Sistema de Recomendación de Librerías de Código Fuente basado en Tf-idf y Similitud por Coseno

Amaury Hernández Águila
Instituto Tecnológico de Tijuana
Maestría en Ciencias Computacionales
amherag@gmail.com

Resumen

Se creó un sistema de recomendación de librerías de código fuente de Common Lisp. El sistema también está hecho en Common Lisp. Se utilizan los métodos tf-idf (Term frequency - Inverse document frequency) y similitud de cosenos para comparar los códigos fuentes y las documentaciones entre paquetes (librerías). Además se utiliza una adaptación del tf-idf y similitud de cosenos que calcula la similitud entre dos paquetes tomando en consideración el uso de otras librerías dentro de este paquete. El resultado del sistema de recomendación es una lista de paquetes similares al paquete del usuario junto con las librerás recomendadas.

Palabras clave: sistema de recomendación, similitud de código fuente, tf-idf, similitud por coseno, metalambda

1. Introducción

La mayoría de los sistemas cuyo enfoque es encontrar la similitud entre códigos fuente tienen como finalidad la detección de plagiarismo. En este trabajo se presenta una forma de utilizar estas medidas de similitud para hacer un sistema de recomendación de código fuente, específicamente de librerías de Common Lisp.

Un desarrollador puede utilizar este sistema de recomendación para encontrar paquetes que sean similares al suyo, o paquetes que podrían ser de utilidad para su proyecto.

La utilidad de un sistema de recomendación como este sólamente se presenta si se tiene a nuestra disposición una gran colección de librerás las cuales podamos procesar fácilmente. Especialmente, nuestro sistema de recomendación necesita una forma de determinar de qué librería proviene cualquier definición (funciones, variables, etc.) siendo usada en un paquete. Esto no presenta un problema para nuestro sistema de recomendación, ya que Common Lisp nos da la posibilidad de obtener el paquete de donde provienen los símbolos en un programa.

El sistema de recomendación utiliza el promedio de tres métricas para determinar la similitud entre un paquete y otro. De cada paquete se extrae las documentaciones de sus funciones, su código fuente excluyendo las documentaciones de sus funciones, y la usabilidad de cada librería en el paquete. Éste último se refiere a la cantidad de veces que un paquete está usando alguna función, método, clase, variable, etc. (o símbolo, en términos de Common Lisp) de cierta librería; entre mayor sea la cantidad de símbolos externos siendo usados en el paquete, mayor será la usabilidad de esa librería dentro de ese paquete.

La similitud entre documentaciones y código fuente de los paquetes se obtiene utilizando una implementación de los métodos tf-idf y similitud de cosenos. Utilizando tf-idf se obtiene un vector con las frecuencias de las palabras en cada documentación y código fuente, y con similitud de cosenos obtenemos la similitud entre los vectores obtenidos en el paso anterior con tf-idf.

El utilizar estas tres métricas diferentes ayuda enormemente a que el sistema de recomendación reconozca mejor las necesidades del paquete. Es frecuente que los desarrolladores olviden u omitan deliberativamente la documentación de sus proyectos, sin embargo, qué paquetes están siendo utilizados son datos que están ímplicitos en el código, así como el código mismo.

2. Motivacion

Este proyecto sirve como un prototipo de un sistema de recomendación que será implementado en la plataforma de programación en línea de MetaLambda.

MetaLambda es esencialmente un servicio de *hosting* y una plataforma de desarrollo de software de propósito general colaborativa.

Las librerías en MetaLambda son construidas por los mismos usuarios de la plataforma, por lo que la colección de paquetes que pueden ser incluidas en los proyectos de los usuarios es enorme. Varias herramientas están siendo desarrolladas para facilitarle la búsqueda de librerías y paquetes al usuario, como un motor de búsqueda, y este mismo sistema de recomendaciíon de librerías.

Common Lisp es el mismo lenguaje de programación que está siendo usado para desarrollar MetaLambda, así que se decidió utilizarlo también para nuestro sistema de recomendación.

3. Trabajos Previos

Hay numerosos trabajos relacionados con la detección de similitud en códigos fuente, sin embargo, hay dos trabajos que tienen bastante similitud con mi trabajo, CLAN [2] y MUDABlue [5] cuyo objetivo es encontrar software similar dentro de un repositorio. La finalidad de nuestro sistema es diferente a la de estos dos trabajos, pero comparten similitud en los procesos, ya que todos se encargan de encontrar software similar.

4. Modelo Propuesto

El modelo que proponemos en este trabajo se basa en en los métodos de tf-idf y similitud de cosenos para encontrar la similitud entre paquetes de software. Las características utilizadas que serían más intuitivas son la comparación entre las documentaciones de los paquetes y la comparación entre los códigos fuentes. La tercera métrica es la menos intuitiva y más innovadora, ya que se hace una adaptación de los métodos tf-idf y similitud de cosenos tomando como datos la frecuencia de utilización de funciones de las diferentes librerías incluidas en el paquete.

Otras características pueden ser usadas en este sistema, pero se decidió tomar estas tres para iniciar este prototipo, ya que consideramos que son las más relevantes.

4.1. Tf-idf

El método tf-idf, descrito en [1], tiene como finalidad determinar la relevancia de un término en un documento con respecto a una colección de documentos o *corpus*. Este método fácilmente puede ser adaptado para que calcule un vector con la relevancia de cada una de las palabras presentes en cierto documento, y de esta forma se pueden realizar comparaciones entre dos documentos.

Este método consiste en en el producto de dos estadísticas [6]: *tf* o *term frequency* e *idf* o *inverse document frequency*. Para la estadística *tf*, el método debe calcular la frecuencia de un término en un sólo documento. La forma más simple sería determinar las veces que un término aparece en un documento. Sin embargo, existen otras posibilidades [3], como una forma booleana, en la si el término está presente en el documento, sin importar el número de veces, $tf(t,d) = 1$, donde t es el término y d es el documento. Este es el camino que tomamos para determinar la frecuencia tf en nuestra método. La estadística *idf* es una medida que determina qué tan común es cierto término en todo el *corpus*. Para calcular esta medida utilizamos la ecuación

$$idf(t, D) = log \frac{|D|}{|\{d \in D : t \in d\}|} \tag{1}$$

en donde t es el término del cual deseamos obtener la relevancia dentro del *corpus*, D, $|D$ es la cardinalidad de D o el número total de documentos, y $|d \in D : t \in d|$ es el número de documentos dentro de D en los que ocurre el término t. $tf(t,d) \neq 0$ ya que ocurriría una división entre 0; una solución común a esto es ajustar la fórmula a $|d \in D : t \in d| + 1$, pero en nuestro método decidimos implementar la condición de que si $tf(t,d) = 0$, entonces $tf := 1$.

La similitud entre dos códigos fuentes y las documentaciones de dos paquetes es llevada a cabo utilizando una implementación del método tf-idf.

Para medir la relevancia entre el uso de diferentes librerías en dos paquetes diferentes, se usa este mismo método, pero adaptándolo para que en lugar de utilizar términos en formato de cadena de caracteres, utilice el idetificador o nombre de un paquete. La frecuencia del uso de estas librerás es obtenida determinando el número de funciones de cada librería presentes en el código fuente del paquete. As'ı, si una librería que está siendo incluida en el paquete, pero ninguna de sus funciones forma parte de la estructura del código fuente, sería equivalente a una situación en la que no está siendo usada dicha librería. Al igual que la implementación del método tf-idf para medir la similitud entre las documentaciones y el código fuente, esta implementación regresa un vector que califica el uso de cada librería presente en un paquete.

4.2. Similitud por coseno

Habiendo obtenido los vectores resultantes de la aplicación del método tf-idf, podemos utilizar el método de similitud por coseno para determinar qué tan similares son dos paquetes [7] de acuerdo a las tres características que estamos tomando en consideración.

La similitud por coseno puede ser obtenida a través de la ecuación

$$cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} \tag{2}$$

donde $cos(\theta)$ representa a la similitud entre los vectores A y B, $A \cdot B$ es el producto punto de ambos vectores y $\|A\|\|B\|$ es el producto de la magnitud de los vectores A y B. El producto punto de dos vectores a y b esta definido como

$$a \cdot b = \sum_{i=1}^{n} a_i b_i = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n = ab^T \tag{3}$$

y la magnitud de un vector x como

$$\|x\| = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}. \tag{4}$$

De esta forma, podemos reescribir (2) como

$$cos(\theta) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \sqrt{\sum_{i=1}^{n} (B_i)^2}} \tag{5}$$

y resulta fácil implementar el método utilizando ciclos que iteren por cada uno de nuestros vectores resultantes del *tf-idf* aplicado a los documentos siendo comparados.

Una vez calculada la similitud por coseno entre los vectores de dos documentos, el resultado es un escalar $r \in \Re$, donde $0 \leq r \leq 1$, y r nos representa una calificación de la similitud entre los dos documentos.

4.3. Promedio de calificaciones

En nuestro método aplicamos similitud por coseno a los vectores resultantes del *tf-idf*, pero este método tiene que ser aplicado tres veces, una vez a cada una de nuestras características. Por lo tanto, el resultado sigue siendo un vector que contiene las tres calificaciones por cada una de las características.

Para poder encontrar las librerías que tengan mayor similitud con nuestro paquete, es más conveniente trabajar con una calificación escalar que vectorial. En este trabajo se optó por utilizar el promedio de las tres calificaciones finales. Así que

$$similitud(P_D, P_S, P_L) = \frac{cos(\theta_D) + cos(\theta_S) + cos(\theta_L)}{3}, \quad (6)$$

donde P_D, P_S y P_L son la documentación, el código fuente y la frecuencia de uso de las librerías en los paquetes siendo comparados. Así, obtenemos como resultado final la similitud $s \in \Re$, donde $0 \leq s \leq 1$.

Podríamos utilizar otro método para reducir nuestro vector de calificaciones a un escalar, pero este método resulta ser suficiente para este prototipo. Notablemente, podríamos realizar esta misma operación de un promedio de las calificaciones, pero utilizando una ponderación. Los pesos utilizados en la ponderación pueden ser optimizados mediante algún algoritmo de optimización, tal como un algoritmo genético, y esto llevaría a un sistema de recomendación más efectivo.

4.4. Recomendación de paquetes

Al darle al sistema de recomendación nuestro paquete, esperamos que nos recomiende otros paquetes o librerías. Utilizando el promedio de calificaciones podemos obtener una lista de calificaciones de similitud de nuestro paquete con respecto a cada uno de los paquetes en el *corpus*. Sin embargo, la recomendación de los paquetes que son similares al nuestro puede no ser la mejor respuesta para el usuario, ya que el sistema simplemente nos está estableciendo cuáles paquetes son los más similares a nuestro paquete. La solución que se toma es dar como resultado las librerías que utilizaron los paquetes similares al nuestro, excluyendo aquellos paquetes que ya estamos utilizando en nuestro proyecto.

5. Trabajos futuros

El estado actual de este sistema de recomendación no es apto para ser aplicado a un proyecto en producción. Hay varios problemas que son notorios.

Un gran problema es que los vectores generados por el método *tf-idf* conservan las calificaciones de aquellos términos que no estuvieron presentes en el documento, pero que sí lo estuvieron en el *corpus*. Esto podría ser beneficioso cuando se aplica a la característica de la frecuencia de uso de librerías en el paquete, ya que el que dos paquetes no usen cierta librería puede ser considerada una semejanza. Sin embargo, no deberían ser de utilidad alguna cuando el método es aplicado a las documentaciones y a los códigos fuente de los paquetes.

Otro problema es que el sistema de recomendación presenta en los resultados cada uno de los paquetes dentro del *corpus* excluyendo al paquete siendo comparado. En una colección de paquetes pequeña no representaría un problema grande pero, conforme crece la colección de paquetes, la lista de resultados podría ser difícil de interpretar por el usuario.

En la lista de paquetes que se arroja como resultado del sistema, se le presentan al usuario cada uno de los paquetes dentro de la colección de paquetes junto con su porcentaje de similaridad. Como se ha mencionado dentro del planteamiento del sistema, estos paquetes similares no representan un resultado de utilidad para el usuario. Una mejor estimación de recomendaciones para nuestro paquete puede ser la suma de los productos de la semejanza de cada uno de los paquetes del *corpus* con respecto a nuestro paquete, con el $tf - idf(t, d, D)$ de cada una de las librerías siendo usadas por cada uno de los paquetes similares, donde t es cada uno de las librerías que están en uso en el paquete similar al nuestro, d es cada uno de los paquetes similares al nuestro, y D es nuestra colección de paquetes o *corpus*.

Una vez que el sistema de recomendación esté en una forma más robusta, debe ser implementado en la plataforma de MetaLambda, explicada en la sección 2. Como características más importantes, nuestro sistema de recomendación debe ser adaptado para trabajar con la base de datos de grafos *Titan* [4], y el sistema debería extraer automáticamente los paquetes de los cuales pertenecen las funciones que forman parte de la estructura de los paquetes - tarea que es fácilmente lograda gracias a Common Lisp.

Sin duda alguna, el trabajo más urgente es crear un método que nos permita medir la efectividad del sistema de recomendación. Un método propuesto por el Doctor Mario García Valdez, del Instituto Tecnológico de Tijuana, es tomar una muestra de paquetes del *corpus* y eliminar el uso de algunas librerías en los paquetes de la muestra. Si el sistema es capaz de recomendar las librerías que fueron eliminadas, entonces se encuentra en la dirección correcta. Podemos utilizar el porcentaje de recomendación descrito en esta misma sección como medida de rendimiento.

6. Conclusiones

La pieza central de nuestro sistema de recomendación es el método *tf-idf*. Gracias a este método resulta muy fácil encontrar similitudes entre documentos tomando en consideración una gran variedad de características, y puede ser

adaptado de una forma muy simple. Además de esta simplicidad, el *tf-idf* es muy versatil, ya que hay muchas posibilidades de encontrar las estadísticas *tf* y *idf*. Estamos seguros de que la modificación de este método es vital para el descubrimiento de maneras más efectivas de obtener mejores resultados.

Además de la modificación del método *tf-idf*, un área de investigación que se abre para nosotros es la elección de las características a considerar. Gracias a las grandes cantidades de información sobre el código fuente y los usuarios que es almacenada en la plataforma MetaLambda, se abre la posibilidad de crear un sistema de recomendación más robusto y refinado.

Una parte importante del sistema de recomendación a la cual no se le dio la importancia necesaria fue a la sección 4.3. El método elegido - simplemente promediar los valores de las calificaciones del *tf-idf* en el vector - es una solución demasiado burda. Podríamos utilizar algún método de optimización para encontrar pesos para una ponderación de las calificaciones que nos den resultados más significativos. Para poder lograr esto, es indispensable establecer un método que nos ayude a probar el sistema, un trabajo que se encuentra actualmente en desarrollo, tal como mencionamos en la sección 5.

Como conclusión general, hemos podido desarrollar un prototipo que cumple satisfactoriamente con las necesidades iniciales de un sistema de recomendación de librerías de código fuente para la plataforma de MetaLambda.

7. Agradecimientos

Quiero agradecer el apoyo de mi novia Elizabeth Quetzali Madera Hernández, quien me ayudó con muchas ideas para mejorar este sistema de recomendación, además de ser un miembro vital del equipo de MetaLambda.

También ha sido de gran importancia para mí tener el apoyo de CONACYT, el cual ha sido definitivo para que haya podido continuar con mis estudios de posgrado, así como el apoyo de todos los catedráticos del Instituto Tecnológico de Tijuana, especialmente al Doctor Mario García Valdez, quien contribuyó enormemente con su dirección en este proyecto.

Referencias

[1] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

[2] S. Kawaguchi, P. Garg, M. Matsushita, and K. Inoue. Mudablue: an automatic categorization system for open source repositories. In *Software Engineering Conference, 2004. 11th Asia-Pacific*, pages 184–193, 2004.

[3] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[4] M. B. Marko A. Rodriguez. Titan: Distributed graph database. http://thinkaurelius.github.io/titan/, June 2013.

[5] C. McMillan, M. Grechanik, and D. Poshyvanyk. Detecting similar software applications. In *Software Engineering (ICSE), 2012 34th International Conference on*, pages 364–374, 2012.

[6] M. S. . V. K. P.-N. Tan. *Introduction to Data Mining*. Addison-Wesley, 2005.

[7] A. Singhal. Modern information retrieval: a brief overview. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 24:2001, 2001.