Posicionamiento de Proyectos de Software en Base al Análisis de los Grafos de sus Relaciones de Dependencia

Jorge Ruiz-Robles
jorge.ruiz@udep.pe
Universidad de Piura, Perú
Calle Bellavista 199,
Miraflores
Lima – Perú

Juan Carlos Sotelo
jsc@andestec.com
Universidad de Piura Perú
Calle Bellavista 199,
Miraflores
Lima – Perú

Alejandro Ruiz-Robles
alejandro.ruiz@udep.pe
Universidad de Piura, Perú
Av. Ramón Mugica 131,
Urb. San Eduardo
Piura – Perú

Jorge Morato
jmorato@inf.uc3m.es
Universidad Carlos III
de Madrid, España
Av. Universidad 30, Leganés
Madrid – España

Resumen: *El software de código abierto es cada vez más popular y utilizado a nivel mundial, gracias a la calidad de sus productos. Los repositorios de código abierto son herramientas para acceder a este tipo de software, pero cuando se trata de buscar algún componente en particular, no siempre se puede encontrar rápidamente lo que se necesita. El presente trabajo estudia la viabilidad de utilizar otros algoritmos de ordenamiento para mejorar los resultados ofrecidos por los repositorios de software de código abierto. Con este fin se analiza el uso de algoritmos de ordenamiento basados en los grafos de relaciones que existen entre proyectos de software de código abierto. Se han comparado los resultados de cuatro diferentes algoritmos de ordenamiento y la opinión de un conjunto de expertos en el área de dominio donde se realizó el experimento. Los resultados muestran que existen ligeras discrepancias entre el ordenamiento proveído por el repositorio de código abierto, los algoritmos de ordenamiento y la opinión de los expertos. Estos resultados ponen en relieve la posibilidad de incluir nuevas técnicas de ordenamiento en repositorios de código abierto con el fin de obtener mejores resultados en las búsquedas realizadas por los usuarios.*

Palabras claves: grafos, posicionamiento, HITS, PageRank, open source, repositorios, búsqueda.

Abstract: *Open source software is becoming more popular worldwide due to the quality of its products. Open source repositories are tools to access this kind of software, but when it comes to search any particular component, it is not easy to find what is required quickly. This paper studies the feasibility of using other sorting algorithms, in order to improve the results provided by open source software repositories; for this purpose the use of sorting algorithms based on graphs of relationships between open source software projects is analyzed. The results of four different sorting algorithms have been compared with the opinion of a group of experts in the domain area where the experiment was conducted. The results show that there are slight discrepancies between the ranking provided by the open source repository, sorting algorithms and expert opinion. These results underscore the possibility of including new sorting techniques in open source repositories in order to obtain better results in searches performed by users.*

Keywords: graphs, positioning, HITS, PageRank, open source, repositories, search.

1. Introducción

Se define como software open source a aquellos productos de software que han sido distribuidos junto con su código fuente, bajo una licencia que asegura que el trabajo basado en él también esté disponible como código fuente, protegiendo ciertos derechos de los autores originales y prohibiendo restricciones acerca del uso que se le da al software y quién puede utilizarlo (O'Reilly, 2012) (Chris DiBona, 1999).

Los repositorios de software open source son portales web cuyo propósito es alojar diversos tipos de software que pueden ser descargados libremente para ser reutilizados en nuevos proyectos. Esta forma de desarrollo goza de gran popularidad y aceptación entre los desarrolladores de software (Weber, 2003) gracias al crecimiento y seguimiento que les hacen sus equipos de desarrollo (Deshpande & Riehle, 2008), conformados por comunidades open source que de manera voluntaria llevan a cabo el lanzamiento frecuente de actualizaciones y la buena calidad en general de sus productos (Lakhani & Wolf, 2003). Entre los proyectos open source más relevantes podemos encontrar: el servidor Web Apache, responsable del 60% de los servidores Web a nivel mundial (Royal Pingdom, 2011) y la fundación Mozilla, creadora del navegador web Firefox, actualmente usado en el 24% de las ordenadores a nivel mundial (StatCounter, 2013) y el sistema operativo Linux, entre otros.

En la actualidad, muchos proyectos open source son de vital importancia para el funcionamiento, desarrollo y crecimiento de las tecnologías de la información a nivel mundial (O'Reilly, 2012). Entre los repositorios de código más conocidos, tenemos a CodePlex, GitHub, Google Code y SourceForge.

Cuando se va a llevar a cabo un proyecto de software, basado en componentes open source, se recurre a los buscadores de los repositorios (Weber, 2003) para la búsqueda de componentes. El orden sugerido por el motor de búsqueda no siempre muestra primero los componentes que efectivamente se requieren, por lo que además se consultan opiniones de expertos en foros, blogs, redes sociales, para afinar la búsqueda y luego descargar y probar los componentes para validar que cumplen con lo que se requiere para el proyecto. Este proceso puede tomar un tiempo mayor al previsto dado que existen diversos componentes que realizan una misma función.

En el presente trabajo, se ha estudiado la viabilidad de utilizar diferentes algoritmos de ordenamiento que permitan mejorar los resultados ofrecidos por los portales de código open source.

En la sección 2, se menciona algunos trabajos previos sobre ordenamiento en base a enlaces. En la sección 3, se describe brevemente la teoría de grafos. En la sección 4, se plantea el problema de ordenamiento que se busca resolver, y, en la sección 5, se propone una solución para dicho problema. En la sección 6, se lleva a cabo una prueba experimental para evaluar la efectividad de la solución propuesta, y los resultados son analizados en la sección 7. Las conclusiones y recomendaciones pueden encontrarse en la sección 8 y los trabajos futuros en la sección 9.

2. Trabajos previos

Los algoritmos de posicionamiento son usados por los buscadores en la Web para valorar los resultados de las búsquedas que realizan. La valoración asignada a cada resultado es la que permite definir el orden en que se presentan los resultados, y se determina en base a la estructura de enlaces que presenta cada página Web respecto a un subconjunto de páginas relacionadas.

Existe gran cantidad de estudios acerca de cómo explotar la estructura de enlaces de un conjunto de páginas Web. Pitkow hizo una tesis doctoral con una gran variedad de análisis basados en enlaces (Pitkow, 1997). Weiss estudió los métodos de agrupamiento que toman en cuenta la estructura de los enlaces (Weiss, y otros, 1996). Spertus estudió qué información se puede obtener de la estructura de enlaces para toda una variedad de aplicaciones (Spertus, 1997). Kleinberg desarrolló un modelo de la Web en forma de *Hubs y Authorities* llamado Algoritmo HITS, basado en el cálculo del vector propio de una matriz de co-citaciones de la Web (Kleinberg, 1998). Page y Brin plantearon el método PageRank de valoración de páginas Web medido en base al interés humano y la atención puesta en dichas páginas (Page & Brin, 1998).

En la actualidad, los repositorios open source presentan sus resultados en base al grado de relevancia, el cual se mide mediante el análisis textual de los contenidos de cada proyecto, cuantificando las coincidencias con el texto ingresado en el buscador (SourceForge, 2012). Este análisis textual no toma en cuenta el grado de referenciación que tiene el proyecto, por parte de otros proyectos del mismo tipo.

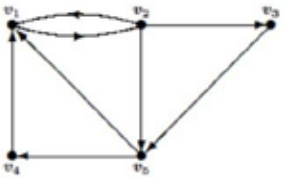
3. Teoría de Grafos

Un grafo G consiste en un conjunto finito no vacío V(G) de elementos llamados vértices o nodos y un conjunto finito A(G) de pares ordenados de distintos vértices llamados arcos o aristas. Se denomina V(G) al conjunto de vértices y A(G) al conjunto de arcos de G. El grafo G puede representarse como G = (V, A) (Gross & Yellen, 2003) (Bang-Jensen & Gutin, 2007).

3.1. Matriz de adyacencia

Dado el grafo G = (V,A) de orden N, a él se asocia una matriz cuadrada M de orden NxN, tal que cada fila se asocia a un nodo de V, y cada columna se asocia también un nodo de V. La celda M_{ij} contiene la cantidad de aristas de A de la forma (i,j), donde M_{ij} es el número de arcos que tienen a v_i como extremo inicial y a v_j como extremo final. A esta matriz se le conoce como matriz de adyacencia.

La matriz de adyacencia de un grafo direccional no es simétrica. Es una matriz binaria. El número de unos que aparecen en una fila es igual al grado de salida del correspondiente vértice y el número de unos que aparecen en una determinada columna es igual al grado de entrada del correspondiente vértice. En la **Figura 28¡Error! No se encuentra el origen de la referencia.** se puede apreciar un grafo dirigido junto a su matriz de adyacencia.



	V1	V2	V3	V4	V5	
V1	0	1	0	0	0	1
V2	1	0	1	0	1	3
V3	0	0	0	0	1	1
V4	1	0	0	0	0	1
V5	1	0	0	1	0	2

Figura 28. Grafo dirigido y su matriz de adyacencia

3.2. Medidas de centralidad

La centralidad se refiere a una medida posible de un vértice en un grafo que determina su importancia relativa dentro de éste (Bavelas, 1958).

Las medidas de centralidad se pueden agrupar en medidas radiales y medidas mediales. Las radiales toman como punto de referencia un nodo que inicia o termina recorridos por la red, mientras que las mediales toman como referencia los recorridos que pasan a través de un nodo. Existen cinco medidas muy usadas en análisis de redes, las radiales: grado nodal, cercanía, vector propio y excentricidad, y las mediales, como la intermediación (Fiedler, 1975) (Newman, 2005).

4. Planteamiento del problema

Al iniciar un nuevo proyecto de software basado en componentes open source, parte de las tareas que se realizan durante el diseño del sistema, es definir con qué componentes se va a trabajar (Lakhani & Wolf, 2003). Si el equipo de desarrollo ya cuenta con experiencia podrán definir rápidamente qué componentes utilizar. Por otro lado, si se trata de un nuevo tipo de proyecto, se deberá hacer una búsqueda de los componentes disponibles para las tareas que necesitan que el sistema lleve a cabo.

Para esta búsqueda, los desarrolladores optan por una de las dos siguientes alternativas:

- Consultar foros y blogs de expertos: en donde deben buscar información relacionada al tema de su interés, con el propósito de pedir o encontrar alguna recomendación si es que la hay o esperar respuesta.
- Buscar los proyectos más populares en los repositorios de software: deben elegir un repositorio con el que trabajar, luego descargar y probar cada uno de los proyectos hasta encontrar el que cumpla con sus necesidades. Los repositorios generalmente presentan sus resultados ordenados en función de los criterios de búsqueda, es decir, muestran primero los proyectos cuyo nombre y descripción coincide más con las

palabras ingresadas por el usuario al buscador. Cada uno de estos portales maneja sus propios criterios de posicionamiento al momento de hacer búsquedas. En algunos casos, los ordenan por número de descargas, por calificación de los usuarios o inclusive por promedio de descargas diarias/semanales/etc. (SourceForge, 2013) (Microsoft, 2012) (Google, 2013) (GitHub.com, 2012). Esta situación no asegura que entre los primeros resultados estará lo que los desarrolladores necesitan.

En cualquiera de los dos casos, se necesita invertir una determinada cantidad de tiempo y ser especialmente cuidadoso con la elección de componentes, pues un componente que no fuera totalmente adecuado para el proyecto, puede afectar la calidad del producto desarrollado.

Hipótesis: La aplicación de un método de ordenamiento en base a los parámetros obtenidos de un grafo, construido con las relaciones entre proyectos de software, permite mejorar el ordenamiento que los repositorios de dichos proyectos ofrecen, al tratar de satisfacer las necesidades del usuario.

5. Solución propuesta

Para comprobar la hipótesis planteada fue necesario verificar si era posible utilizar un algoritmo de ordenamiento que genere un mejor posicionamiento de los proyectos de software buscados. Para la validación se propusieron tres métodos diferentes de posicionamiento, y se contrastaron con el ordenamiento generado por un portal de repositorio de software (SourceForge, 2013).

Los métodos propuestos se desarrollaron tomando como punto de partida un grafo, el cual fue diseñado en base a un conjunto de proyectos y sus dependencias, de manera que los proyectos fueron representados como nodos del grafo, y las dependencias entre proyectos como aristas del grafo. El análisis del grafo permitió encontrar una serie de características que presentan sus nodos y aristas (por ejemplo, las medidas de centralidad), las mismas que definieron un puntaje o valoración para cada nodo, en función de su participación en el grafo. Esta valoración fue clave para las propuestas de posicionamiento. Se utilizó de manera que los proyectos correspondientes a los nodos con mayor puntaje fueron los más relevantes en cada propuesta.

5.1. Limitaciones del estudio

Existen ciertas limitaciones importantes que se deben considerar al aplicar la propuesta planteada:

- El análisis se ha realizado sobre un sólo tipo de proyecto específico y sobre un único repositorio de software.
- En la muestra sólo se consideran proyectos que han sido referenciados al menos una vez. No aplican proyectos no referenciados. El tamaño de la muestra (cantidad de proyectos analizados) está definido sólo con el propósito de realizar una validación preliminar de la viabilidad de la propuesta.

5.2. Método de grafo propuesto

El método propuesto se presenta en la **Figura 29¡Error! No se encuentra el origen de la referencia.** y consta de 6 pasos.

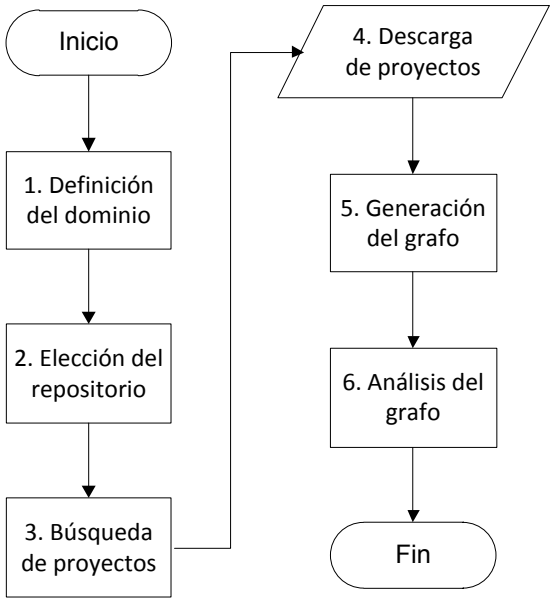


Figura 29. Diagrama de flujo del método propuesto

Paso 1: Definición del dominio

Consiste en definir de qué tipo serán los proyectos de software a analizar. Definir un dominio permite encontrar más relaciones entre los proyectos a considerar y permite analizar las similitudes y diferencias que presentan entre ellos.

Paso 2: Elección del repositorio

Consiste en elegir un repositorio de software open source, en el cual realizar la búsqueda de proyectos del dominio previamente definido.

Paso 3: Búsqueda de proyectos

Consiste en buscar dentro del repositorio seleccionado, proyectos del dominio elegido.

Paso 4: Descarga de proyectos

Consiste en descargar una cantidad determinada de proyectos, con el propósito de analizar sus dependencias. Se descargaron 30 proyectos, pues era necesario hacer una comparación manual entre ellos, y de acuerdo con el alcance de esta investigación, con sólo 30 fue posible analizar y encontrar posibles mejoras para las búsquedas.

Paso 5: Generación del grafo

Consiste en analizar las relaciones de dependencia entre los proyectos descargados. A partir de esto, se generó un grafo dirigido utilizando cada proyecto como un nodo y las relaciones de dependencia como arcos entre los nodos. Cada una de estas aristas tuvo como peso uno, debido a que solamente se evaluó qué proyecto depende de qué otro proyecto y no el número de veces que lo invocaba.

Paso 6: Análisis del grafo

Consiste en evaluar las características de centralidad de cada nodo para encontrar cuáles debían considerarse más relevantes al momento de ordenarlos. Este análisis permitió encontrar características que ofrecieron la información necesaria para proponer tres posibles formas de ordenamiento, que se explican en la sección 6.

6. Prueba experimental

Paso 1: se decidió usar como dominio el procesamiento de lenguaje natural o NLP (Natural Language Processing), porque se contaba con acceso a un grupo de expertos en un foro especializado. Los expertos corroborarían si el ordenamiento propuesto era el más adecuado.

Paso 2: El repositorio seleccionado para realizar las búsquedas, fue Sourceforge (SourceForge, 2013), dado que contiene más de 300.000 proyectos de software open source que proveen el código fuente necesario para el análisis de esta investigación. La mayoría de estos proyectos cuentan con varios años de tiempo de vida y se mantienen activos desarrollando nuevas versiones hasta la actualidad. Era necesario trabajar con un solo repositorio para poder comparar su método actual de ordenamiento con los métodos propuestos.

Pasos 3 y 4: Se ubicaron y se descargaron 30 proyectos de software del dominio seleccionado en el orden ofrecido por SourceForge. Se definieron 3 términos comúnmente utilizados en este dominio: Natural Language Processing, NLP y Lucene. Se realizaron 7 búsquedas usando las siguientes combinaciones: Natural Language Processing, NLP, Lucene, Natural Language Processing + NLP, Natural Language Processing + Lucene, NLP + Lucene y Natural Language Processing + NLP + Lucene. Estas combinaciones nos ofrecieron los proyectos de software más populares relacionados a NLP. Los proyectos descargados de SourceForge al realizar las búsquedas e encuentran en la Tabla 8**¡Error! No se encuentra el origen de la referencia.**.

Tabla 8. Proyectos descargados de SourceForge

1. Lucene Search Engine	
2. LogicalDOC Document Management – DMS	
3. Regain	4. Open Search Server
5. Compass	6. Rivulet Enterprise Search
7. TML – Text Mining Library for LSA	
8. Oxyus Search Engine	
9. LIUS: Lucene Index Update and Search	
10. Katta	11. Red Piranha
12. Lu Collector	13. Titli
14. Mustru: A QA Search Engine	
15. eXtensible Text Framework: XTF	
16. easyGIS	17. Cralwer
18. JEndX	19. Kneobase
20. Luigi Open Search Engine	
21. irLessons	22. Aggregated Search
23. Archtea	24. Axyl
25. Binghamton University Search (BUS)	
26. Clairv	27. Dias
28. DocInfoRetriever	29. FathomFive
30. FlySearch	

Pasos 5 y 6: Para generar el grafo, se utilizó el software open source Gephi, que permite la visualización y manipulación de grafos (Gephi, 2012).

El grafo dirigido generado a partir de las relaciones que existen entre los proyectos se puede apreciar en la **Figura 30**, cuando aún no ha sido sometido a ningún orden.

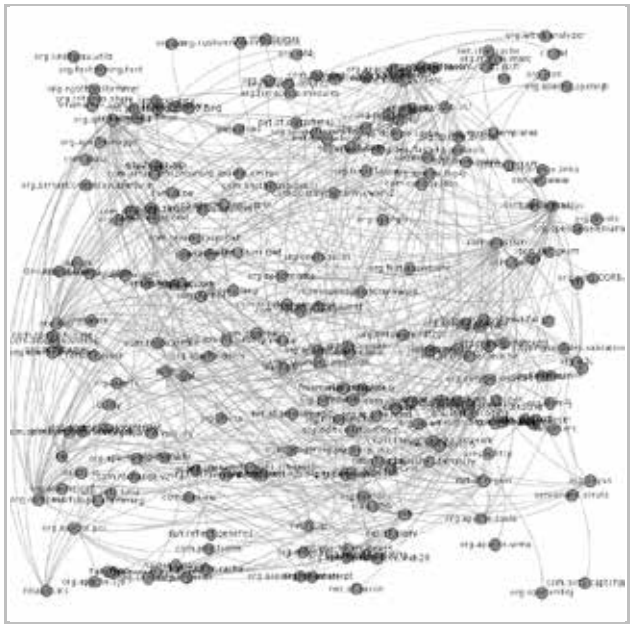


Figura 30. Grafo antes del ordenamiento

A continuación se presentan las medidas de centralidad obtenidas del grafo. Se ha utilizado una escala de colores que va desde crema hasta rojo para lograr el impacto visual de los nodos, siendo rojo el color correspondiente a las medidas mayores. Además, se ha utilizado radios menores y mayores en combinación con la escala de colores, para un mayor refuerzo visual.

El grafo presenta aristas con dirección definida. La dirección de las aristas indica la dependencia de los proyectos. Por ejemplo, dado el nodo A del que sale una arista, para entrar al nodo B. Esta dirección indica que A depende de B.

6.1. Medidas de centralidad

Grados: De acuerdo con esta clasificación los proyectos más conectados aparecen como tal, debido a por lo menos uno de los siguientes motivos:

Gran cantidad de proyectos de software dependen de ellos. Esto se mide en el grafo como grados de entrada.

Dependen de una gran cantidad de proyectos de software. Este tipo de conexiones se mide como grados de salida.

Los grados no pueden usarse como medida de relevancia debido a que no diferencian entradas y salidas. Si se toma el total de conexiones como medida, esto haría más relevante un proyecto no sólo cuando aparece frecuentemente como dependencia, sino también cuando tiene una gran cantidad de proyectos de los que depende.

Grados de entrada: Se denomina grados de entrada al número de conexiones entrantes con las que cuenta un nodo en un grafo dirigido.

Los grados de entrada de un nodo indican qué proyectos dependen del proyecto correspondiente a dicho nodo. Por lo tanto, cuanto mayor sea la medida de grados de entrada de un proyecto, éste es invocado con mayor frecuencia volviéndose más importante para todo el conjunto en general. Se concluye entonces que los grados de entrada son un importante factor de relevancia en el conjunto de proyectos.

Grados de salida: Los grados de salida de un nodo indican de qué proyectos depende el proyecto correspondiente a dicho nodo. Por lo tanto, cuanto mayor sea la medida de grados de salida de un proyecto, éste se vuelve más dependiente de otros proyectos. El hecho de que un proyecto sea muy dependiente no lo vuelve más relevante, porque no aporta nada a los proyectos de los que depende. Se concluye entonces que los grados de salida no son una medida apropiada para la relevancia de los proyectos.

Excentricidad: La excentricidad no presenta ningún patrón que permita sacar conclusiones respecto a los proyectos más excéntricos, por lo tanto no aporta ninguna información respecto a la importancia entre los proyectos.

Intermediación: En el grafo (**Figura 31**), el nodo Lucene aparece como el más intermedio, pues conecta todos los conjuntos de nodos al ser la dependencia más frecuente de los proyectos. El segundo más intermedio es LIUS, pues permite a algunos nodos llegar hasta Lucene.

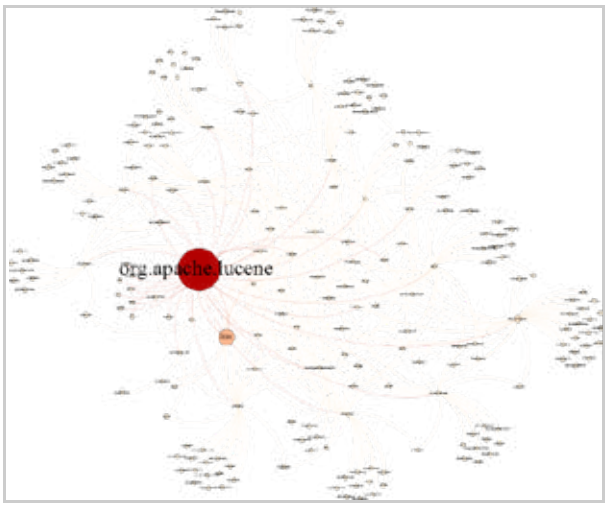


Figura 31. Grafo en función de la intermediación

La intermediación es una medida útil para la relevancia porque es indicador de la cantidad de dependencias de un proyecto y a la vez de la relevancia de sus dependencias.

Cercanía: Los proyectos con pocas dependencias y aquellos con pocos o ningún proyecto que dependa de ellos son los más cercanos. Sin embargo, esta información no aporta ninguna relevancia a los proyectos.

Vector propio: En el grafo (**Figura 32**) se aprecia que los nodos grandes (color rojo) como Lucene Search y LIUS, cumplen con ambas condiciones, mientras que los nodos medianos (color melón) como FlySearch, cumplen una de las 2 condiciones, quedando en tamaño menor y casi inapreciable el resto de nodos (color crema) que cumplen en menor medida alguna o ninguna de las 2 condiciones.

Éste es un buen indicador de relevancia, una vez que se ha tenido en cuenta el grado de entrada de los nodos, porque le da importancia no sólo al grado del nodo sino también a los grados de los nodos con los que está conectado.

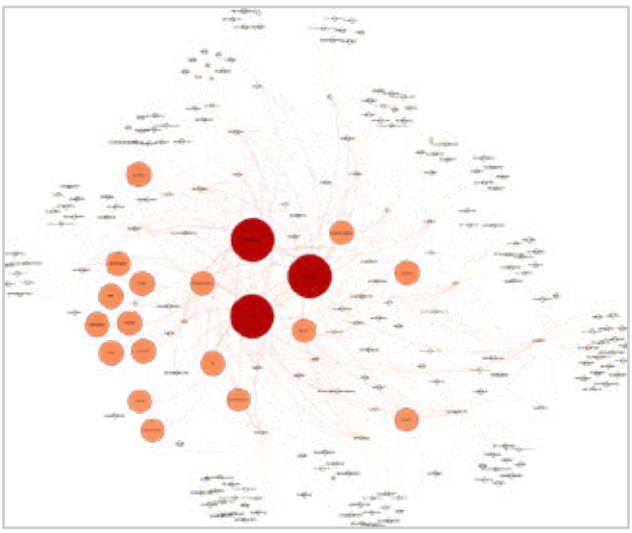


Figura 32. Grafo ordenado en base al Vector Propio

6.2. Métodos de posicionamiento

Basándose en el grafo generado, se plantearon tres metodologías distintas de posicionamiento, con el objetivo de contrastarlas con el posicionamiento presentado por el repositorio de software SourceForge en las búsquedas realizadas, y se evaluó si es que se logró algún tipo de mejora. A continuación se presentan las tres metodologías evaluadas

6.2.1. Posicionamiento en base a la centralidad

Esta propuesta consiste en considerar los siguientes factores para el ordenamiento:

- El grado de entrada: porque es el indicador principal de lo dependiente que es todo o parte del grafo de uno o más nodos en particular.
- El vector propio: porque le da una valoración ponderada mayor a los nodos que están altamente conectados. Esto permite definir, con mayor detalle, la importancia de las dependencias entre nodos.
- La intermediación: porque participa en las comunicaciones entre proyectos que no dependen directamente entre sí, estos proyectos cumplen la función de mediadores entre conjuntos de otros proyectos, que de otro modo no tendrían conexión alguna.

Este ordenamiento propuesto se basa en buscar los nodos con mayor grado de entrada, si es que coinciden en este valor, se comparan sus valores de vector propio, y finalmente, si este también coincidiera, se compara sus valores de intermediación.

6.2.2. HITS: Hubs y Authorities

Se decidió usar el método HITS porque es uno de los algoritmos de ordenamiento más populares y relevantes que existen en la actualidad. En la **Figura 33**, se pueden

apreciar los resultados obtenidos analizando el grafo de authorities según el método HITS.

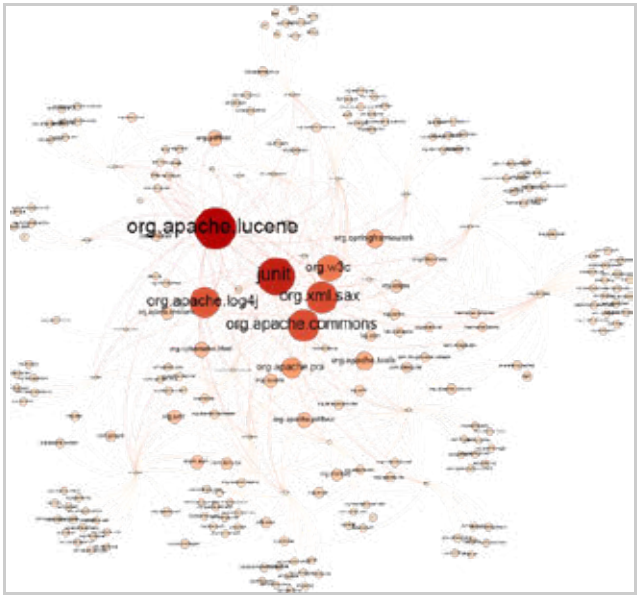


Figura 33. Grafo ordenado mediante HITS

6.2.3. PageRank

Se seleccionó el algoritmo PageRank debido a que es el algoritmo usado por el motor de búsqueda de Google para ordenar sus resultados. La **Figura 34** muestra el grafo resultante.

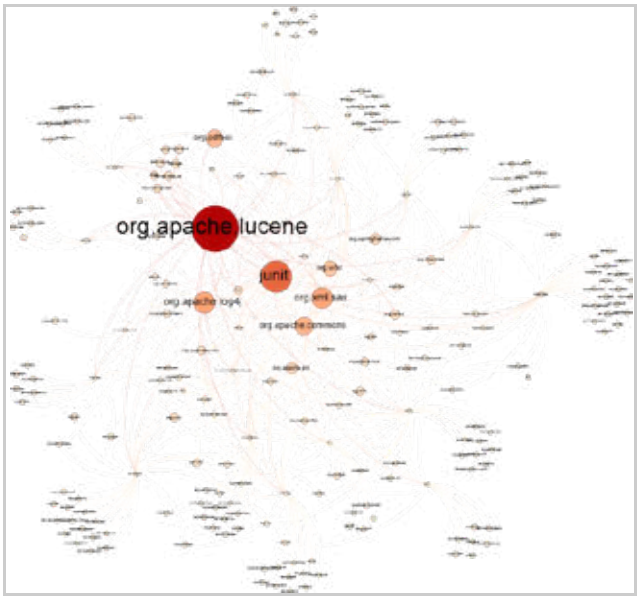


Figura 34. Grafo ordenado mediante PageRank

7. Análisis de los resultados

La Tabla 9 y la Tabla 10 muestran los resultados de posicionamiento obtenidos con los 3 métodos evaluados y se comparan con el ordenamiento ofrecido por el repositorio SourceForge.

Tabla 9. Ordenamiento de SourceForge vs Centralidad

SourceForge	Propuesta 1: Centralidad
Lucene Search Engine	Lucene Search Engine
LIUS	LIUS
TML	Compass
Compass	TML
LogicalDOC	LogicalDOC
Luigi Open Search Engine	Luigi Open Search Engine
Mustru: A QA Search Engine	Mustru: A QA Search Engine
Open Search Server	Open Search Server
Oxyus Search Engine	Oxyus Search Engine
Red Piranha	Red Piranha
Regain	Regain
Rivulet Enterprise Search	Rivulet Enterprise Search
Titli	Titli
Aggregated Search	Aggregated Search
Kneobase	Kneobase
Clairv	Clairv
DocInfoRetriever	DocInfoRetriever
easyGIS	easyGIS
XTF	XTF
FathomFive	FathomFive
HIT Search Engine	HIT Search Engine
My	my
Hibernate Search	Hibernate Search
irLessons	irLessons
Katta	Katta
Archtea	Archtea

Tabla 10. Ordenamiento de HITS vs PageRank

Propuesta 2: HITS	Propuesta 3: PageRank
Lucene Search Engine	Lucene Search Engine
LIUS	LIUS
TML	Compass
Compass	TML
LogicalDOC	LogicalDOC
Luigi Open Search Engine	Luigi Open Search Engine
Mustru: A QA Search Engine	Mustru: A QA Search Engine
Open Search Server	Open Search Server
Oxyus Search Engine	Oxyus Search Engine
Red Piranha	Red Piranha
Regain	Regain
Rivulet Enterprise Search	Rivulet Enterprise Search
Titli	Titli
Aggregated Search	Aggregated Search
Kneobase	Kneobase
Clairv	Clairv
DocInfoRetriever	DocInfoRetriever
easyGIS	easyGIS
XTF	XTF
FathomFive	FathomFive
HIT Search Engine	HIT Search Engine
My	my
Hibernate Search	Hibernate Search
irLessons	irLessons
Katta	Katta
Archtea	Archtea

Una comparación entre las 4 propuestas se muestra en la Figura 35.

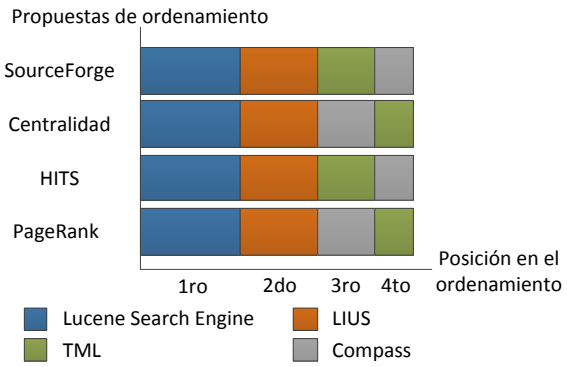


Figura 35. Comparación de los resultados

Consulta a expertos

Paralelamente, se recurrió a un grupo de expertos con conocimiento y experiencia en el dominio definido (NLP). La opinión de los expertos se recogió mediante una encuesta anónima a través de Internet, que tomaba los 16 proyectos más destacados del experimento y los agrupaba aleatoriamente en grupos de 4. La encuesta fue anónima para poder conseguir la mayor cantidad de respuestas posibles. Se solicitaba a los usuarios colocar cada proyecto en orden de importancia respecto a los demás proyectos en su grupo (Ruiz, 2013).

El enlace de acceso a esta encuesta se publicó en un foro especializado en NLP: Lucene – Java – Nabble, con el propósito de recoger la opinión de los expertos que interactúan en dicho foro. Se recogieron 8 opiniones, estos resultados se utilizaron para compararlas con cada una de las alternativas propuestas de ordenamiento, así como el ordenamiento presentado por SourceForge.

A continuación se presentan los resultados:

- De acuerdo con las opiniones recogidas, al igual que en SourceForge y en las alternativas propuestas, el proyecto destacado como el más importante de todos es Lucene Search Engine. En este caso se puede afirmar que tanto SourceForge como las alternativas propuestas son correctas en el ordenamiento que plantean.
- El segundo proyecto más importante, según las opiniones, es Katta. En este caso, los expertos no coinciden con SourceForge ni con las alternativas propuestas.
- De acuerdo a los expertos, el tercer proyecto más importante es Compass. Este resultado es muy similar a las alternativas propuestas, pero no se asemeja al ordenamiento de SourceForge.
- El cuarto proyecto más importante es Open Search Server. En este caso, la opinión de los expertos se asemeja más a SourceForge que a las alternativas propuestas.
- El resto de los proyectos evaluados no eran conocidos para los expertos, por lo que no fueron tomados en cuenta para las comparaciones.

8. Conclusiones y recomendaciones

Los grafos constituyen una alternativa de mucha utilidad al momento de analizar las relaciones entre proyectos, pero solamente permiten analizar aquellos que hayan sido referenciados como dependencia por otros proyectos.

El ordenamiento de SourceForge no coincide con la opinión de los expertos, por lo que es razonable considerar que el repositorio podría no estar utilizando el método más adecuado de ordenamiento para presentar los proyectos de software.

La opinión de los expertos, coincide con el método propuesto en 3 de los 4 proyectos que se seleccionaron, esto indica que es posible plantear un método de posicionamiento más inteligente, que complemente a los métodos usados por los repositorios como SourceForge.

Los ordenamientos presentados por los métodos propuestos, se asemejan en varios casos a la opinión de los expertos, lo que abre la posibilidad del ordenamiento basado en grafos.

Al utilizar grafos para el posicionamiento es preferible trabajar principalmente con proyectos altamente referenciados, de manera que todos los nodos ofrezcan información que les permita diferenciarse unos de otros.

Para probar estadísticamente las propuestas planteadas, es necesario trabajar con una muestra mucho mayor de proyectos.

Para una mejor validación, es necesario consultar a una mayor cantidad de expertos. Si es posible, acudir a personas de renombre en el área de estudio que ofrezcan formalmente su opinión sobre las propuestas planteadas.

9. Trabajos futuros

Extender la investigación para abarcar otros tipos de proyectos y otros repositorios de software open source e incrementar la cantidad de proyectos analizados, con el fin de validar estadísticamente la hipótesis planteada.

Otra alternativa de posicionamiento a analizar, podría basarse en *Sentiment Analysis*, es decir, recurrir a la opinión de expertos y desarrolladores que utilizan estos proyectos, información que puede ser extraída de blogs de expertos, foros especializados y redes sociales (customer THINK, 2010) (Pang & Lee, 2008).

Una tercera alternativa sería la combinación de un análisis basado en grafos y *Sentiment Analysis*.

Referencias bibliográficas

Bang-Jensen, J., & Gutin, G. (2007). *Digraphs Theory, Algorithms and Applications.* Springer-Verlag.

Bavelas, A. (1958). Patrones de comunicación en grupos orientados a la tarea.

Chris DiBona, S. O. (1999). *Open Sources: Voices from the Open Source Revolution.* O'Reilly Media.

customer THINK. (11 de Marzo de 2010). *customer THINK*. Recuperado el 28 de Enero de 2013, de think. feel. connect:

http://www.customerthink.com/blog/sentiment_analysis_hard_but_worth_it

Deshpande, A., & Riehle, D. (2008). The Total Growth of Open Source. *Fourth Conference on Open Source Systems (OSS 2008).*

Fiedler, M. (1975). A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. *Czechoslovak Mathematical Journal*, 25 (100).

Gephi. (2012). *Gephi, an open source graph visualization and manipulation software*. Obtenido de Gephi: https://gephi.org/

GitHub.com. (2012). *GitHub*. Recuperado el 28 de Enero de 2013, de Build software better, together: https://github.com/about

Google. (2013). *Google Code*. Recuperado el 28 de Enero de 2013, de http://code.google.com/intl/es/

Gross, J. L., & Yellen, J. (2003). *Handbook of Graph Theory.* Taylor & Francis.

Kleinberg, J. M. (1998). Authoritative Sources in a Hyperlinked Environment. *ACM-SIAM Symposium on Discrete Algorithms.*

Lakhani, K., & Wolf, R. G. (Setiembre de 2003). *Why Hackers Do What They Do: Understanding Motivation and Effort in Free/Open Source Software Projects.* Recuperado el Marzo de 2013, de Social Science Research Network: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=443040

Microsoft. (22 de Enero de 2012). *CodePlex*. Recuperado el 28 de Enero de 2013, de Project Hosting for Open Source Software: http://www.codeplex.com/site/help

Newman, M. E. (13 de Mayo de 2005). *The mathematics of networks.* Obtenido de Center for the Study of Complex Systems, University of Michigan: http://www-personal.umich.edu/~mejn/papers/palgrave.pdf

O'Reilly, T. (2012). *O-Reilly ONLamp*. Recuperado el 13 de Marzo de 2013, de http://onlamp.com/onlamp/2005/09/15/what-is-opensource.html

Page, L., & Brin, S. (1998). *The PageRank Citation Ranking: Bringing Order to the Web.*

Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval vol.2 nos.1-2*, 1-135.

Pitkow, J. E. (1997). *Characterizing World Wide Web Ecologies.* Georgia Institute of Technology.

Royal Pingdom. (4 de Enero de 2011). *Ramblings from the Pingdom team about the Internet and web tech*. Recuperado el 13 de Enero de 2013, de Royal Pingdom: http://royal.pingdom.com/2011/01/04/apache-web-server-hit-a-home-run-in-2010/

Ruiz, J. (1 de Febrero de 2013). *Survey to measure the utility of Lucene's projects*. Recuperado el 13 de Marzo de 2013, de Survey Gizmo: http://www.surveygizmo.com/s3/1150178/Survey-to-measure-the-utility-of-Lucene-s-projects

SourceForge. (11 de Diciembre de 2012). *Forge / Documentation / Relevance*. Obtenido de SourceForge: http://sourceforge.net/p/forge/documentation/Relevance/

SourceForge. (2013). *SourceForge*. Recuperado el 28 de Enero de 2013, de http://sourceforge.net/about

Spertus, E. (1997). Parasite: Mining structural information on the web. *Sixth International WWW Conference.* Santa Clara.

StatCounter. (1 de Enero de 2013). *Global stats*. Recuperado el 13 de Enero de 2013, de StatCounter: http://gs.statcounter.com/#browser-ww-monthly-201112-201212-bar

Weber, S. (2003). *The Success of Open Source.* Recuperado el 31 de 03 de 2013, de Harvard University Press: http://brie.berkeley.edu/research/SW%20Ch.1%20Dec02.pdf

Weiss, R., Vélez, B., Sheldon, M. A., Manprempre, C., Szilagyi, P., Duda, A., & Gifford, D. K. (1996). HyPursuit: A hierarchical network search engine that exploits content-link hypertext clustering. *7th ACM Conference on Hypertext* (págs. 180-193). New York: ACM Press.