los resultados muestran que el ecosistema Python no solo crece en tamaño, sino que experimenta una reorganización estructural significativa, con aumento de densidad, reducción de modularidad y desplazamiento de librerías centrales hacia áreas como ciencia de datos, testing y tipado estático.

2. Trabajos relacionados

El análisis de ecosistemas de software mediante herramientas de ciencia de redes se inscribe dentro del estudio general de sistemas complejos representables como grafos. Las redes de dependencias de paquetes constituyen ejemplos naturales de redes tecnológicas dirigidas, comparables a infraestructuras complejas como la Web, redes de citaciones o redes biológicas. Estudios previos han mostrado que los sistemas tecnológicos de gran escala tienden a exhibir regularidades estructurales universales —distribuciones de grado de cola pesada, presencia de hubs, organización jerárquica y modularidad no aleatoria (Valverde y Solé, 2003; Valverde et al., 2005)— que responden a mecanismos generativos subyacentes como el crecimiento incremental y la conexión preferencial (Barabási y Albert, 1999). A nivel macroscópico, se han identificado leyes que caracterizan el crecimiento de grafos reales, como las leyes de densificación y reducción del diámetro efectivo, que muestran que muchas redes empíricas se vuelven progresivamente

más densas a medida que crecen (Leskovec et al., 2007), constituyendo un marco teórico relevante para el análisis de ecosistemas de software.

Dentro del campo específico de redes de dependencias de paquetes, (Kikas et al., 2017) analizaron múltiples repositorios encontrando evidencia de centralización creciente y distribución desigual de dependencias, mientras que (Decan et al., 2019) realizaron una comparación longitudinal de siete ecosistemas de paquetes y observaron que el crecimiento suele acompañarse de mayor acoplamiento estructural y dependencia de un subconjunto reducido de librerías centrales. Un análisis similar ha sido aplicado también en ecosistemas específicos como el de paquetes de R, caracterizando su estructura de dependencias y patrones de colaboración (Mora-Cantallops et al., 2020; Salgado y Caridi, 2022). Desde la perspectiva de robustez, estudios en el ecosistema npm han mostrado que nodos en posiciones estructurales críticas pueden actuar como puntos únicos de falla (Zimmermann et al., 2019). Complementariamente, desde una perspectiva sociotécnica, (Bogart et al., 2016) mostraron que las normas comunitarias y decisiones de compatibilidad gobiernan la evolución de estos ecosistemas, sugiriendo que cambios en prácticas de desarrollo tienen correlatos estructurales observables.

En este contexto, el presente trabajo se distingue de los estudios previos al examinar en profundidad la evolución estructural interna de un único ecosistema —Python— mediante dos instantáneas separadas por casi una década. Este enfoque permite contrastar directamente si regularidades como densificación, reconfiguración de hubs y pérdida de modularidad, documentadas en otros dominios (Leskovec et al., 2007), se manifiestan también en el ecosistema PyPI.

3. Construcción del grafo y métricas de red

3.1. Fuentes de datos

Para llevar a cabo el análisis de la red de dependencias de librerías de Python, construimos dos conjuntos de datos correspondientes a los años 2016 y 2025, ambos originados desde Python Package Index (PyPI), la principal fuente pública y oficial de distribución de paquetes Python.

El dataset de 2025 fue obtenido directamente desde PyPI mediante un proceso de scraping sobre sus metadatos públicos, descrito en detalle en la Sección 3.2. El dataset de 2016 proviene de un trabajo previo realizado por Kevin Gullikson, estudiante de Texas University (Gullikson, 2016), quien también extrajo la información desde PyPI, aunque mediante un método de acceso distinto —disponible en ese entonces— que en la actualidad ya no se encuentra operativo. Ambos datasets son, por tanto, comparables en origen, diferenciándose únicamente en el método de extracción empleado en cada período.

Todo el código, los datos y los grafos resultantes de este trabajo están disponibles públicamente en el repositorio:
https://github.com/empromptu/python-ecosystem-evolution

3.2. Preprocesamiento y limpieza

La descarga de información se realizó a partir de los metadatos publicados por cada paquete en PyPI, los cuales incluyen, entre otros campos, el nombre del paquete, su versión, la lista de dependencias declaradas y una breve descripción del paquete. Una vez recopilados los datos brutos, aplicamos un proceso de limpieza y estandarización que comprendió los siguientes pasos:

1. Eliminación de paquetes duplicados o versiones redundantes, conservando únicamente la versión relevante para cada año.
2. Normalización de nombres de paquetes, debido a variaciones de estilo o inconsistencias en la forma en que algunos proyectos declaran sus dependencias.
3. Filtrado de dependencias inválidas o huérfanas, es decir, aquellas que refieren a paquetes que no existen en el índice.
4. Construcción final de una lista consistente de pares (`paquete` $\rightarrow$ `dependencia`), apta para su modelado como grafo dirigido.

Este proceso garantizó que ambos datasets fueran comparables entre sí y que la red obtenida reflejara con la mayor fidelidad posible la estructura real del ecosistema de librerías de Python para cada año analizado. El código utilizado para la limpieza de datos forma parte del repositorio del proyecto mencionado anteriormente, mientras que los datasets finales se encuentran depositados en Zenodo (Catania et al., 2025).

3.3. Construcción del grafo de dependencias y limitaciones

A partir de los datos preprocesados, construimos un grafo dirigido para cada año utilizando la biblioteca NetworkX, donde cada nodo representa un paquete de Python y cada arista dirigida $(u \rightarrow v)$ indica que el paquete u declara a v como dependencia. Las aristas son no ponderadas, dado que el análisis se centra en la estructura topológica de la red y no en la frecuencia o intensidad de las dependencias. Los grafos resultantes para 2016 y 2025 están depositados en Zenodo (Catania et al., 2025).

Este trabajo presenta algunas limitaciones que deben considerarse al interpretar los resultados. En primer lugar, el dataset de 2025 se construyó a partir de metadatos de PyPI, por lo que puede incluir sesgos propios del índice (paquetes desactualizados, metadatos incompletos o dependencias opcionales no declaradas). En segundo lugar, la comparación temporal se realiza entre dos cortes puntuales (2016 y 2025), lo que impide capturar dinámicas anuales o efectos de corto plazo.

3.4. Métricas de red consideradas

Sea $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ un grafo dirigido que representa la red de dependencias, donde $\mathcal{V}$ es el conjunto de nodos (paquetes) y $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ el conjunto de aristas dirigidas (relaciones de dependencia). Para caracterizar su estructura se calcularon las siguientes métricas.

Grado. Para cada nodo $v \in \mathcal{V}$, el grado de entrada $k_{in}(v)$ se define como el número de aristas que apuntan a v, mientras que el grado de salida $k_{out}(v)$ corresponde al número de aristas que parten de v. El grado total se define como $k(v) = k_{in}(v) + k_{out}(v)$.

Betweenness centrality. Mide la fracción de caminos mínimos entre pares de nodos que atraviesan un nodo $v \in \mathcal{V}$: $C_B(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$, donde σ_{st} es el número total de caminos mínimos entre s y t, y $\sigma_{st}(v)$ aquellos que pasan por v.

PageRank. Se utilizó el algoritmo PageRank (Page et al., 1999), que asigna a cada nodo un puntaje definido recursivamente como

$$PR(v) = \frac{1-d}{|\,\mathcal{V}\,|} + d \cdot \sum_{u \in \mathcal{V}_{in}(v)} \frac{PR(u)}{k_{out}(u)},$$

donde d es el factor de amortiguación (usualmente $d = 0.85$) y $\mathcal{V}_{in}(v)$ el conjunto de nodos que apuntan a v.

Modularidad y detección de comunidades. La estructura comunitaria se detectó mediante el algoritmo de Louvain (Blondel et al., 2008), que maximiza la modularidad Q, definida como

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j),$$

donde A_{ij} es la matriz de adyacencia, $m = |\,\mathcal{E}\,|$, k_i es el grado del nodo i, y $\delta(c_i, c_j)$ es 1 si los nodos pertenecen a la misma comunidad y 0 en caso contrario. Todas las métricas fueron calculadas utilizando la biblioteca NetworkX en Python.

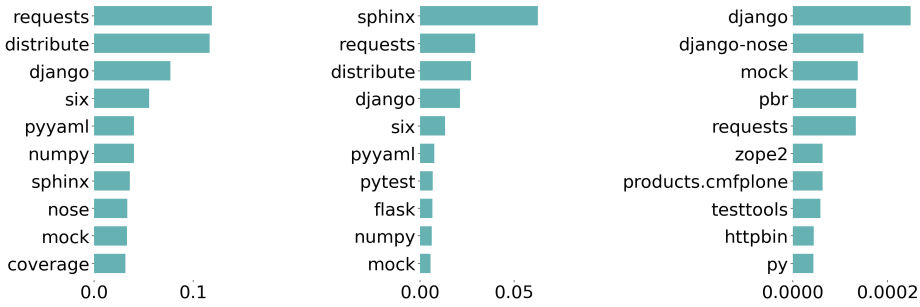
4. Propiedades estructurales del ecosistema en 2016

En esta sección analizamos la estructura del grafo desde propiedades globales hasta organización comunitaria. El grafo, al que se hará referencia como $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, cuenta con $|\,\mathcal{V}\,| = 25,819$ nodos y $|\,\mathcal{E}\,| = 72,189$ aristas dirigidas. El grado de entrada y de salida promedio son idénticos, ambos con un valor aproximado de 2.79, lo que deja un grado promedio total cercano a $\langle\, k \,\rangle \simeq 5.59$.

4.1. Distribución de centralidad

Examinamos $\mathcal{G}$ con mayor detalle mediante las tres métricas de centralidad mencionadas en la sección de metodología. Los resultados de los diez nodos más destacados en cada métrica se muestran en la Figura 1.

Como primera observación, se identifica que la librería requests aparece de forma recurrente en el top 5 de todas las medidas, lo que la posiciona como un nodo altamente relevante desde múltiples perspectivas. El hecho que requests



(a) Degree centrality (b) PageRank (c) Betweenness centrality

Figura 1. Distribución de las principales métricas de centralidad en el grafo de dependencias de 2016. Se muestran los diez paquetes con mayor valor según cada métrica. La recurrencia de ciertos paquetes entre métricas indica la existencia de nodos estructuralmente dominantes que cumplen roles centrales en la organización global de la red.

ocupe posiciones tan altas en *degree centrality* y *PageRank* indica que es apuntada por numerosos nodos, que se ubica estructuralmente cerca de gran parte del grafo y que recibe enlaces desde otros nodos importantes. Además, su presencia en *PageRank* en el puesto número 2 refuerza la idea de que su rol no se debe únicamente a la cantidad de dependencias – sino que también es central dentro de la jerarquía del sistema.

Los resultados muestran que tanto en *Degree centrality* y *PageRank*, hay varios de los paquetes que se repiten en el top. Sin embargo, al observar *Betweenness centrality*, el panorama cambia: muchas de las librerías destacadas en esta métrica no aparecen en los rankings previos. Esto es esperable, dado que la métrica captura nodos que, sin tener grandes grados o posiciones privilegiadas en términos de cercanía, resultan indispensables para conectar distintas regiones del grafo.

A nivel centralidad, se distinguen nodos que son importantes por su visibilidad global (como requests y distribute) y otros cuya relevancia se encuentra relacionada con su función de "articulación" entre grupos (como mock, y pbr). Esta diversidad de roles estructurales es consistente con las dinámicas típicas de redes tecnológicas amplias y descentralizadas como la del ecosistema Python.

4.2. Estructura comunitaria

Se analizó la red completa para identificar comunidades utilizando el algoritmo de Louvain buscando maximizar la modularidad (Blondel et al., 2008). Este paso permite conocer cuántas comunidades existen, el tamaño de cada una y la importancia relativa de cada nodo mediante el cálculo de grado y *PageRank*. Aquí se encontraron 639 comunidades, teniendo la comunidad más grande 5840 nodos. La modularidad de esa división fue de 0.615.

Para poder graficar y hacer un análisis más detallado, se crearon subgrafos con los nodos más importantes según *PageRank*, permitiendo analizar la estructura de la red focalizándose en los nodos centrales. Podemos ver uno de estos gráficos en la Figura 2.

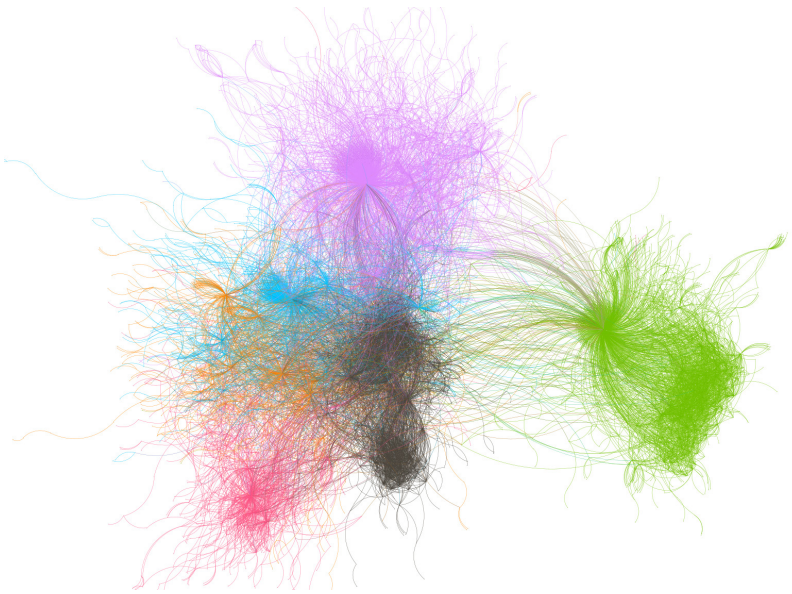


Figura 2. Subgrafo del ecosistema 2016 a partir de los nodos de mayor *PageRank*, filtrando por la componente conexa más grande y las 6 comunidades más grandes de ese subgrafo. El resultado tiene $6,548$ nodos y $16,244$ aristas. Se puede ver que la mayoría de las comunidades se encuentran bien separadas, con poca superposición entre ellas.

4.3. Caracterización semántica de comunidades

Para poder comprender mejor la composición de cada comunidad, decidimos estudiar las descripciones textuales de los paquetes que las componen. El texto se procesó eliminando palabras comunes o irrelevantes con el fin de identificar las más descriptivas de cada comunidad. Paralelamente, evaluamos la relevancia de cada nodo mediante su valor de *PageRank*. Esta combinación permite comprender los temas predominantes de cada comunidad.

Analizando las 10 comunidades más grandes, podemos ver que la mayoría tienen un enfoque en herramientas web. En la Figura 3 vemos que la primer comunidad es liderada por distribute y contiene muchos paquetes de bajo *PageRank*. La segunda comunidad está liderada por el paquete requests y son herramientas de parseado y APIs. Es interesante destacar que las librerías científicas recién aparecen como séptima comunidad más grande en el 2016.

(a) Primer comunidad más grande (b) Segunda comunidad más grande

Figura 3. Wordclouds de las descripciones breves de los paquetes en las dos comunidades más grandes en cantidad de paquetes del grafo de 2016. Barplots con los nodos pertenecientes a cada comunidad ordenados según su valor de PageRank en la red completa. Se ve una fuerte presencias de herramientas de desarrollo web.

5. Propiedades estructurales del ecosistema en 2025

Para el dataset correspondiente al año 2025 se repitió la estrategia inicial de exploración, aunque con ciertos ajustes necesarios debido al tamaño significativamente mayor del conjunto de datos. A diferencia del grafo correspondiente a 2016, la red de 2025 no solo presenta un aumento significativo en tamaño, sino también cambios estructurales relevantes en términos de densidad, centralidad y organización comunitaria, que se analizan a continuación.

El grafo construido $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ es un grafo dirigido compuesto por $|\mathcal{V}'| = 405,872$ nodos y $|\mathcal{E}'| = 2,076,516$ aristas. El análisis de sus grados revela que tanto el in-degree como el out-degree presentan un promedio cercano a 5.11, lo que sitúa el grado total medio alrededor de $\langle k \rangle \simeq 10.23$.

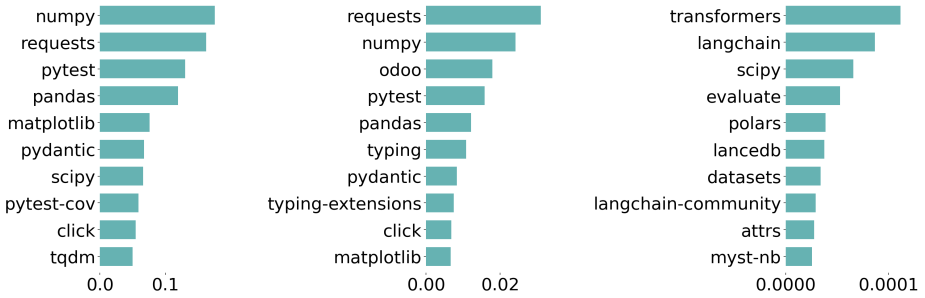
5.1. Distribución de centralidad

Al momento de calcular las métricas de centralidad, no fue posible computar directamente la *betweenness centrality* debido a la alta complejidad computacional del algoritmo y el dataset significativamente mayor. Por eso, optamos por estimarla mediante una aproximación basada en un muestreo aleatorio.

En la Figura 4 se ilustran los nodos más relevantes según las métricas consideradas. Vemos que requests vuelve a posicionarse entre los paquetes más influyentes del grafo, reflejando su uso sostenido dentro del ecosistema y su rol central en proyectos que requieren comunicación HTTP.

Emerge una librería entre las más influyentes llamada pytest, que es una herramienta ampliamente utilizada para la automatización de pruebas. Su ubicación en los primeros puestos refleja la adopción masiva de prácticas de testing en el desarrollo de software en los últimos tiempos.

Otro nombre que adquiere relevancia es pandas, biblioteca fundamental para el análisis de datos. Su presencia entre los nodos con mayor centralidad es coherente con el crecimiento exponencial de las tareas de procesamiento de datos en la



(a) Degree centrality (b) PageRank (c) Betweenness centrality

Figura 4. Distribución de las principales métricas de centralidad en el grafo de dependencias de 2025. Se muestran los diez nodos con mayor valor para cada métrica.

comunidad Python. Por último, la librería NumPy también aparece entre las más influyentes. Es una biblioteca central del ecosistema científico de Python que provee estructuras de datos optimizadas para el cálculo numérico, en particular el arreglo multidimensional. NumPy constituye la base sobre la cual se desarrollan numerosas bibliotecas orientadas a análisis de datos, modelado matemático y aprendizaje automático.

Como en 2016, los resultados muestran similitudes entre los paquetes en *degree centrality* y *PageRank*, mientras que en *degree centrality* aparecen otros paquetes. Esto distingue entre nodos que son importantes por su visibilidad global y otros cuya relevancia se encuentra relacionada con su función de "articulación" entre grupos. En conjunto, estos resultados muestran que, si bien ciertos paquetes mantienen su predominio histórico, el grafo de 2025 revela un escenario donde nuevas áreas de interés —como el testing automatizado y el análisis de datos— adquieren un peso estructural considerable dentro de la red.

5.2. Estructura comunitaria

Se realizó un análisis similar al hecho con la red anterior pero con resultados muy distintos. Se encontraron $3,829$ comunidades, la más grande con $116,312$ nodos y la modularidad cayó notablemente a 0.426. En la Figura 5 podemos ver como quedó uno de los gráficos hechos sobre este dataset.

De estas imágenes podemos entender mejor la caída de la modularidad ya que se ve que las líneas divisorias de las comunidades son mucho más difusas. El nuevo dataset tiene muchas más aristas, lo que complejiza la separación. Por ejemplo, en el subgrafo que contiene los $10,000$ nodos de mayor *PageRank*, el subgrafo de 2016 (sin filtrar) tiene $24,915$ enlaces, mientras que el subgrafo de 2025 contiene $50,823$ enlaces.

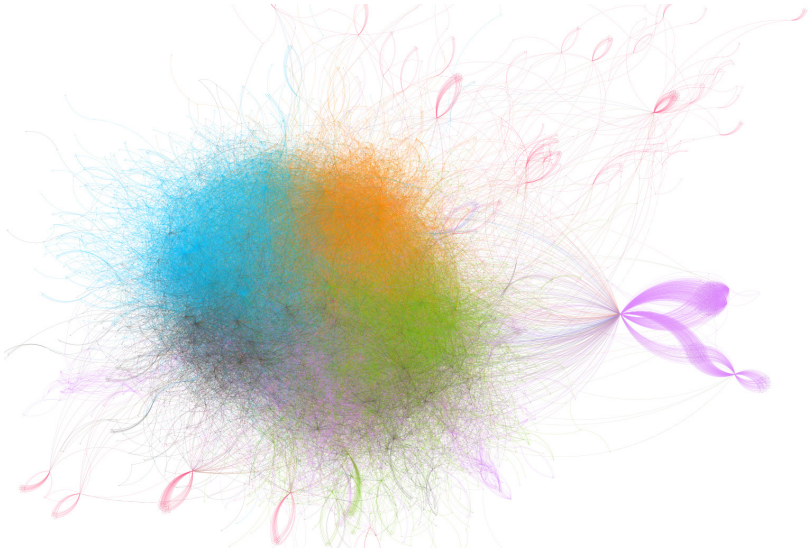
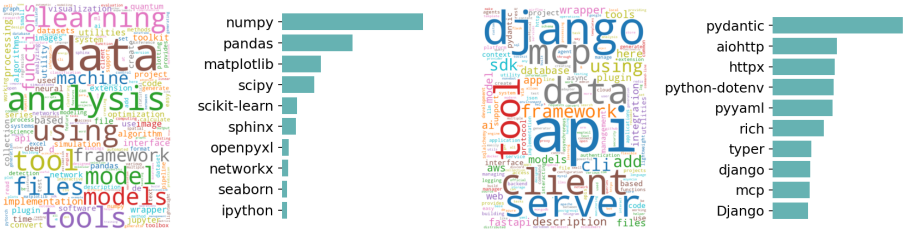


Figura 5. Subgrafo del ecosistema 2025 a partir de los nodos de mayor *PageRank*, filtrando por la componente conexa más grande y las 6 comunidades más grandes de ese subgrafo. El resultado tiene 7,899 nodos y 41,318 aristas. El aumento de densidad y la difuminación de fronteras entre comunidades evidencian una reducción de modularidad y una mayor interconectividad global en comparación con la red de 2016.



(a) Primer comunidad más grande (b) Segunda comunidad más grande

Figura 6. Wordclouds de las descripciones breves de los paquetes en cada una de las dos comunidades más grande en cantidad de paquetes del grafo de 2025. Barplots con los nodos pertenecientes a cada comunidad ordenados según su valor de PageRank en la red completa.

5.3. Caracterización semántica de comunidades

Al comparar los wordclouds de la red de 2016 con la red de 2025 se ven cambios significativos. En la Figura 6 vemos que la comunidad de paquetes científicos (numpy, pandas, matplotlib, scipy, scikit-learn) pasó de ser la séptima comunidad más grande a ser la primera. Pero además, la sexta comunidad más grande es más específica de machine learning e inteligencia artificial (torch, openai, tensorflow, transformers). Es llamativo que en esta comunidad aparezca tqdm, una librería para mostrar barras de progreso, dando a entender que muchos de estos paquetes tienen tiempos de ejecución largos. Por otro lado, los paquetes relacionados a desarrollo web han perdido centralidad: recién en quinto lugar aparece la comunidad con paquetes de parsers (requests, beautifulsoup4, lxml). Por último, mirando a nivel general, podemos ver que cuando antes las palabras más comunes eran API, django, client; ahora son data, API, y tool.

6. Dinámica evolutiva del ecosistema

Con el objetivo de caracterizar la evolución estructural del ecosistema de dependencias de Python, comparamos métricas globales de los grafos correspondientes a 2016 y 2025 (ver Cuadro 1). Esta comparación permite analizar no solo el crecimiento cuantitativo del sistema, sino también cambios cualitativos en su organización topológica.

Cuadro 1. Métricas globales del grafo de dependencias por año.

Métrica	2016	2025
Nodos	25,819	405,872
Aristas	72,189	2,076,516
Grado promedio	5.59	10.23
Modularidad	0.615	0.426

6.1. Ley de densificación

En la evolución del ecosistema se observa un crecimiento masivo: el número de nodos aumenta más de un orden de magnitud y el número de aristas crece aún más rápidamente. Este patrón sugiere un proceso de densificación, fenómeno característico de redes reales en crecimiento, en el cual el número de enlaces aumenta superlinealmente respecto al número de nodos.

Para evaluar si el crecimiento del ecosistema sigue patrones conocidos de redes reales, analizamos la relación entre el número de aristas y el número de nodos en ambos snapshots. En muchas redes complejas se observa una ley de densificación de la forma $|\mathcal{E}(t)| \sim |\mathcal{V}(t)|^{\alpha}$, donde $\alpha > 1$ indica crecimiento

superlineal. A partir de los valores observados, el exponente puede estimarse como

$$\alpha = \frac{\log |\, \mathcal{E}_{2025} \,| - \log |\, \mathcal{E}_{2016} \,|}{\log |\, \mathcal{V}_{2025} \,| - \log |\, \mathcal{V}_{2016} \,|}.$$

El exponente de densificación estimado es $\alpha \approx 1.22$, lo que indica crecimiento superlineal del número de aristas respecto del número de nodos. Este resultado es consistente con patrones observados en redes tecnológicas y de información previamente estudiadas, donde valores $\alpha > 1$ reflejan procesos de integración estructural progresiva.

6.2. Interdependencia y reorganización funcional

El incremento del grado promedio indica que los paquetes modernos tienden a depender de más librerías que en etapas anteriores del ecosistema. Este resultado sugiere una mayor reutilización de componentes y una integración más profunda entre módulos, lo que implica un aumento del acoplamiento estructural.

Desde el punto de vista mesoscópico, la disminución de la modularidad observada en 2025 indica que las fronteras entre comunidades se vuelven menos definidas. Este comportamiento es consistente con un proceso de integración progresiva entre dominios funcionales previamente más separados, como desarrollo web, computación científica y herramientas de testing.

Finalmente, el análisis de centralidad muestra una reconfiguración de los nodos estructuralmente dominantes. Mientras que en 2016 predominaban librerías asociadas a compatibilidad y desarrollo web, en 2025 emergen como hubs centrales paquetes vinculados a ciencia de datos, tipado estático y testing automatizado. Este desplazamiento refleja cambios profundos en las prácticas de desarrollo y en los dominios de aplicación predominantes del lenguaje.

En conjunto, estos resultados sugieren que el ecosistema Python no solo crece en tamaño, sino que atraviesa una transformación estructural sistemática caracterizada por densificación, mayor interdependencia y reorganización funcional de sus nodos centrales. Es importante notar que librerías altamente populares como PyTorch no aparecen como nodos centrales en la red de dependencias, lo que refleja la diferencia entre popularidad de uso y centralidad estructural.

7. Conclusiones

En este trabajo realizamos un análisis longitudinal del ecosistema de dependencias de librerías de Python mediante herramientas de ciencia de redes, comparando dos instantáneas separadas por casi una década. Los resultados muestran que el sistema experimenta un crecimiento masivo acompañado por transformaciones estructurales significativas.

En primer lugar, el incremento simultáneo en número de nodos, número de aristas y grado promedio indica un proceso de densificación progresiva, patrón característico de redes tecnológicas en expansión. Este comportamiento sugiere

que el crecimiento del ecosistema no es meramente cuantitativo, sino que implica una integración cada vez mayor entre sus componentes.

En segundo lugar, la reducción de modularidad observada en el grafo más reciente evidencia una transición hacia una arquitectura más interconectada, en la cual las fronteras entre comunidades funcionales se vuelven menos definidas. Este resultado es consistente con la convergencia entre dominios tecnológicos y la reutilización transversal de librerías.

Asimismo, el análisis de centralidad revela una reconfiguración del núcleo estructural del ecosistema. Mientras que en etapas tempranas predominaban librerías asociadas al desarrollo web y compatibilidad, en el periodo más reciente emergen como nodos centrales paquetes vinculados a ciencia de datos, computación científica, tipado estático y testing automatizado. Este desplazamiento refleja cambios profundos en las prácticas de desarrollo y en los usos predominantes del lenguaje, y sugiere que la red de dependencias actúa como un registro macroscópico de dichas transformaciones.

En conjunto, estos hallazgos muestran que el ecosistema Python constituye un sistema dinámico cuya evolución sigue patrones estructurales típicos de redes complejas en crecimiento, incluyendo densificación, reorganización de hubs y aumento del acoplamiento global, consistentes con las leyes de densificación y reducción del diámetro efectivo documentadas en grafos empíricos (Leskovec et al., 2007). Desde una perspectiva más amplia, este estudio aporta evidencia empírica de que los ecosistemas de dependencias de software pueden analizarse como infraestructuras tecnológicas complejas sujetas a regularidades evolutivas.

Referencias

Barabási, A.-L., & Albert, R. (1999). Emergence of Scaling in Random Networks. *Science*, *286*(5439), 509-512. https://doi.org/10.1126/science.286.5439.509

Blondel, V. D., Guillaume, J.-L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, P10008. https://doi.org/10.1088/1742-5468/2008/10/P10008

Bogart, C., Kästner, C., Herbsleb, J., & Thung, F. (2016). How to Break an API: Cost Negotiation and Community Values in Three Software Ecosystems. *Proceedings of the 24th ACM SIGSOFT Symposium on Foundations of Software Engineering*, 109-120.

Catania, J. I., Pérez, M. R., Guerrero Schmidt, M., Gutierrez, S., & Sarraute, C. (2025). *Python Package Dependency Network Dataset (2016–2025)*. Zenodo. https://doi.org/10.5281/zenodo.18912507

Decan, A., Mens, T., & Claes, M. (2019). An Empirical Comparison of Dependency Network Evolution in Seven Software Packaging Ecosystems. *Empirical Software Engineering*, *24*, 381-416. https://doi.org/10.1007/s10664-018-9630-5

Gullikson, K. (2016). *Python dependency analysis*. Consultado el 28 de febrero de 2026, desde https://kgullikson88.github.io/blog/pypi-analysis.html

Kikas, R., Gousios, G., Dumas, M., & Pfahl, D. (2017). Structure and Evolution of Package Dependency Networks. *Proceedings of the 14th International Conference on Mining Software Repositories*, 102-112. https://doi.org/10.1109/MSR.2017.55

Leskovec, J., Kleinberg, J., & Faloutsos, C. (2007). Graph Evolution: Densification and Shrinking Diameters. *ACM Transactions on Knowledge Discovery from Data*, *1*(1), 2. https://doi.org/10.1145/1217299.1217301

Mora-Cantallops, M., Sánchez-Alonso, S., & García-Barriocanal, E. (2020). A complex network analysis of the Comprehensive R Archive Network (CRAN) package ecosystem. *Journal of Systems and Software*, *170*, 110744. https://doi.org/https://doi.org/10.1016/j.jss.2020.110744

Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The PageRank citation ranking: Bringing order to the web* (inf. téc.). Stanford InfoLab. Consultado el 28 de febrero de 2026, desde http://ilpubs.stanford.edu:8090/422/

Salgado, A., & Caridi, I. (2022). Network of R packages: A characterization of an empirical collaborative network. *Chaos, Solitons & Fractals*, *155*, 111756. https://doi.org/https://doi.org/10.1016/j.chaos.2021.111756

Valverde, S., Ferrer-Cancho, R., & Solé, R. V. (2005). Scale-Free Networks from Optimal Design. *Europhysics Letters*, *60*(4), 512-517.

Valverde, S., & Solé, R. V. (2003). Hierarchical Small Worlds in Software Architecture. *arXiv preprint cond-mat/0307278*.

Zimmermann, M., Staicu, C.-A., Tenny, C., & Pradel, M. (2019). Small World with High Risks: A Study of Security Threats in the npm Ecosystem. *Proceedings of the 28th USENIX Security Symposium*, 995-1010.