Carlos Javier Asensio Alloza
-Grado de Física-

Análisis de las Materias de Especialización de Comunidades Científicas mediante Redes Complejas

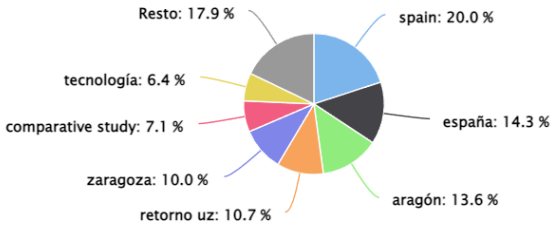
Directores: Alfonso Tarancón Lafita y David Íñiguez

Contents

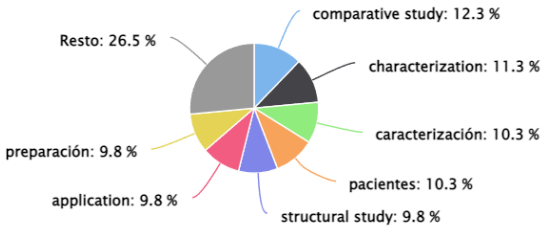
1 Presentación

Una de las maneras que tenemos de saber a qué se dedica un investigador, más allá de conocer su departamento (el cual puede darnos información demasiado general) es a través de las palabras clave (keywords) que aparezcan en sus trabajos. A veces estas keywords las aportan los propios autores en sus papers, pero también pueden extraerse a partir de sus publicaciones.

El objetivo de este trabajo es asignar keywords a varias comunidades de autores de papers científicos dentro de una misma red de coautoría. Aunque se espera que los resultados sean generalizables a distintas redes, incluso de otros ámbitos, nos centraremos en la red de autores de la Facultad de Ciencias de la Universidad de Zaragoza. Actualmente la herramienta Kampal Research, desarrollada por la spin-off de la Universidad de Zaragoza Kampal Data Solutions, se ocupa de esta tarea mediante un método que no tiene en cuenta la estructura de la red, con resultados mixtos (demasiadas palabras genéricas o no muy representativas de la comunidad) que trataremos de mejorar. Por ejemplo[1]:



a) Comunidad del área de química inorgánica



(b) Comunidad del área de química orgánica

Figure 1: Keywords asignadas a las dos mayores comunidades de autores, por Kampal

Dicho de otro modo: partiendo de las keywords de cada autor dentro de una comunidad, que está inmersa en una red más grande con varias comunidades, queremos asignar keywords a cada comunidad que sean representativas del trabajo que llevan a cabo sus autores, independientemente de las particularidades de cada uno.

Para ello se han probado dos algoritmos distintos además de un método análogo al que utiliza Kampal, habiendo mejorado la asignación de keywords además de ofrecer mayor flexibilidad respecto al nivel de agregación que se desea (keywords más generales o más concretas) o la importancia de los autores con mayor grado dentro de la red.

[1]Por motivos de privacidad vamos a omitir usar nombres propios en este trabajo. Nótese que cuando usemos el término "Comunidad del área X" no nos referimos a todos los autores de esa área, sino a una subcomunidad de dicha área.

2 Planteamiento del problema

Para enfocar este problema partiremos de unos datos muy similares a los de Kampal Research, que consisten por un lado en la estructura de la red y por otro en las keywords de los autores. Todo el análisis lo haremos con estos mismos datos en vez de con las comunidades de Kampal para que no haya distorsiones como una distinta clusterización de la red o un fichero de keywords desactualizado.

2.1 Estructura de la red

Trabajaremos con una red en la que cada nodo representará a un autor, el peso de los enlaces (links) entre dos nodos estará asociado a los papers que esos dos autores hayan publicado conjuntamente, de tal forma que el peso del link será mayor cuanto más hayan colaborado tanto en artículos como en proyectos, y cuanto más impacto estos trabajos hayan tenido.
Más concretamente se asignan 4 puntos si la revista en la que el artículo ha sido publicado está en el primer cuartil de relevancia, 3 para el segundo, 2 para el tercero y 1 para el último, mientras que para los proyectos se utilizará una magnitud proporcional a los fondos de los mismos. Este valor se divide entre el número total de autores. Sumamos para cada pareja de autores todos los puntos que tengan y formamos la matriz de adyacencia $\mathbf{W}$ que será simétrica (i.e. la red será no dirigida) y no presentará autoloops (i.e. $W_{ii} = 0\,\forall i$).

Antes de exponer los algoritmos que se correrán sobre la red conviene estudiarla un poco.

Una característica muy común en las redes complejas es que estén libres de escala, lo cual significa que su distribución de grado sigue una ley de potencias tal que[2] $p(k) \propto k^{-\gamma}$ donde γ es un parámetro característico de la red, que típicamente cumple $2 < \gamma < 3$.

Aunque nuestra red es relativamente pequeña (N=2830), si calculamos el parámetro γ a partir de un histograma de la distribución de grados (normalizado para que su área valga 1) obtenemos que $\gamma = 2.62 \pm 0.06$:

[2]Ver que en el límite contínuo $p(k) = (\gamma - 1)k_{min}^{\gamma-1}k^{\gamma-1}$

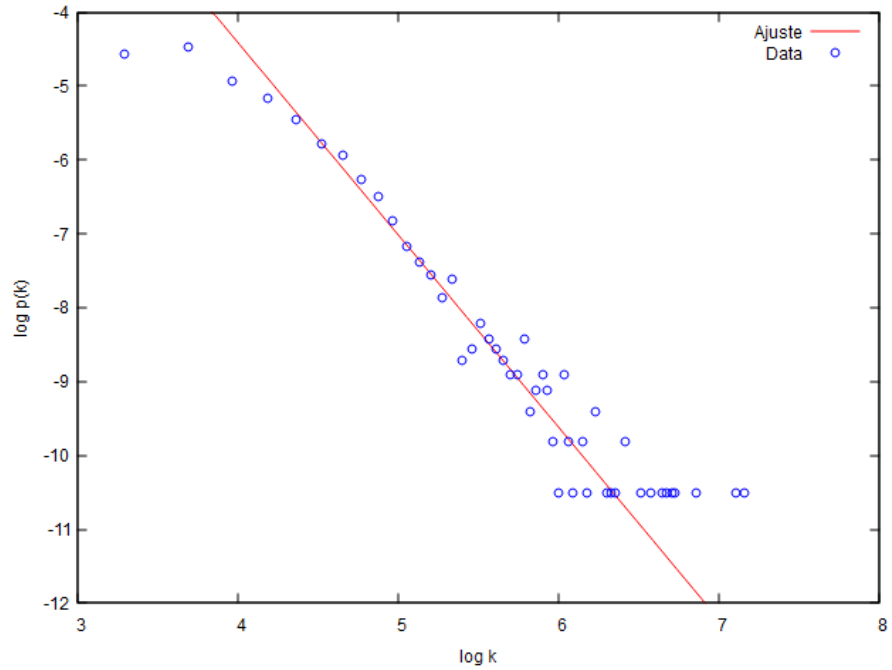


Figure 2: Estimación de γ según $p(k) = k^{-\gamma}$

Obtenemos un valor totalmente compatible con el intervalo de $\gamma \in (2, 3)$ característico de este tipo de redes (Barabasi et al. 2001).

Para la clusterización utilizaremos el algoritmo walktrap (Pons and Latapy 2005) teniendo en cuenta los pesos de los links (interpretamos que a mayor peso más corto es el camino entre dos nodos). Es mucho más rápido que otros algoritmos como edge betweenness y ha arrojado mejores resultados que spinglass, fastgreedy o infomap.

Walktrap identifica comunidades mediante paseos aleatorios que utiliza para calcular la distancia entre nodos. Su mayor inconveniente es que clasifica los nodos en comunidades de intersección nula, sin embargo no vamos a dar demasiada importancia a este problema porque Kampal Research utiliza algoritmos de clusterización similares, con comunidades que no se superponen. Dicho esto, los algoritmos propuestos deberían funcionar igual de bien en ambos casos ya que no dependen de la clusterización de la red. Una vez obtenidas las comunidades, nombraríamos a cada una por su autor con mayor grado, pero por motivos de privacidad vamos a omitirlo en este trabajo. Representando el grafo, con el tamaño de los nodos proporcional a la raíz cuadrada del grado y el grosor de los links proporiconal al peso:

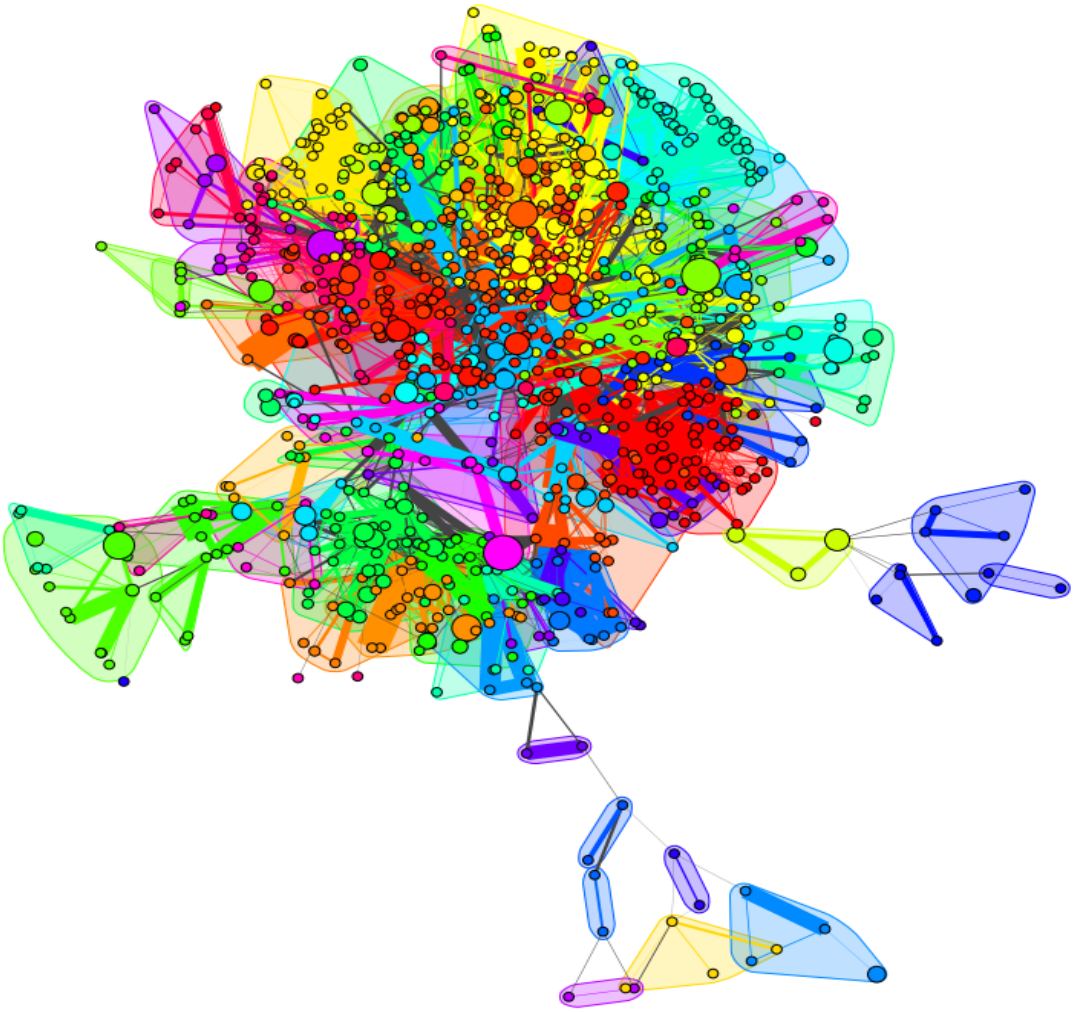


Figure 3: Red de coautoría segmentada por comunidades (diferenciadas por colores)

Si ordenamos las comunidades de mayor a menor número de autores observamos que nuestra red presenta una alta heterogeneidad, tal y como era esperable:

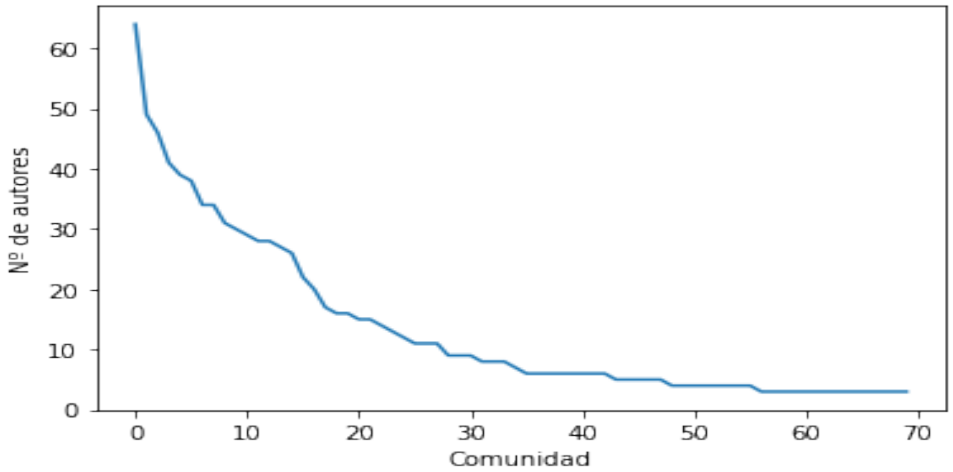


Figure 4: Tamaño de las comunidades, ordenadas de mayor a menor

2.2 Asignación de keywords

Respecto a las keywords de cada autor, éstas se han extraído de los títulos de sus papers recientes. Para ello se utilizan "stop words" (palabras genrealmente con sólo valor gramatical y no semántico) y signos de puntuación para seleccionar posibles candidatos. Posteriormente aplicamos la fórmula TF-IDF para filtrarlas por importancia (Qaiser and Ali 2018) y tratamos de filtrar manualmente aquellas keywords muy genéricas como "data" o "análisis".

No disponemos de un dato cuantitativo que indique el peso de cada keyword dentro de un mismo autor, pero sí sabemos que están ordenadas de tal forma que la primera es más relevante que la segunda, etc. Un detalle importante es que el número de keywords por autor es muy variable, de hecho su distribución decae muy rápidamente, pero presenta una cola larga que no podemos obviar:

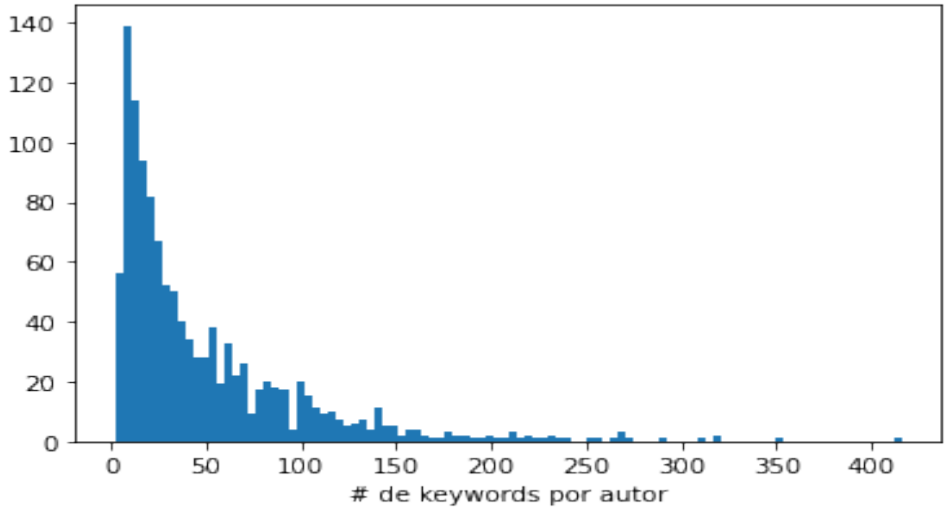


Figure 5: Histograma del número de keywords por autor

Explicaremos la relevancia de la distribución de keywords y su peso dentro de cada autor en la siguiente sección.

3 Algoritmos propuestos

Aunque es cierto que podríamos mejorar la asignación de keywords a comunidades científicas sin trabajar directamente sobre la red de autoría, por ejemplo ponderando las keywords de los autores por su impacto, como hemos visto en la sección anterior estamos ante una red compleja, que se caracteriza por una alta no linealidad, lo que nos impulsa a atacarlo teniendo en cuenta toda la estructura de la red.

En concreto, usaremos algoritmos basados en la propagación de etiquetas. Una "etiqueta" es un valor que puede estar asociado a un nodo o un link. En nuestro caso asociaremos las keywords a cada autor, que son los nodos de la red. Normalmente estos algoritmos se uti-

lizan en algoritmos supervisados de machine learning, etiquetando manualmente algunos de los nodos que propagarán su etiqueta a nodos sin etiquetar. Este problema es algo distinto al nuestro, en el que todos los nodos tienen alguna etiqueta y ningún nodo está privilegiado sobre cualquier otro teniendo sus etiquetas fijas de antemano, como ocurre con estos algoritmos.

Uno de los trabajos más relevantes en este campo es Zhu and Ghahramani 2002. Para el algoritmo propuesto, los autores demuestran la convergencia del mismo sólo en el caso en el que estrictamente hay nodos etiquetados (cuya etiqueta además no puede cambiar) y sin etiquetar, además de trabajar sólo con una etiqueta booleana, aunque posteriores trabajos han relajado las condiciones de la convergencia. En este trabajo aplicaremos casi directamente uno de estos algoritmos, sin embargo, aunque este converja no podemos garantizar que lo haga hacia una solución satisfactoria, por lo que también probaremos una variante propia con la idea de adaptarse a las circunstancias particulares de nuestro problema, esto es, tener todos los nodos etiquetados mientras que cada nodo puede tener cientos de etiquetas simultaneamente.

3.1 LPA: Label Propagation Algorithm

El primero de nuestros algoritmos está basado en el trabajo de Zhou et al. 2004. Sea un conjunto de n autores $\mathbf{X} = \{x_1, x_2, \ldots, x_n\}$ y un conjunto de c keywords $\mathbf{L} = \{l_1, l_2, \ldots, l_c\}$, los elementos de $\mathbf{X}$ tendrán asociadas keywords dadas por:

$$\mathbf{Y} = \{\overline{y}(x_1), \overline{y}(x_2), \ldots, \overline{y}(x_n) | \overline{y}(x_i) = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_c \end{pmatrix}, \lambda_{i,j} \in [0,1] \ \forall i_{\{1,\ldots,n\}}, j_{\{1,\ldots,c\}}\} \qquad (1)$$

Esto es, el elemento de matriz Y_{ij} nos indica la intensidad con la que el i-ésimo autor tiene asociada la j-ésima keyword, con un valor entre 0 y 1. La red vendrá caracterizada por la matriz de adyacencia $\mathbf{W}$ vista en el apartado anterior que recordemos no posee autoloops (i.e. $W_{ii} = 0 \, \forall i_{\{1,\ldots,n\}}$), con una matriz de grado $\mathbf{D}$ cuyos elementos cumplen $\mathbf{D}_{ij} = \delta_{ij} \sum_i W_{ij} = \delta_{ij} \sum_j W_{ij}$ y δ_{ij} es la delta de Kronecker. Denotaremos por $\mathbf{Y}_0$ a la matriz de condiciones iniciales, es decir, aquella cuyos vectores $\mathbf{Y}$ se dan como input.

Definimos la siguiente función de coste:

$$C(\mathbf{Y}) = \frac{1}{2} \sum_{i,j=1}^{n} W_{ij} \left\| \frac{Y_i}{\sqrt{D_{ii}}} - \frac{Y_j}{\sqrt{D_{jj}}} \right\|^2 + \left(\frac{1}{\alpha} - 1 \right) \sum_{i=1}^{n} ||\mathbf{Y}_i - \mathbf{Y}_{0,i}||^2, \ \ \alpha \in (0,1] \qquad (2)$$

El primer término de la función de coste corresponde a la restricción de continuidad suave, esto es, queremos que nodos muy cercanos tengan unas etiquetas similares y la transición no sea brusca, mientras que el segundo corresponde a la restricción de condiciones iniciales, esto es, un autor al finalizar el algoritmo no debería tener unas etiquetas radicalmente distintas a las que tenía inicialmente. Para controlar este término está el parámetro

α. Ver que si $\alpha = 1$ este término desaparece y las condiciones iniciales dejan de ser una restricción, mientras que si $\alpha \to 0^+$ el término tendrá cada vez más peso.

Para demostrar la convergencia, optimizaremos esta función de coste. Sea $\mathbf{Y}^*$ la matriz que minimiza el coste y $\mathbf{S}$ la matriz de adyacencia normalizada tal que $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$:

$$\frac{\partial C}{\partial Y}\Big|_{\mathbf{Y}=\mathbf{Y}*} = \mathbf{Y}^* - \mathbf{S}\mathbf{Y}^* + \left(\frac{1}{\alpha} - 1\right)(\mathbf{Y}^* - \mathbf{Y_0}) = 0 \tag{3}$$

Operando se tiene inmediatamente que:

$$(\mathbf{I} - \alpha\mathbf{S})\mathbf{Y}^* = (1 - \alpha)\mathbf{Y_0} \tag{4}$$

Como el término $\mathbf{I} - \alpha\mathbf{S}$ es invertible tendremos:

$$\mathbf{Y}^* = (1 - \alpha)(\mathbf{I} - \alpha\mathbf{S})^{-1}\mathbf{Y_0} \tag{5}$$

O escrito en forma iterativa sin perder generalidad:

$$\mathbf{Y}_{t+1} = \alpha\mathbf{S}\mathbf{Y}_t + (1 - \alpha)\mathbf{Y_0} \tag{6}$$

Ver que esta recursividad la podemos escribir como:

$$\mathbf{Y}(t) = (\alpha\mathbf{S})^{t-1}\mathbf{Y}_0 + (1 - \alpha)\sum_{i=0}^{t-1}(\alpha\mathbf{S})^i\mathbf{Y}_0 \tag{7}$$

Como $0 < \alpha \leq 1$ y los autovalers de $\mathbf{S}$ sólo pueden ir desde $[-1, 1]$ entonces se cumplirá que:

$$\begin{cases} \lim_{t\to\infty}(\alpha\mathbf{S})^{t-1} = 0 \\ \lim_{t\to\infty}\sum_{i=0}^{t-1}(\alpha\mathbf{S})^i = (\mathbf{I} - \alpha\mathbf{S})^{-1} \end{cases} \tag{8}$$

Luego:

$$\mathbf{Y}^* = \lim_{t\to\infty}\mathbf{Y}(t) = (1 - \alpha)(\mathbf{I} - \alpha\mathbf{S})^{-1}\mathbf{Y}_0 \tag{9}$$

Ver que hemos recuperado la expresión (5), luego está demostrado que nuestra solución converge y además podemos escribirla de forma iterativa. De esta forma tenemos una recursión que podemos iterar indefinidamente porque conocemos $\mathbf{Y}_{t=0}$. Ver además como α tiene el significado físico que hemos comentado, pues si $\alpha = 1$ el término de $\mathbf{Y_0}$ valdrá cero.

Por último, necesitamos un criterio para inicializar $\mathbf{Y}_0$. La opción más sencilla sería asignar $\mathbf{Y}_{0,ij} = \delta$, de tal forma que δ valga 0 si el i-ésimo autor no tiene asignada la j-ésima keyword, y 1 en el caso contrario. Sin embargo, como sabemos que en el fichero que disponemos para las keywords éstas están ordenadas de mayor a menor importancia, vamos a elegir un criterio un tanto arbitrario pero que dé más relevancia a las primeras keywords. Así, sea la keyword j, en posición k para el autor i, tendremos que:

$$\mathbf{Y}_{0,ij} = max(1 - 0.05k, 0.25) \tag{10}$$

Explícitamente, el algoritmo es el siguiente:

	Algoritmo LPA:
1:	Leer la matriz de adyacencia $\mathbf{W}$
2:	Calcular la matriz de grado $\mathbf{D}$
3:	Calcular la matriz de adyacencia normalizada $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$
4:	Inicializar la matriz $\mathbf{Y}_0$, en nuestro caso con el criterio (10)
5:	Definir umbral de convergencia (threshold), error>threshold y α
6:	$\mathbf{Y}_{old} = \mathbf{Y}_0$
7:	while error >threshold:
8:	$\quad \mathbf{Y}_{new} = \alpha \mathbf{S} Y_{old} + (1-\alpha)\mathbf{Y}_0$
9:	$\quad error = sum(abs(\mathbf{Y}_{new} - \mathbf{Y}_{old}))$
10:	$\quad \mathbf{Y}_{old} = \mathbf{Y}_{new}$
11:	return $\mathbf{Y}_{new}$

Si ejecutamos el algoritmo podemos ver que efectivamente el error tiende a 0, nótese que el siguiente gráfico está en escala logarítmica, alcanzando un valor muy cercano a 0 con muy pocas iteraciones:

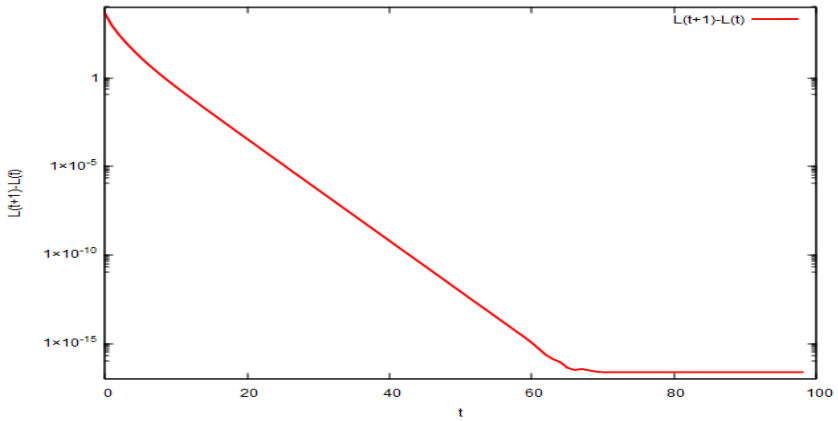
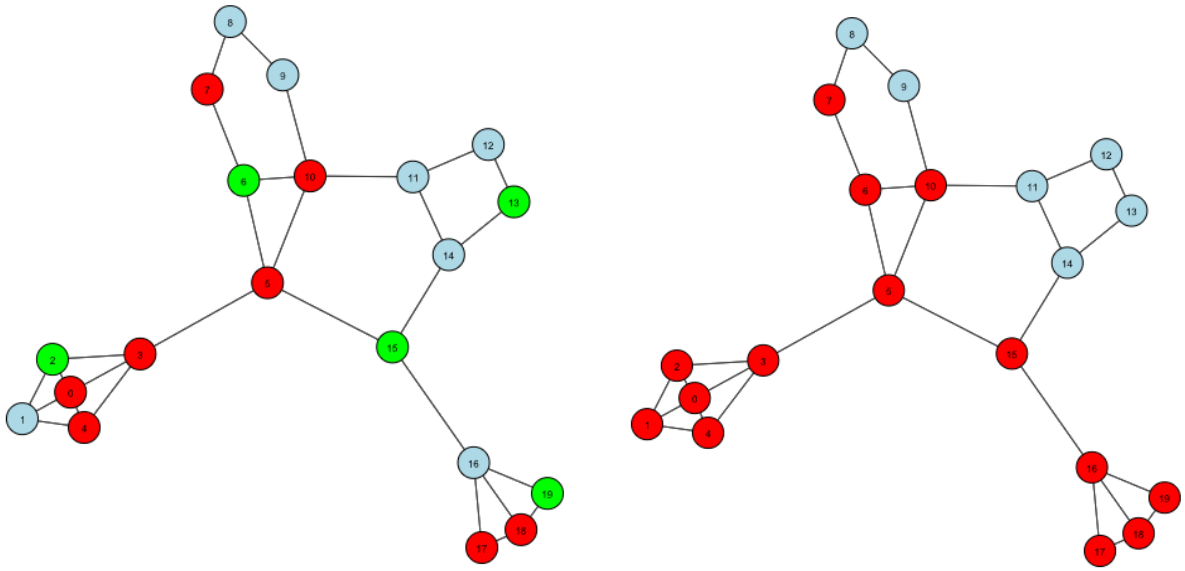


Figure 6: $\sum_{ij} \|\mathbf{Y}_{ij}(t+1) - \mathbf{Y}_{ij}(t)\|$

Una de las ventajas de este algoritmo es que como tanto $\mathbf{S}$ como $\mathbf{Y}$ son matrices que presentan un alto grado de dispersión (i.e. un alto porcentaje de elementos que son 0) podemos trabajar con matrices sparse, que son mucho más rápidas que las matrices clásicas a la hora de operar. En nuestro caso, con aproximadamente el doble de las interacciones necesarias para que el algoritmo converja hemos tardado 26,9 segundos. Dicho esto, como la matriz de adyacencia es del orden de $O(n^2)$ e invertirla del orden de $O(n^3)$, para c keywords tendremos que la complejidad del algoritmo será $O(n^3 + cn^2)$, esto es, aunque en nuestro caso haya convergido muy rápido, el tiempo necesario para alcanzar la solución crece muy rápidamente con el número de nodos. Este problema puede ser solucionado fácilmente sin sacrificar calidad, hasta el punto de reducir la complejidad a $O(ctn)$ donde t es el promedio de iteraciones en cada nodo (t«n), tal y como se expone en Fujiwara and Lrie 2014.

Para dejar más claro el funcionamiento de estos algoritmos, hemos corrido el LPA en una

red de juguete en la que cada nodo tiene asignada una única etiqueta que interpretaremos como $\mathbf{L} = \{rojo, azul, verde\}$, además inicializaremos la matriz $\mathbf{Y}_0$ tal que $\mathbf{Y}_{0,ij} = \delta$, esto es, 1 para la etiqueta que le corresponda y 0 para el resto. Una vez finalice el algoritmo elegiremos la etiqueta de cada nodo tal que $k_i = argmax_{j \leq c} \mathbf{Y}^*_{ij}$. Para la representación gráfica pintaremos cada nodo del color de su etiqueta asignada. Con $\alpha = 0.75$:



a) Red antes de ejecutar el LPA

b) Red posterior al de LPA

Figure 7: Ejecución del LPA sobre una red pequeña y con sólo 3 posibles etiquetas

3.2 bLPA: Bayesian Label Propagation Algorithm

Como hemos comentado, el LPA es un algoritmo diseñado para aprendizaje supervisado o semisupervisado, por lo tanto aunque converja a una solución no hay garantías de que esta cumpla con los requisitos que le pedimos. Por este motivo hemos desarrollado un segundo algoritmo inspirado en la propagación de etiquetas mediante actualización de probabilidad bayesiana de Yamaguchi, Faloutsos, and Kitagawa 2015. La idea principal es que la probabilidad de que el i-ésimo autor tenga asociada la j-ésima keyword vendrá dada por un valor que conocemos a priori β_{ij} y por la probabilidad de que sus vecinos tengan esa misma keyword. Más formalmente:

$$Y_{ij} \propto \frac{\sum_k^N W_{ik} Y_{kj} + \beta_{ij}}{\sum_k^N W_{ik}} \tag{11}$$

Donde el índice k recorre todos los nodos vecinos del nodo i. Ver que volvemos a usar las restricciones de continuidad suave (asumiendo que la probabiliad depende de las probabilidades de los nodos vecinos) además de la restricción de condiciones iniciales a través de los términos β_{ij}, que pueden entenderse como las condiciones iniciales (por analogía con el LPA, $\mathbf{Y}_0$).

Los autores proponen un algoritmo iterativo para resolver este sistema de ecuaciones pero

demuestran su convergencia únicamente para el caso en el que se tienen nodos etiquetados y nodos sin etiquetar, por lo que nosotros propondremos un algoritmo ligeramente distinto basado en la misma idea.

Sea el conjunto $\{N_i\}$ los nodos vecinos del nodo i. Mantendremos la nomenclatura del algoritmo anterior, usando el parámetro $\alpha \in (0, 1]$ para modelar la importancia de las condiciones iniciales. De esta forma tendremos el siguiente algoritmo, donde la función "normalice" normaliza a 1 cada una de las filas de una matriz:

	Algoritmo bLPA:
1:	Leer la matriz de adyacencia $\mathbf{W}$
2:	Dar un valor al parámetro $\alpha \in (0, 1]$
3:	Umbral de convergencia, threshold=0.0001, y número máximo de pasos, nsteps=1500
4:	Crear una variable error >threshold y otra variable n=0
5:	$\mathbf{Y}_{new}$ =$\mathrm{Y}_{old} = normalice(\mathrm{Y}_0)$
6:	while error >threshold or n<nsteps:
7:	$\mathbf{Y}_{new} = normalice(\mathrm{Y}_{new})$
8:	$\mathbf{Y}_{old} = \mathbf{Y}_{new}$
9:	for i in range(N): //recorremos todos los nodos de la red
10:	if $\mathbf{D}_{ii}$!=0: //recordemos que $\mathbf{D}$ era la matriz de grado
11:	for j in N_i: //recorremos todos los vecinos del nodo i
12:	peso=$\mathbf{W_{ij}}/\mathbf{D}_{ij}$
13:	$\mathbf{Y}_{new,j}$ =peso*$(\alpha\mathbf{Y}_{old,i} + (1-\alpha)\mathbf{Y}_{0,j})$ // Un solo índice indica fila
14:	error=$sum(abs(\mathbf{Y}_{new} - \mathbf{Y}_{old}))$
15:	n+=1
16:	return $\mathbf{Y}_{new}$

Como este algorimto es algo más brusco que el LPA es más dado a entrar en bucles internos que evitan que la convergencia tienda a 0, por lo que para ver si converge estudiaremos los cambios en las 10 keywords con más peso para cada autor en cada iteración. Esto es, en cada iteración sumaremos un punto por cada keyword que un autor cambie de entre sus 10 keywords con mayor λ_{ij}. Una puntuación baja significa entonces que aunque hagamos más iteraciones no va a haber cambios significativos en la asignación de las keywords más relevantes.

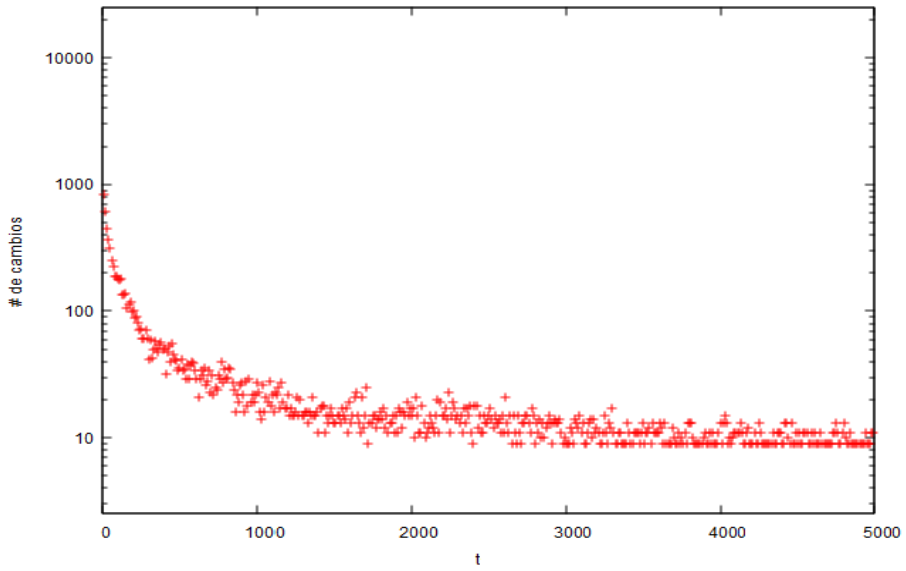


Figure 8: Número total de keywords que cambian entre las 10 con mayor λ_{ij} para cada autor.

Podemos ver que aunque aunque el error decae de manera sustancial (ver que estamos en escala logarítmica) a partir de las 2000 iteraciones se reduce muy lentamente y con una varianza significativa. Al contrario que en el caso anterior, no podemos usar matrices sparse porque estamos cambiando la "dispersión" de las matrices con cada operación, algo muy costoso. En parte es por este motivo que 2000 iteraciones hayan necesitado 21 minutos, cuando el LPA convergía en menos de 20 segundos.

4 Análisis de resultados

Una vez tenemos la matriz $\mathbf{Y}^*$ tenemos que asignar a cada comunidad sus keywords. Ver que lo que hemos hecho hasta ahora es propagar las keywords de los autores con la esperanza de que autores cercanos hayan obtenido unas keywords que, respetando sus keywords originales en mayor o menor medida según el parámetro α, sean parecidas entre ellos. Así, a cada autor le asignaremos las 10 keywords que tengan asociadas con mayor intensidad, de forma que a cada autor le corresponderá un vector de 10 componentes tal que:

$$\overline{k}_i = \begin{pmatrix} argmax^1_{j \leq c} \mathbf{Y}^*_{ij} \\ \vdots \\ argmax^{10}_{j \leq c} \mathbf{Y}^*_{ij} \end{pmatrix} \tag{12}$$

Donde hemos usado la nomenclatura $argmax^n$ para indicar que devuelve el argmax del n-ésimo valor más alto. Ver que a cada autor también puede tener asociado un vector con los valores $\mathbf{Y}^*_{ij}$ más altos, de tal forma que no perdemos información de con cuánta intensidad estaba asociada una keyword a un autor.

El número 10 no es completamente arbitrario. Una vez ejecutado el algoritmo, si ordenamos las keywords de los autores de mayor a menor intensidad asociada vemos una

exponencial decreciente[3], que decae a la mitad de su valor máximo torno a las 10 keywords. Para representarlo hemos ordenado las keywords de cada autor individualmente y promediado sobre cada columna, de tal forma que en el siguiente gráfico podemos ver el promedio para la i-ésima keyword más relevante, pero no es ninguna keyword concreta sino la más relevante de cada autor:

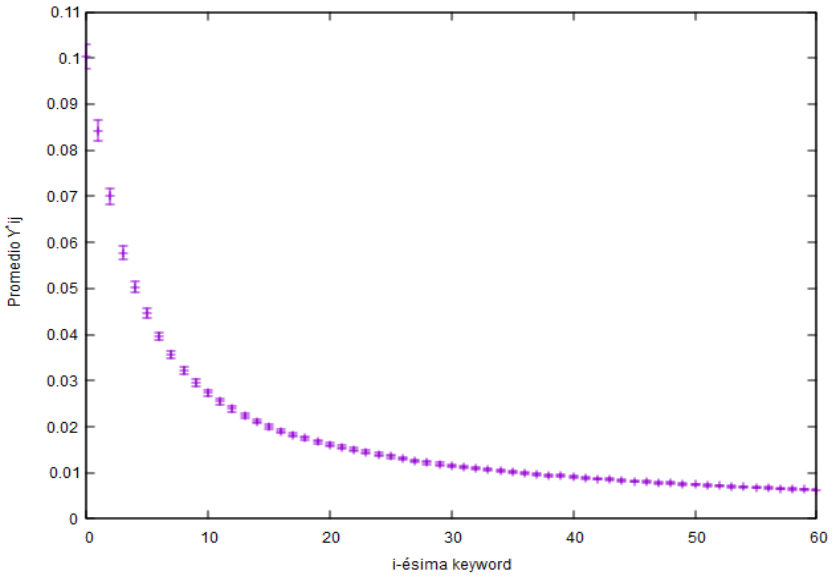


Figure 9: Valor promedio de las keywords más relevantes para cada autor

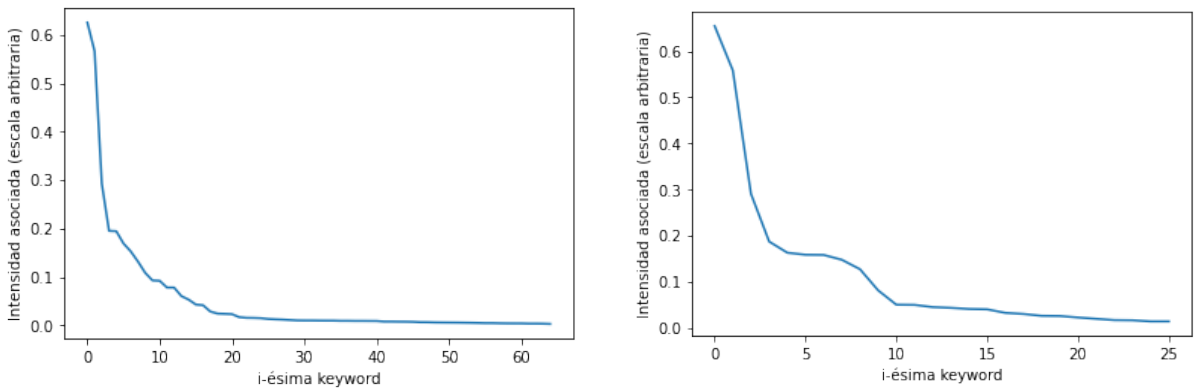
Una vez hemos asignado las keywords a los autores debemos hacer lo propio con las comunidades, pues esta re-asignación de keywords a autores es tan solo un paso intermedio con el objetivo de realizar una buena asignación a las comunidades. Para ello simplemente sumaremos las intensidades asociadas a cada keyword de cada autor dentro de una comunidad. Una vez hecho esto tenemos dos opciones: normalizar las sumas a 1 y elegir todas las keywords con un valor superior a un umbral arbitrario o elegir las m keywords con mayor valor.

Aunque Kampal parece inclinarse por la primera idea, nosotros haremos lo segundo con m=10, esto es, asignaremos a cada comunidad las 10 keywords cuya suma de las intensidades para todos los auotres de la comunidad sea mayor. Si estudiamos la intensidad asociada a las keywords de las comunidades vemos que con m=10 nos aseguramos de tener en cuenta las keywords relevantes sin coger demasiadas poco significativas, como vemos en Figure 10.

4.1 Comparación manual de asignación de keywords

Una vez asignadas las keywords conviene realizar una pequeña revisión manual para asegurarse de que el proceso se haya ejecutado correctamente. Efectivamente, vemos que

[3]Esto podría llevarnos a pensar que tendríamos que cambiar la función de inicialización (10) a una que tenga esta forma, sin embargo hemos comprobado que los resultados son muy robustos ante cambios de ésta.



a) Comunidad del área de física nuclear y astropartículas

b) Comunidad del área de física cuántica y de la materia

Figure 10: Ejemplo de curva de intensidad asociada a las keywords de dos comunidades

en general la asignación de keywords es como mínimo coherente entre sí[4]. En líneas generales, concluimos que en comunidades numerosas no hay mucha diferencia entre el resultado del bLPA y el resultado sin correr ningún algoritmo, aunque aún así se obtiene una buena asignación. Sin embargo, en comunidades más bien pequeñas el bLPA consigue una asignación con menos palabras genéricas y más representativas de la comunidad. A continuación presentaremos algunos ejemplos, el área correspondiente a cada keyword es proporcional a su peso dentro de la comunidad. Para evitar cherrypicking pondremos de ejemplo primero las dos comunidades más grandes y a continuación dos comunidades elegidas al azar de entre el 20% de comunidades más pequeñas. El algoritmo utilizado será bLPA con $\alpha = 0.92$ porque es el que mejores resultados da (profundizaremos en este punto en el apartado siguiente):

a) Sin algoritmo

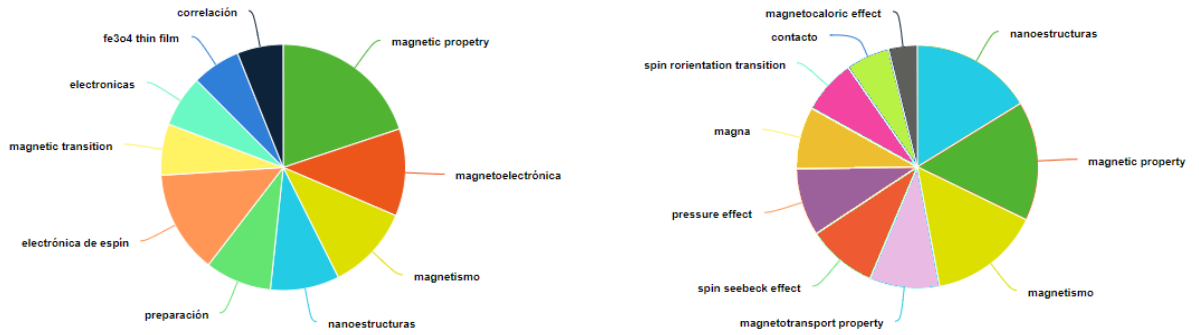
b) Con algoritmo

Figure 11: Comunidad del área de física nuclear y astropartículas

Este es un ejemplo de comunidad muy grande (63 autores) en la que apenas se nota diferencia entre las keywords elegidas sin ejecutar ningún algoritmo y ejecutando el bLPA. Vemos que en ambos casos obtenemos una buena asignación de keywords que nos permiten identificar perfectamente el área de estudio de la comunidad. Esto es algo que

[4]Con "coherente entre sí" nos referimos a que forman parte de la misma área temática. Para saber si la asignación es correcta este es un criterio necesario pero no suficiente, pues es necesario conocer a los autores de la comunidad.

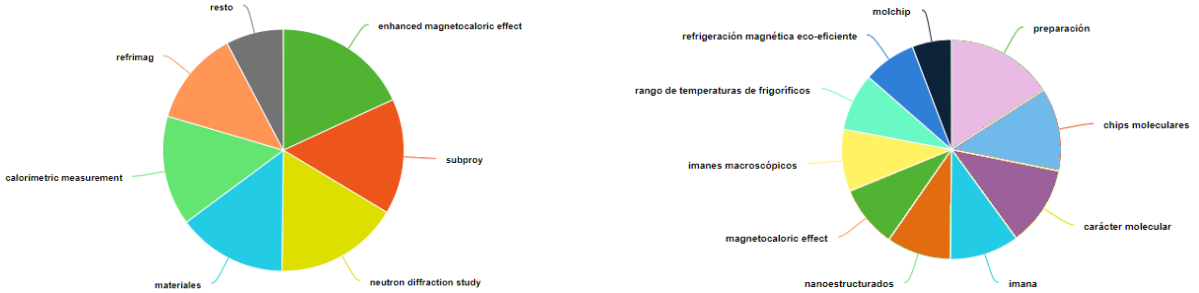
ocurre con la mayoría de comunidades grandes, por ejemplo para la comunidad tenemos que:



a) Sin algoritmo b) Con algoritmo

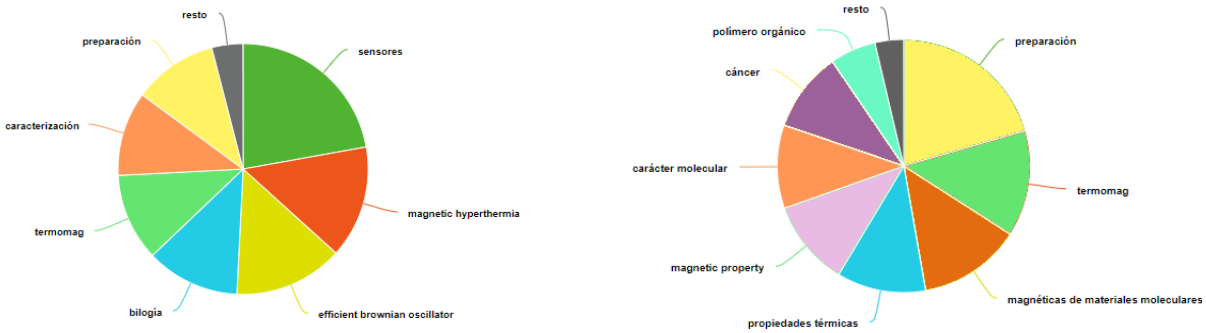
Figure 12: Comunidad del área de física de la materia condensada

Sin embargo, si nos vamos a comunidades más pequeñas vemos que las keywords asignadas sin ejecutar el algoritmo guardan menos relación entre ellas y no nos permiten una identificación clara del campo de estudio, situación que parece mejorar tras aplicar el bLPA. Veamos dos ejemplos:



a) Sin algoritmo b) Con algoritmo

Figure 13: Comunidad del área de física de la materia condensada



a) Sin algoritmo b) Con algoritmo

Figure 14: Comunidad del ára de nanociencia y materiales

Hemos usado el término "resto" para englobar a aquellas keywords cuyo peso es mucho más pequeño que el resto. Si nos fijamos por ejemplo en Figure 13 podemos ver que

aunque haya aparecido una palabra demasiado genérica ("preparación") el resto parecen ser más coherentes (vemos que se repiten palabras relacionadas con moléculas, magnetismo y calor), aunque igualmente aparecen keywords que no tienen mucho sentido (véase "cáncer" en Figure 14). Llama la atención que en general la asignación de keywords sin ejecutar ningún algoritmo es superior a la asignación que hace actualmente Kampal Research[5], es decir, una asignación que sólo tiene en cuenta la frecuencia con la que aparece cada keyword dentro de una comunidad, obviando por ejemplo la "intensidad" con la que cada autor tiene asociada su keyword (esto podría explicar por qué vemos tantas palabras muy genéricas, pues se repiten en muchos autores aunque con muy poca intensidad). Pueden verse dos ejemplos de los resultados que tiene ese método en Figure 1.

Esta revisión manual, sin embargo, es muy limitada por varios motivos. El primero es que comparar todas las comunidades llevaría mucho trabajo, pero hay problemas más de fondo. Sin conocimiento de primera mano de la especialización de los distintos autores es difícil concluir si la asignación de keywords obtenidas es mejor o peor que la que se tiene sin usar ningún algoritmo de propagación de etiquetas.

4.2 Presentación de métricas auxiliares

En última instancia, no queda más remedio que revisar manualmente la asignación de keywords para comprobar si ésta ha sido satisfactoria. Sin embargo, podemos desarrollar algunas métricas auxiliares que pueden ayudarnos con esa tarea. En primer lugar calculemos el solapamiento que hay entre las keywords con y sin algoritmo. Para ello conviene imaginar las keywords de cada comunidad como un vector en un espacio cuya dimensión sea el número de keywords únicas correspondientes a cada comunidad usando y sin usar el algoritmo. Más formalmente:

Sean $\{\mathbf{C}^0\}$ y $\{\mathbf{C}\}$ los conjuntos de keywords asociados a las distintas comunidades, donde el índice "0" hace referencia a la asignación sin aplicar ningún algoritmo, entonces para cada comunidad definiremos un vector de tamaño $\#C_i \cup C_i^0$, donde "#" es la función "cardinal" que relaciona un conjunto con el número de elementos que pertenecen a él y donde el valor de cada elemento será la intensidad asociada a cada keyword, asignando 0 en caso de que una keyword de $C_i \cup C_i^0$ no esté presente en C_i ó C_i^0. Es decir, si una comunidad concreta tuviera antes de ejecutar el alcoritmo las keywords $C_0 = \{"gato",\ "perro"\}$ y tras el algoritmo $C_i = \{"gato",\ "casa"\}$, supongamos por simplificar, sin perder generalidad, que todas con intensidad asociada 1, entonces definiríamos los vectores de tamaño 3 (tres elementos, "gato", "perro" y "casa"):

$$\vec{v}_0 = (1, 1, 0) \qquad \vec{v} = (1, 0, 1)$$

[5]Aunque las comunidades de autores que hemos obtenido no tienen que coincidir con las de Kampal Research, si es cierto que hay algunas en las que coinciden sus autores más relevantes, como las vistas en Figure 1

A continuación normalizamos cada vector y calculamos su producto escalar, de tal forma que para el ejemplo anterior el solapamiento sería 0,5. Aplicando este método para nuestras comunidades obtenemos el siguiente solapamiento. En el eje x tenemos las comunidades identificadas por un número y ordenadas de mayor a menor tamaño, y en la etiqueta arriba a la derecha observamos una métrica más general, que se ha calculado con la media ponderada por tamaño del solapamiento de las comunidades:

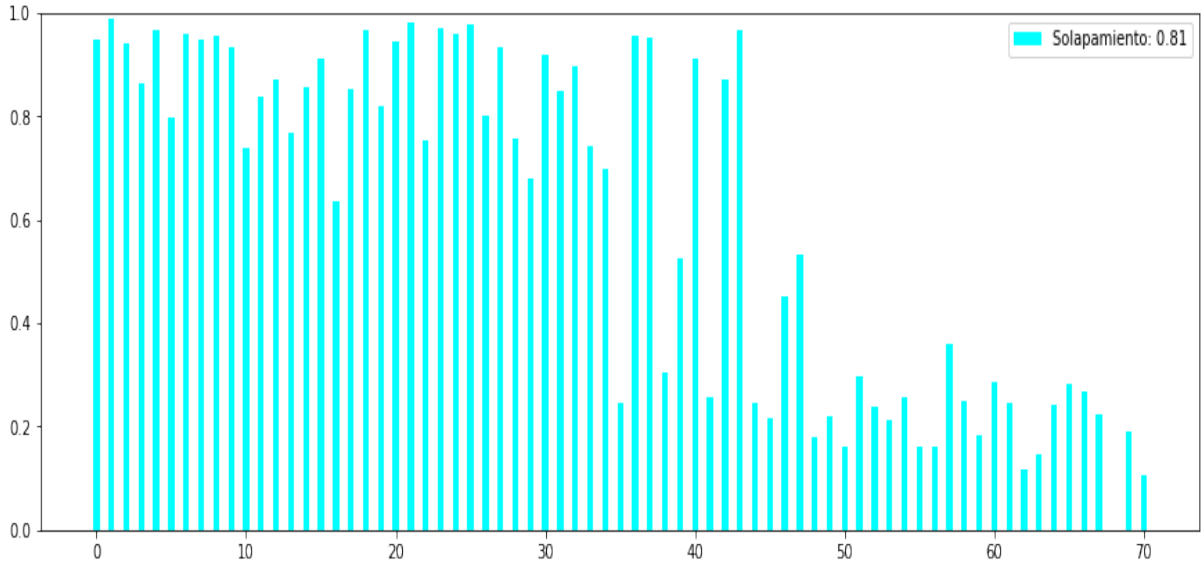


Figure 15: Solapamiento por comunidad y métrica global

Vemos que, tal como se había comentado en la revisión manual, mientras que las comunidades grandes tienen un gran solapamiento (i.e. las keywords asignadas son muy similares con y sin algoritmo) en las comunidades pequeñas el algoritmo cobra relevancia. Sin embargo, esta métrica tampoco nos permite distinguir si la asignación de keywords del algoritmo es mejor o no. Para ello definiremos dos métricas auxiliares que, aunque por sí sólas no puedan responder a esta pregunta, sí nos pueden servir de ayuda. Con estas métricas también podremos comparar fácilmente los resultados del LPA y el bLPA y explicar por qué hemos desarrollado la sección anterior con el bLPA:

- Homogeneidad interna: Para realizar una buena asignación de keywords a una comunidad a partir de las keywords de sus autores hemos ejecutado dos algoritmos con la idea de que éstas converjan, de tal forma que las keywords de los autores de una misma comunidad deberían ser más o menos iguales. Idealmente todos los autores de la comunidad convergerían a las mismas 10 keywords, que serían las 10 keywords que luego se asignarían a la comunidad. Para tratar de parametrizar esta convergencia definiremos una métrica consistente en recorrer las keywords del autor 1 de cada comunidad, sumando un punto por cada keyword que se repita en alguno de los autores siguientes. Haremos lo mismo con el autor 2, comparando ahora con los autores 3,4,...,N_i, así hasta comparar al autor N_i-1 con el número N_i, siendo N_i el tamaño de la i-ésima comunidad. Normalizamos dividiendo por $10\,[(N-1)+(N-2)+\ldots+1]$. La explicación puede quedar más clara explicitando el algoritmo:

Algoritmo homogeneidad interna

```
Leer L, lista tal que L_ijk sea la k-ésima keyword del j-ésimo autor de la i-ésima comunidad
for i in range(N): //N es el número de comunidades (N=len(L))
 for j in range(N_i): // N_i es el tamaño de la i-ésima comunidad (N_i = len(L_i))
  for k in L_ij: //Recorremos las keywords del autor
   for l in range(j+1,N_i):
    if k in L_il: aux+=1
  for j in range(N_i - 1): norm+=10*(N_i - j)
  Hint_i=(aux/norm) //vector con la homogeneidad interna de cada comunidad
  aux=norm=0
```

Como comparar la métrica para cada comunidad puede ser algo engorroso a la hora de evaluar la calidad de los algoritmos, haremos una suma ponderada por el tamaño de las comunidades de tal forma que queda definido el escalar: $H.int = \frac{1}{\sum_i^N N_i} \sum_i^N Hint_i N_i$.

- Heterogeneidad externa: Además de que cada comunidad sea homogénea, queremos que haya diferencias entre comunidades, esto es, que las keywords en $\mathbf{C}_i$ no se repitan demasiado en $\mathbf{C}_j \, \forall i \neq j$. Esta unicidad no tiene por qué ser perfecta, de hecho esperamos que haya keywords que se repitan en distintas comunidades. Sin embargo, una heterogeneidad externa muy baja podría significar que hay keywords que se han propagado demasiado por toda la red, por lo que dejan de ser representativas de cada comunidad de investigadores. También es posible que veamos una heterogeneidad externa baja en comunidades mal clusterizadas, por ejemplo si lo que debería ser una única comunidad se ha dividido en dos más pequeñas es esperable que sus keywords sean muy parecidas. Para mitigar los problemas que una mala clusterización haya podido causar y no penalizar demasiado la aparición de keywords generales, definiremos un parámetro umbral (threshold) que indicará el número de veces que una keyword de una comunidad puede aparecer en otra comunidad antes de que penalicemos que eso suceda (esto es, si definimos un umbral de valor 4 pero la keyword de una comunidad se repite en otras 3, lo contaremos como si no se repitiera en ninguna). Esto último se entiende mejor echando un vistazo al algorítmo explícito:

Algoritmo heterogeneidad externa

```
aux1=0, aux2=0
threshold=4 //número de veces que una keyword puede repetirse en otras comunidades
for i in range(N): //N es el número de comunidades
  for j in C_i: //Recorremos las keywords de cada comunidad
    for k in range(len(C)): //Recorremos las comunidades
      if k!=i:
        if j in C_k: aux1+=1
    if aux1<=threshold: aux2+=1
    aux1=0
  Hext_i = aux2/len(C_i)
```

Análogamente a la métrica anterior, definimos: $H.ext = \frac{1}{\sum_i^N N_i} \sum_i^N Hext_i N_i$.

Por toda la casuística hay que tomar estas métricas siempre como algo auxiliar, y desde luego no como condiciones suficientes, pero sí hasta cierto punto necesarias. Un ejemplo claro de que ambas son necesarias es cuando las mismas keywords se propagan por toda la red, es decir, tras ejecutar el algoritmo se les asigna a todos (o casi todos) los autores de la red las mismas keywords, lo cual ocurre en el LPA cuando $\alpha = 1.0$. En este caso la homogeneidad interna es casi 1, pero una puntuación tan alta no quiere decir que hayamos hecho una buena asignación, y eso podemos verlo inmediatamente ya que la heterogeneidad externa es prácticamente 0. Podemos ver los valores de $H.int$ y $H.ext$, que tratan de representar el valor "total" de las métricas auxiliares, en la leyenda de los gráficos:

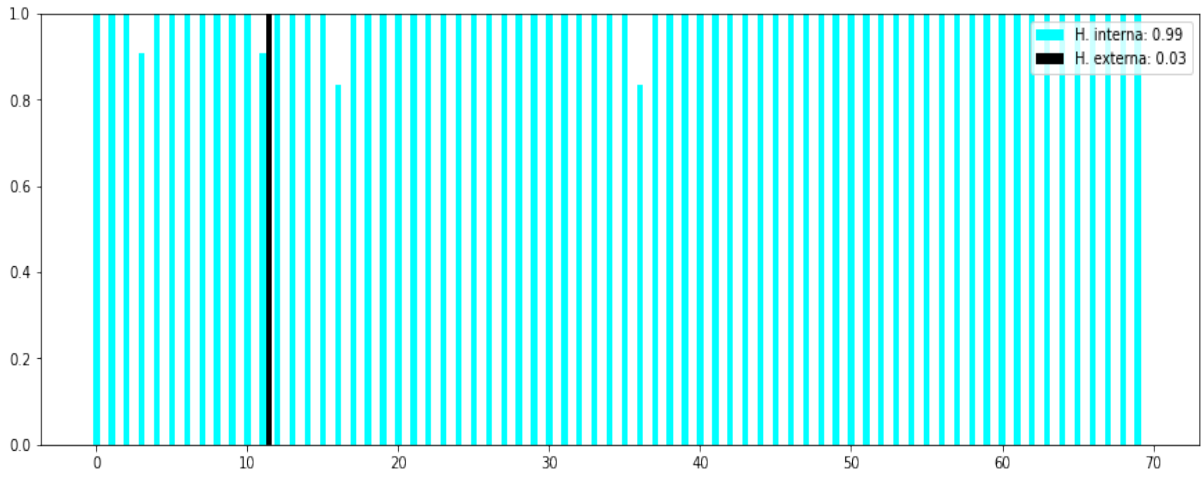


Figure 16: Homogeneidad interna y heterogeneidad externa en LPA para $\alpha = 1$ (la heterogeneidad externa vale 0 para todas las comunidades menos una)

4.3 Análisis de las métricas auxiliares

Antes de comparar los distintos algoritmos conviene hacer una estimación de qué valor del parámetro α es óptimo. Para ello correremos los algoritmos (obtenemos idénticos resultados para el LPA y el bLPA) para distintos valores de α y calcularemos para cada caso la homogeneidad interna y la heterogeneidad externa:

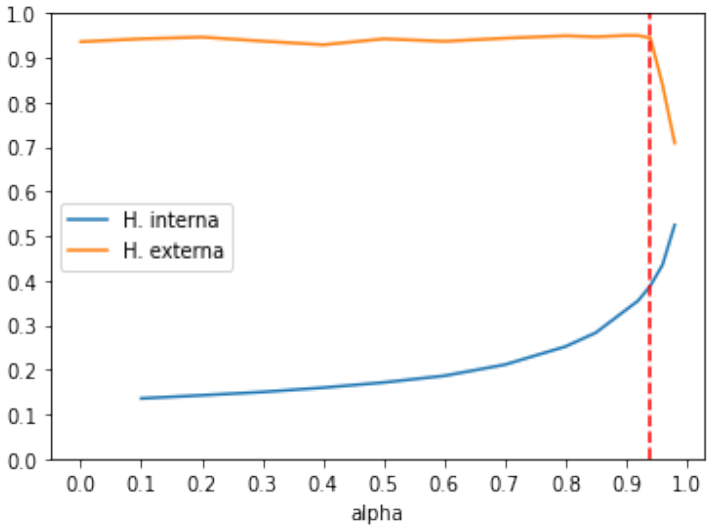


Figure 17: Homogeneidad interna y heterogeneidad externa en función de α para el bLPA

Ver que, tal y como esperábamos, la homogeneidad interna crece con alpha. Vemos como la heterogeneidad externa cae dramáticamente a partir de $\alpha = 0.94$, esto es lo que justifica que hayamos utilizado $\alpha = 0.92$ en las secciones anteriores. De esta forma evitamos que las keywords más relevantes de la red se propaguen demasiado.

A continuación, comparamos ambas métricas para el caso sin utilizar ningún algoritmo[6], el LPA y el bLPA con $\alpha = 0.92$:

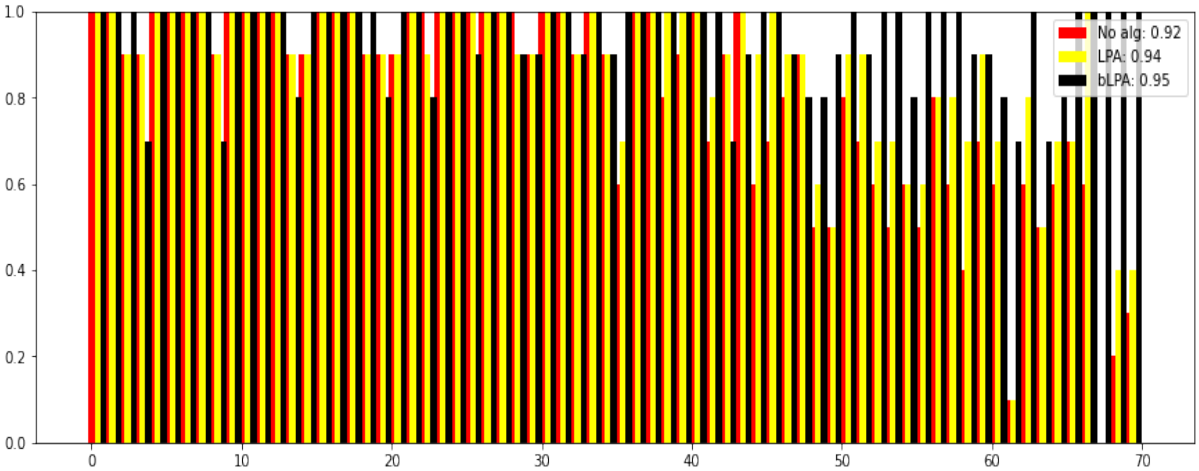


Figure 18: Heterogeneidad extrena de cada comunidad y total

[6]La asignación de keywords se realiza de la misma forma que en el resto de casos. Se inicializa la matriz $\mathbf{Y}_0$ y se seleccionan las keywords de los autores y las comunidades de tal forma que $\mathbf{Y}^* = \mathbf{Y}_0$

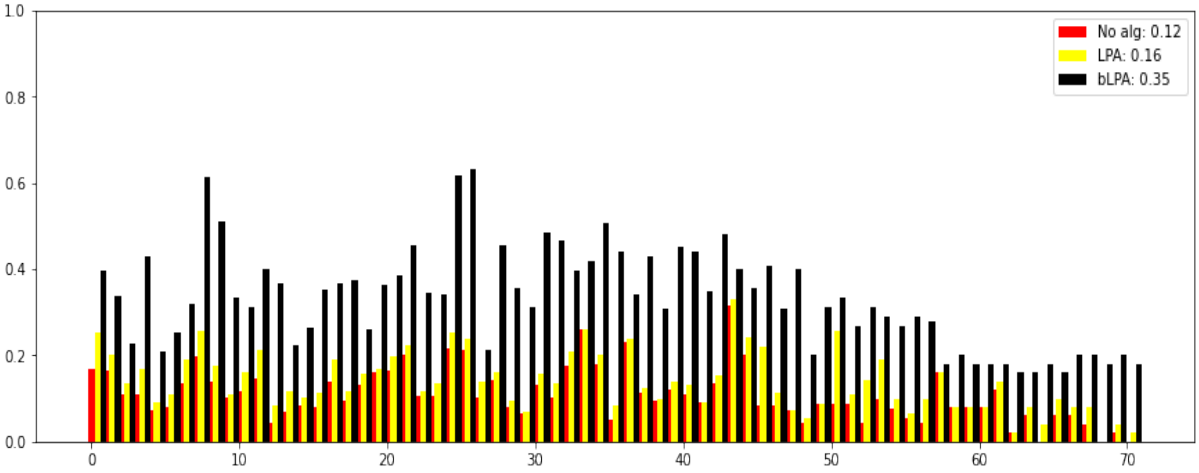


Figure 19: Homogeneidad interna de cada comunidad y total

Destaca la poca convergencia que alcanza el LPA (para $\alpha \neq 1$), prácticamente indistinguible de no haber utilizado ningún algoritmo. Este algoritmo está diseñado para el caso en el que hay nodos etiquetados y nodos sin etiquetar, de tal forma que los nodos etiquetados actúan como "fuentes" de etiquetas y el resto como "sumideros". Aunque en una red muy pequeña con pocas etiquetas el algoritmo funciona satisfactoriamente, en una red más grande (aunque como hemos comentado anteriormente la nuestra apenas tiene 2830 nodos), pero en especial, en la que cada nodo tiene inicialmente varias etiquetas (con una intensidad dada por (10)), la convergencia es mínima.

Por otro lado, vemos que el bLPA ofrece una homogeneidad interna significativamente superior al resto. Destaca que ésta sea mayor en comunidades de tamaño grande y mediano que pequeño. Estas comunidades grandes tienen una estructura mejor definida con unos nodos que destacan sobre el resto (por ejemplo, su grado o centralidad son mucho mayor que los de sus vecinos), mientras que en comunidades muy pequeñas de 3, 4 o 5 autores es probable que ningún nodo sea lo suficiente relevante como para influir en el resto. Esto se ve agravado si la existencia de esas comunidades tan pequeñas se debiera más a una mala clusterización que a una característica intrínseca de la red, pues de ser ese el caso los nodos estarían muy débilmente conectados entre ellos respecto al resto de sus vecinos. De todas formas veremos en el siguiente apartado que el bLPA consigue una asignación de keywords satisfactoria incluso en comunidades pequeñas donde no ejecutar ningún algoritmo no ofrece resultados tan buenos.

Es el hecho de que la homogenidad interna del bLPA sea superior a la de los otros dos métodos, mientras que su heterogeneidad externa es igual, es lo que nos permite decantarnos por este algorimo como el mejor de los tres.

Por último y para clarificar lo que hemos hecho, representaremos el grafo de comunidades asignando a cada comunidad una única keyword (la de mayor peso, recordar que tenemos 10 keywords para cada comunidad ordenadas por relevancia), donde podemos apreciar particularidades de nuestra Facultad como la alta relación entre la física y la biología, así como la cercanía entre comunidades etiquetadas con keywords de un mismo campo (como "grupo de tecnologías electrónicas", "grupo de diseño electrónico" y "sensores"):

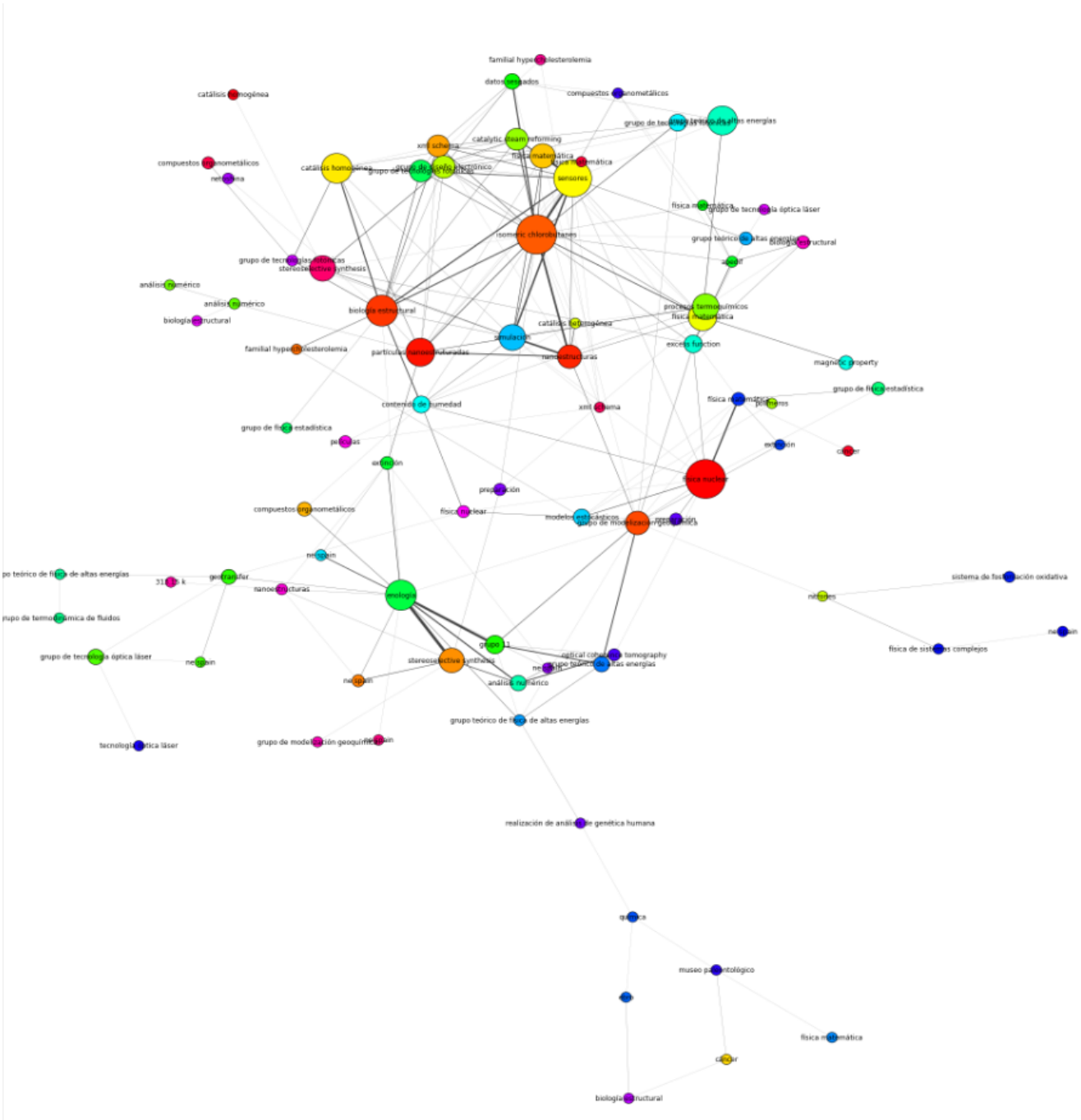


Figure 20: Comunidades y keyword asociada.

5 Conclusión y futuras vías de investigación

Como hemos repetido insistentemente a lo largo de este trabajo, nuestro objetivo era el de realizar una correcta asignación de keywords a una comunidad de autores de trabajos científicos. El principal problema es que no existe una definición de "correcta" asignación, y la casuística es muy elevada. Además, el hecho de que las redes de coautoría en general (y la nuestra en particular) sean redes libres de escala (Figure 2) nos incita a usar teoría de redes complejas en vez de otros métodos más sencillos como ponderar las keywords de los autores según su grado. Para solucionar este problema hemos propuesto dos algoritmos, de los cuales uno ha dado resultados satisfactorios, especialmente en las comunidades pequeñas. Este algoritmo se basa en la idea de que la probabilidad de que un autor tenga asociada una cierta keyword dependerá tanto de la probabilidad de que sus vecinos la tengan como de cierta información a priori.

Como no existe un criterio objetivo para cuantificar el éxito de nuestra estrategia, dada la naturaleza del problema, diseñamos ciertas métricas auxiliares a través de las cuales comprobamos que las keywords de las comunidades obtenidas por nuestro algoritmo cumplen que son más homogéneas dentro de cada comunidad respecto a no usar ningún algoritmo, mientras que entre comunidades mantienen una heterogeneidad similar. Por este motivo, aunque en última instancia siempre habrá que revisarlo manualmente con conocimiento de causa para asegurarlo, podemos afirmar con cierta confianza que nuestro método sí ofrece buenos resultados.

En cuanto al margen de mejora, si bien es cierto que la asignación de keywords parece bastante satisfactoria, tambien es cierto que se podrían probar otros enfoques. Uno de los temas candentes y más prometedores en el ámbito de las redes complejas es el machine learning. El uso de inteligencia artificial podría aportar algunas ventajas frente a nuestro método como la incorporación de conocimiento previo (a través de un preentrenamiento), de tal forma que pueda refinar la asignación de keywords al entender la relación semántica que existe entre ellas. Sin embargo esto también presenta nuevos problemas, pues este tipo de filtros podrían dar por errores algunas características peculiares de la red. Por ejemplo, una inteligencia artificial podría ver errónea la asignación a una misma comunidad de palabras tan dispares como "vida", "cáncer", "nanoestructuras", "propiedades magnéticas" y "unizar", sin embargo atienden a una comunidad real de la Universidad de Zaragoza que estudia el cáncer a través de partículas magnéticas microscópicas. Aún con esto en cuenta, el machine learning es un enfoque prometedor que podría mejorar los resultados propuestos en este trabajo.

Manteniéndonos en el área de redes complejas, otra vía factible de investigación sería la aplicación de distintos algoritmos de propagación de etiquetas. Un tipo de algoritmo particularmente prometedor es el LPAm+: Advanced modularity-specialized label propagation algorithm (Liu and Murata 2010). La ventaja de este algoritmo es que propaga etiquetas a la vez que forma comunidades.

Agradecimientos especiales a Alejandro Rivero y Juan Luis Durán por su apoyo y supervisión.

Bibliografía

Barabasi, Albert-Laszlo et al. (Apr. 2001). "Evolution of the social network of scientific collaborations. Physica A: Statistical Mechanics and its Applications, 311(3), 590-614". In: *Physica A: Statistical Mechanics and its Applications* 311, pp. 590–614. DOI: 10.1016/S0378-4371(02)00736-7.

Fujiwara, Y. and G. Lrie (Jan. 2014). "Efficient label propagation". In: *31st International Conference on Machine Learning, ICML 2014* 3, pp. 2355–2363.

Liu, X. and T. Murata (Apr. 2010). "Advanced modularity-specialized label propagation algorithm for detecting communities in networks". In: *Physica A: Statistical Mechanics and its Applications* 389.7, pp. 1493–1500. URL: https://doi.org/10.1016%5C%2Fj.physa.2009.12.019.

Pons, Pascal and Matthieu Latapy (2005). *Computing communities in large networks using random walks (long version)*. arXiv: physics/0512106 [physics.soc-ph].

Qaiser, Shahzad and Ramsha Ali (July 2018). "Text Mining: Use of TF-IDF to Examine the Relevance of Words to Documents". In: *International Journal of Computer Applications* 181. DOI: 10.5120/ijca2018917395.

Yamaguchi, Yuto, Christos Faloutsos, and Hiroyuki Kitagawa (May 2015). "SocNL: Bayesian Label Propagation with Confidence". In: pp. 633–645. ISBN: 978-3-319-18037-3. DOI: 10.1007/978-3-319-18038-0_49.

Zhou, Dengyong et al. (Mar. 2004). "Learning with Local and Global Consistency". In: *Advances in Neural Information Processing Systems 16* 16.

Zhu, Xiaojin and Zoubin Ghahramani (2002). "Learning from Labeled and Unlabeled Data with Label Propagation". In: URL: http://www.scholar.google.com/url?sa=U%5C&q=http://www.cs.cmu.edu/~zhuxj/pub/propagate.ps.gz.