# Recomendación de contenido en Twitter

Lizbeth Contreras, Samuel Rocha, Alain Cabrera

27 de mayo de 2016

**Resumen**

En este documento se propone un sistema de recomendación de contenido en Twitter basado en la actividad de la red del usuario. En primera instancia, se obtuvieron los tweets compartidos de las personas que se siguen y por medio de algoritmos de minería de textos como Rapid Automatic Keyword Extraction y Text Rank se obtuvieron los temas principales de dichos tweets; en segunda instancia, se colocaron en la base de datos neo4j; finalmente, con Cypher se hicieron recomendaciones con respecto a los link compartidos por los seguidores del usuario.

## Introducción y Motivación

En la actualidad, el problema de recomendación -películas, música, productos y contenido- ha adquirido gran importancia. Las empresas invierten cada vez más fondos en la investigación y desarrollo de un sistema de recomendación para optimizar sus utilidades.
En nuestro caso, buscamos crear un sistema de recomendación de artículos de lectura para los usuarios de Twitter. En especial los artículos compartidos por el círculo inmediato en la red social.
Para el desarrollo del sistema de recomendación se utilizó software libre: Python y Neo4j.

## Metodología

### Obtención de datos

Los datos usados para crear el sistema de recomendación de contenido fueron obtenidos de Twitter por medio de la librería Tweepy de Python y de la API de la red social. Nuestro interés se centra en links compartidos por nuestros *followers*. Lo que se hizo fue obtener una lista de *followeds*, para cada uno de ellos se obtuvieron 200 twitts y se encontraron todos los Urls compartidos en ellos. Los links obtenidos se almacenan en un archivo csv junto con el nombre

del usuario que lo compartió.

## Análisis de contenido

Una vez obtenidos los Urls, el siguiente paso es tener una idea del contenido de dichas direcciones (supondremos que cada dirreción alberga un artículo de noticia). Con este fin, compararemos el desempeño de dos métodos de extracción de palabras clave contra la paquetería *newspaper* de Python.

### Rapid Automatic Keyword Extraction

RAKE es método de extracción de palabras clave extremadamente eficiente, opera en documentos individuales para permitir la aplicación a las colecciones dinámicas, se aplica fácilmente a nuevos dominios, y funciona bien en múltiples tipos de documentos, en particular los que lo hacen no seguir las convenciones gramaticales específicos.
Este método se basa en el hecho de que las palabras clave frecuentemente contienen varias palabras y raramente contienen signos de puntuación o *stopwords*, como las palabras funcionales *y, el, de* u otras palabras con significado léxico mínimo. Las *stopwords* típicamente no se incluyen en los análisis de texto, ya que se considera que son poco informativos o sin sentido. Este razonamiento se basa en la expectativa de que tales palabras son demasiado frecuentes y ampliamente utilizado para ayudar a los usuarios en sus análisis o tareas de búsqueda.
RAKE utiliza las *stopwords* y frases delimitadores para dividir el texto en palabras clave candidatos, que son secuencias de palabras de contenido a medida que ocurren en el texto. La coocurrencias de palabras dentro de estos candidatos son significativos y nos permiten identificar coocurrencia de palabras sin la necesidad de una ventana arbitraria. asociaciones de palabras se miden por lo tanto de una forma que se adapta automáticamente al estilo y contenido del texto, lo que permite la medición de adaptación y de grano fino de coocurrencias de palabras que se utilizarán para anotar palabras clave candidatos.

### TextRank

TextRank es un modelo de rankeo basado en grafos para el procesamiento de textos. Este método se ha utilizado para la extracción no supervisada de palabras claves y sentencias.
Para poder utilizar los algoritmos de rankeo basado en grafos a textos, tenemos que construir una gráfica que represente el texto e interconectar las palabras u otros elementos del texto con relaciones significativas.
El primer paso del algoritmo es tokenizar el texto y taggearlo con POS, se consideran las palabras solas como candidatas para añadir al grafo como nodos (palabras clave con múltiples palabras son reconstruidas en la fase de post procesamiento). Un arista es añadida entre los nodos si éstos coocurren en una ventana fija. Asignamos un *score* inicial igual a 1 a cada vértice y utilizamos

algún método de rankeo para grafos.
Posteriormente, los vértices se clasifican en orden inverso al de su *score*, y las palabras con puntuación mayor (aproximadamente un tercio) se utilizan para el post procesamiento.
Marcamos las palabras candidatas en el texto y las secuencias de palabras clave adyacentes son colapsadas en una palabra clave múltiple.

#### newspaper

Python cuenta con la librería newspaper, la cual permite obtener el autor y un conjunto de *keywords* de un artículo.

#### Comparación de métodos

Con el fin de elegir el método que se empleará para obtener las *keyword* de los artículos se realizará una comparación del desempeño de los tres algoritmos. Se utilizó un corpus compuesto de 183 documentos en formato de texto. Dichos documentos fueron taggeados por más de 330 usuarios, voluntariamente, en el sitio citeulike.org.
No fue posible analizar 4 de los documentos utilizando el método TextRank pues resulta muy costoso computacionalmente, por ellos se descartaron dichos documentos.
El desempeño de los algoritmos en los 179 documentos restantes fue muy similar. Para cada documento, se tomaron 5 *keywords* por método. Estos *keywords* se contrastaron con los *tags* y se obtuvieron los siguientes porcentajes de acertividad:

- Text Rank: 100 %
- RAKE: 99.7 %
- *newspaper*: 97.2 %

Elegiremos utilizar la paquetería *newspaper* para seguir con la construcción de nuestro sistema de recomendación, puesto que los otros algoritmos presentan las siguientes desventajas: Como se mencionó anteriormente, Text Rank es computacionalmente costoso y por ello no es idóneo para emplearse en un sistema de recomendación en línea; dado que nuestro sistema de recomendación será de contenido en español, necesitamos una lista de *stopwords* adecuada para nuestro idioma para poder utiliar el algoritmo RAKE.

## Modelo en Neo4j

El modelo de datos utlizado consta de cuatro tipos de nodos (user, url, author, keyword) y tres tipos de aristas que describen la relación entre los nodos.
Se utilizó la librería py2neo de Python para conectar nuestra instancia de Neo4j y se agregaron los datos por medio de consultas de Cypher.

En la siguiente figura se puede observar los nodos conectados por 3 aristas: Cada usuario que comparte alguna Url está escrito por cierto autor y habla sobre algún tema.

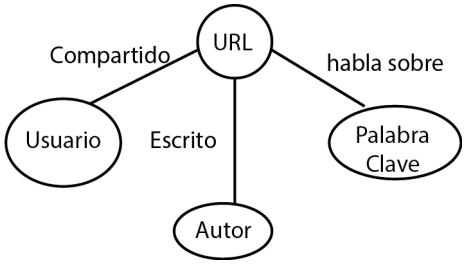


Figura 1: Diagrama general del grafo

Después de determinar el modelo que se iba a utilizar para poder determinar las Url compartidas por los usuarios, se importaron a la base de datos de Neo4j, pero en una prueba que se realizó para un usuario que tenía seguía a 2000 personas se alcanzaba el rate limite, ya que se checaban 200 twitts de cada usuario y se checaba si compartía un artículo. Decidimos usar una cuenta que seguía a 80 personas para realizar un grafo de menor tamaño, pero la idea es escalable para todos los usuarios. A continuación se muestra el grafo ejemplo de 10 usuarios que han compartido artículos.

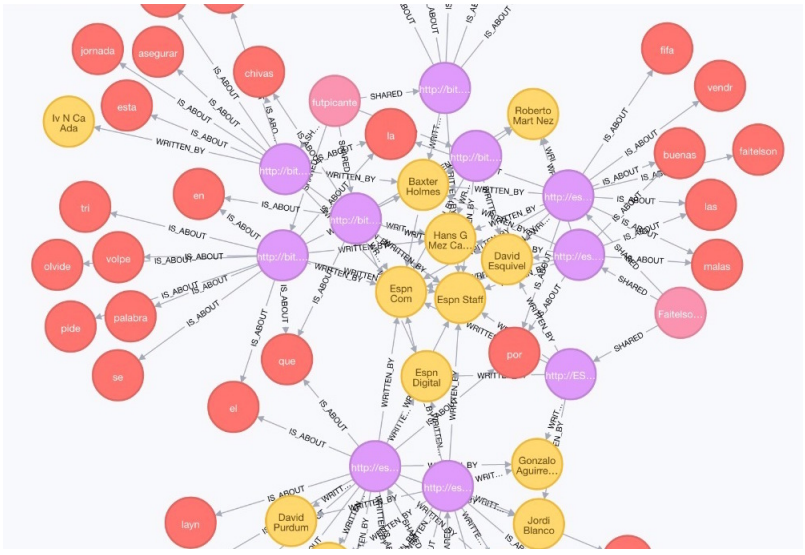


Figura 2: Diagrama general del grafo

## Sistema de Recomendación

En la figura 3 se puede observar el diagrama del usuario jgomezjunco( en color amarillo) que representa el grafo de los últimos articulos compartidos(en color rojo) al igual que las palabras más importantes del artículo(en color rosa). El objetivo está en determinar, dados los artículos que compartió, extraer las palabras clave para determinar cuales son los artículos compartidos por los usuarios que sigue y los usuarios que lo siguen, que otras lecturas le serían factibles leer contando su gusto demostrado previamente.

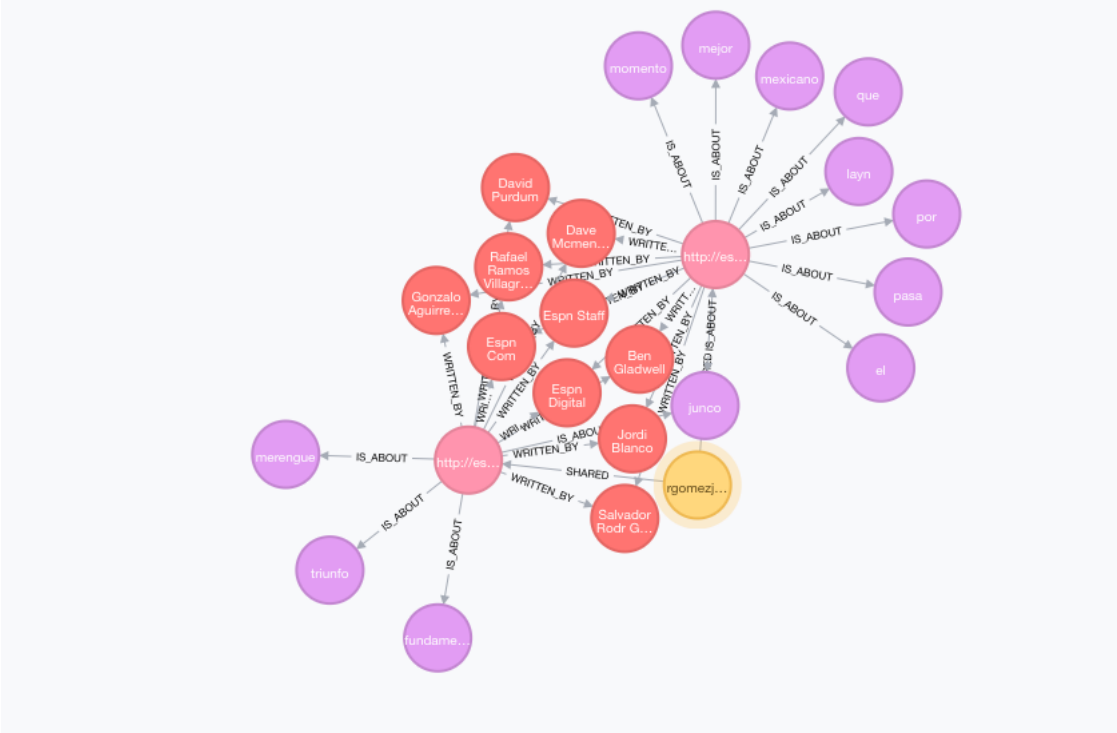


Figura 3: Diagrama de un usuario con los artículos que ha compartido

Si se analiza previamente el grafo, podemos observar que tiene una tendencia a compartir árticulos deportivos que, dependiendo a las personas que siga, se puede determinar que artículos serán de su agrado. Ésto tiene sentido, ya que el usuario que se decidió analizar es un conductor deportivo.

## Extracción de palabras clave

La descripción de los artículos se hizo con una librería de python que se llama newspaper extrae las palabras más importantes de los artículos. En primera

instancia, se obtienen todas las palabras, frases o conceptos que potencialmente pueden ser palabras clave; despues, por cada candidato se calculan sus propiedades que indican si pueden ser una palabra clave; finalmente se hace una puntuación y se selección las palabras clave: normalmente se utilizan técnicas de aprendizaje de máquina para determinar la puntuación que describe la probablidad de que sean escogidas.
Para acelerar el proceso de selección se utilizan varios parámetros como utilizar la frecuencia mínima de los candidatos, el número máximo y mínimo de las palabras y se pueden *stemmatizar* (proceso de sacar la raiz de una palabra).

### Secuencias de Cypher

Ya que se cuenta con un grafo de un usuario, sus seguidores y los artículos que han compartido, se busca cuales son los artículos que fueron compartidos por los la red que serían de gran utilidad dependiendo del historial de usuario.

```
MATCH (u:User {username: 'samo2704'})-[:SHARED]->(url:URL)
MATCH (url)-[r:IS_ABOUT]->(kw:Keyword)<-[r2:IS_ABOUT]-(u2:URL)
WHERE NOT (u)-[:SHARED]->(u2)
WITH u2.url as article, count(r2) as score
RETURN article, score ORDER BY score DESC LIMIT 10;
```

Figura 4: Instrucción Cypher para la recomendación a un usuario

Como se puede observar en la instrucción, buscamos a un usuario en especial y todas los articulos que ha compartido. De esta búsqueda queremos saber las palabras clave de las páginas que ha compartido para poder compararlas con los usuarios que sigue.

Como se puede observar en la anterior tabla, aparecen varios links que se recomiendan al usuario por su gusto en deportes y que 2 de las personas que sigue han compartido.

El único inconveniente que apareció fue que probamos esto para nuestras cuentas de twitter y solo una de ellas no alcanzaba el limite otorgado por twitter. El análisis realizado es escalable, pero solo se realizó la recomendación para algunos usuarios.

## Conclusiones

Una de las desventajas de dar una recomendación en relación a los Url que se han compartido es que los nuevos usuarios de Twitter o gente que no ha compartido muchas cosas, no tendrá recomendaciones, ya que no tendrá tópicos que se puedan acomodar a las recomendaciones que pudieran hacerce.

| article | score |
|---|---|
| http://bit.ly/1RRt2j3 | 2 |
| http://ow.ly/10bz3L | 2 |
| http://bit.ly/1opUBSF | 2 |
| http://ow.ly/106sDM | 2 |
| http://ow.ly/10bELy | 1 |
| http://es.pn/1RsY4vz | 1 |
| http://bit.ly/22lvi15 | 1 |
| http://somosinvictos.com/2016/03/21/ni-como-cr7-ni-como-messi-el-hijo-de-david-barral-delantero-espanol-quiere-ser-como-chicharito/ | 1 |
| http://bit.ly/1SnoDyL | 1 |
| http://bit.ly/1qmvFx5 | 1 |

Returned 10 rows in 148 ms.

Figura 5: Tabla de las recomendaciones para un usuario

Fue muy interesante poder practicar las herramientas que vimos en clase como poder bajar datos de twitter, poder convertirlos en una base de datos de grafos y finalmente utilizar secuencias de cypher y poder analizar los datos que se bajaron. En general, no es tan fácil utilizar las secuencias de Cypher, pero después de un tiempo de utilizarlas, fue más sencillo realizar las peticiones.
Otro problema con el que nos enfrentamos fue el limite que te pone twitter para bajar datos. Como recorriamos muchos de los post que ponían los usuarios, el árbol de petición crecía exponcialmente en relación a los seguidores. El grafo final lo realizamos con una cuenta que solo seguía a 80 personas y analizamos dicho grafo.
Para posteriores análisis, se puede implementar análisis de sentimientos y determinar si los artículos recomendados son positivos y negativos.

## Referencias


[1] Panzarino, Onofrio, 'Learning Cypher', *Packt Publishing Ltd.* (2014).

[2] Robinson, Ian and Webber, Jim and Eifrem, Emil, 'Graph Databases: New Opportunities for Connected Data', *O'Reilly Media, Inc.* (2015).

[3] Blei, D., Lafferty, J. (2009). Text mining: Theory and applications, chapter topic models.

[4] Mihalcea, R., Tarau, P. (2004, July). TextRank: Bringing order into texts. Association for Computational Linguistics.

[5] O. Medelyan. 2009. Human-competitive automatic topic indexing. PhD thesis. Depar tment of Computer Science, University of Waikato, New Zealand.

[6] O. Medelyan, E. Frank, I. H. Witten. 2009. Human-competitive tagging using automatic keyphrase extraction. To appear in Proc. of the Internat. Conference of Empirical Methods in Natural Language Processing, EMNLP-2009, Singapore.