Recomendador de libros

Álvaro Angulo, Lucía Manzorro, Juan José Martín, Álvaro Sánchez

31 de mayo de 2017

Introducción

Para realizar el sistema de recomendación lo primero es obtener la base de datos a partir de la cual se construye. En este caso hemos utilizado una base de datos de libros completos (htttp://www.leemp3.com/disponibles_textos.htm). Hemos desarrollado un recomendador que a partir de las similitudes entre los textos, junto a sus metadatos, busca posibles libros que interesen al usuario. A su vez hemos generado una visualización (mediante gephi) que nos permite identificar distintas agrupaciones de libros en función de sus similitudes.

Desarrollo del recomendador

Extracción de los datos

En primer lugar hemos adecuado los datos de esta base de datos realizando scraping. Por una parte nos hemos quedado con los textos completos (guardados en la carpeta *libros*) y por otra parte nos hemos creado un diccionario con los metadados asociados a cada texto (fichero *scraping.py*).

Recomendador

Este recomendador es una adaptación del script de clase para la recomendación de nuestros libros.

- Seleccionamos un número de libros (con los cual trabajaremos posteriormente) y la función *crearModeloSimilitud* nos indica una serie de libros similares a cada uno.
- A partir de la anterior creamos la función *crearModeloSimilitud_libro* la cual te devuelve los libros similares al que reciba de entrada.
- Por último nuestro *recomendador* recibe una serie de libros a través de sus identificadores (serían los que le gustan a un usuario). Obtiene de los metadatos el autor y el género de cada libro de dicha lista. A partir de éstos, filtra por autor y género los libros de nuestra base de datos. Si existiese alguno de estas características, nuestro recomendador realizaría el tf-idf con esta serie de libros, devolviéndonos el más similar de esa lista. En caso contrario, haría el tf-idf de todos los libros de la base de datos y nos devolvería el más similar de ellos.

Para realizar este recomendador se han creado las funciones: *crearModeloSimilitud_todos*, *crearModeloSImilitud_recomendador* y *recomendador* (fichero funciones.py).

Nota: Para crear las similitudes hemos tomado un fragmento de la totalidad del texto para reducir complejidad.

Posibles mejoras

Nuestro recomendador restringe la búsqueda a libros que tengan autor y género igual a los que recibe de entrada (en caso de que los encuentre en la base de datos seleccionada). Luego una posible mejora sería combinar el resultado de la valoración de los libros con estas características con el resultado del tf-idf de la

base de datos elegida, de esta forma, valoraría positivamente la coincidencia con los autores y géneros de los libros que se le indica a la función frente a los demás.

Visualización

Dada la dimensionalidad de nuestro espacio sería interesante buscar algún modo de representar nuestros elementos de forma que se apreciase cómo se agrupan dichos elementos en función de sus similitudes. Una posible opción sería hacer un análisis de clustering, pero una de las charlas a la que asistimos durante esta asignatura nos mostraron una forma de llevar a cabo esta tarea mediante el uso de grafos. Cada elemento se ha conectado con sus similares de forma que a mayor grosor de la arista, mayor similitud.

En el fichero *visualizacion.py* se indica cómo generar los datos necesarios para luego utilizarlos en gephi.

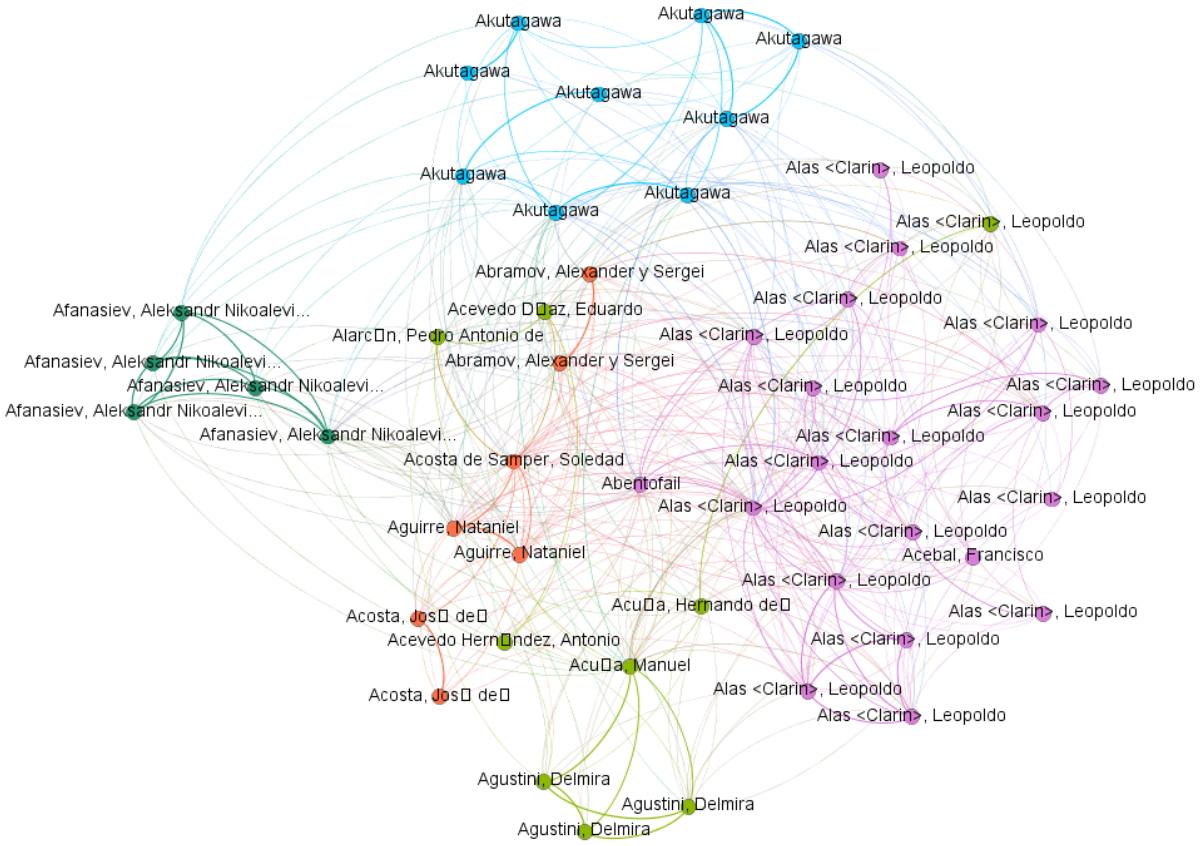

Figure 1: Grafo con 50 libros

Hemos seleccionado 50 libros a los cuales les hemos aplicado el tf-idf, mediante las similitudes obtenidas hemos generado el grafo. Una vez hecho esto, hemos analizado la modularidad del grafo detectando varias familias. En dichas familias, vemos cómo se agrupan los libros del mismo autor.

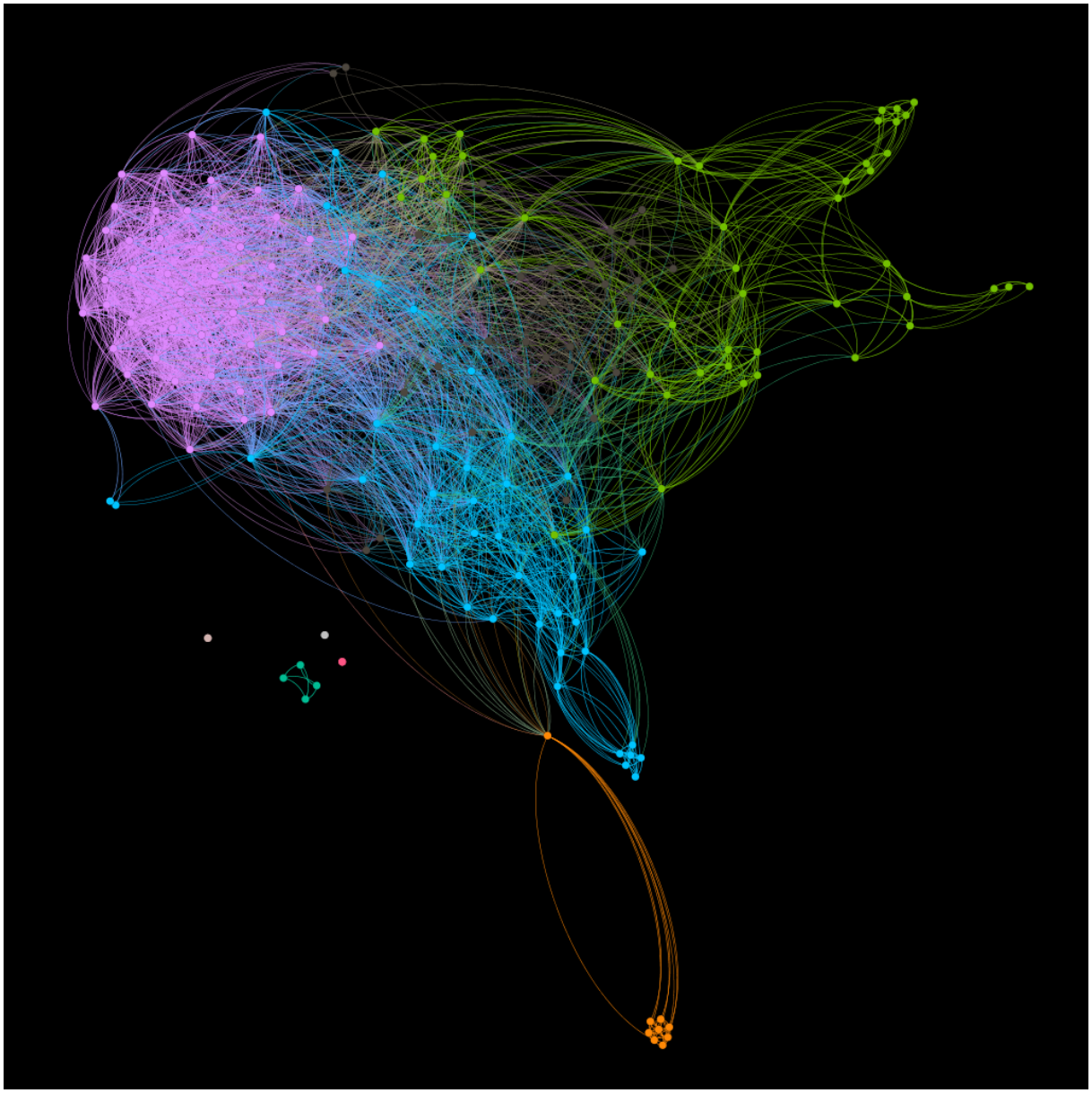

Figure 2: Grafo con 200 libros

Hemos realizado una visualización con un mayor número de elementos en el que se aprecian principalmente cuatro grupos.