Modelos de recuperación de información basados en n-gramas aplicados a la reutilización de código fuente

Enrique Flores[1], Martín Ibarra[2],
Lidia Moreno[1], Grigori Sidorov[2], and Paolo Rosso[1]

[1] Universitat Politècnica de València, España
{eflores,lmoreno,prosso}@dsic.upv.es
[2] Centro de Investigación en Computación, Instituto Politécnico Nacional, México
maibarraromero@yahoo.com.mx, sidorov@cic.ipn.mx

Resumen La detección de reutilización de código fuente es un problema creciente para la industria del software y el ámbito académico. Existe la necesidad de disponer de sistemas automáticos que sean capaces de detectar reutilización de código fuente eficientemente. Con este propósito hemos aplicado modelos de recuperación de información. Los resultados de este trabajo muestran que con la indexación semántica latente se obtiene una mejora significativa desde un punto de vista estadístico (t-Student).

Keywords: similitud entre códigos fuente, LSI, reutilización

1. Introducción

Los sistemas de Recuperación de Información (RI) nos permiten acceder a multitud de trabajos que se han realizado con anterioridad sobre un tema de interés. Este fácil acceso a la información ha cambiado la forma de trabajar propiciando la reutilización de una parte o la totalidad de un trabajo desarrollado por otros autores. No obstante, se puede incurrir en plagio si se reutiliza algún código y no se referencia adecuadamente la autoría del mismo.

Reutilizar código fuente puede obedecer a propósitos tales como reducir tiempo, errores y costes económicos en proyectos de desarrollo de software o, simplemente, plagiarlos. Según un informe del 2011 de la business software alliance[3], las pérdidas por el uso fraudulento del software ascienden a más de 45.000 M€. En el ámbito académico se revela un 30 % de casos de plagio según la encuesta realizada en [1]. Estos datos muestran la necesidad de disponer de herramientas que ayuden a descubrir los casos de plagio en tareas de programación.

Se han desarrollado diversas aproximaciones para la detección de reutilización de código fuente aplicadas a colecciones cerradas como [2,3,4], entre otros. Algunas de ellas han dado lugar a herramientas como JPLAG[4]. Sin embargo,

[3] http://globalstudy.bsa.org/2011/
[4] Actualmente disponible en http://jplag.ipd.kit.edu/.

existe la necesidad de aplicar modelos con buenos resultados en tiempo real. En este trabajo presentamos una comparación de dos modelos para la detección de reutilización de códigos fuente basadas en el modelo de espacio vectorial (*Vector Space Model, VSM*) usando n-gramas, con y sin la aplicación del indexado semántico latente (*Latent Semantic Indexing, LSI*). En la sección 2 describimos el corpus utilizado en la fase experimental. En la sección 3 describimos los modelos utilizados cuya evaluación se presenta en la sección 4. Finalmente, en la sección 5 mencionamos algunas conclusiones.

2. Descripción del corpus

Disponemos de un corpus de códigos fuente que contiene casos reales de plagio cometidos por estudiantes de segundo semestre en una asignatura de Comercio Electrónico Seguro facilitado por los autores de [4]. Para comparar las dos aproximaciones presentadas en la sección 3, se ha tomado una parte de este corpus. Esta consiste en un conjunto de 79 códigos escritos en el lenguaje C (130,934 caracteres). Nuestra aportación consiste en la anotación manual de los casos de reutilización de código. Han intervenido tres anotadores obteniendo un valor de acuerdo de $0,48$ en base a la medida kappa de Fleiss, lo que significa que existe un acuerdo moderado según [5]. Como resultado final se han obtenido 26 pares de códigos fuente considerados como reutilizados por dos o más anotadores y que serán los juicios de relevancia (*gold standard*) en nuestra experimentación.

3. Modelos basados en *n*-gramas

En RI dos documentos, d_q y d', se comparan usando un modelo R, el cual proporciona la representación de los documentos d_q y d' así como una función para calcular su similitud $\phi(d_q, d')$. Cuando R es un modelo algebraico, como VSM, véase por ejemplo [6], los documentos se representan como vectores o matrices de términos cuya similitud se representa como un valor escalar. Cuando d_q y d' son documentos que contienen código fuente, la existencia de reutilización se evaluará en términos de la similitud entre ambos códigos.

El comportamiento de los modelos se prueba bajo pre-procesos distintos según se estimen los términos como n-gramas de caracteres o de palabras. En el caso de n-gramas de caracteres, se convierten todos los caracteres a minúsculas y se eliminan espacios, tabulaciones y saltos de línea. En el caso de palabras, se eliminan los símbolos sintácticos como llaves, paréntesis, etc., palabras que aparecen una sola vez excepto palabras reservadas, palabras de un caracter (p. ej. variables tipo *i, j, k*), comentarios y valores numéricos, además de convertir los caracteres a minúsculas y eliminar el caracter '_' de los identificadores.

3.1. Modelo sin aplicar LSI

El modelo consiste en construcción de VSM utilizando n-gramas —en nuestro caso utilizamos unigramas, bigramas o trigramas de caracteres y palabras— para representar el código fuente dentro de un espacio vectorial; siendo cada n-grama una dimensión y su frecuencia la magnitud. De esa manera, los códigos fuente se representan como un vector dentro del espacio vectorial. Para su comparación hemos utilizado la conocida similitud del coseno cuyo resultado será un valor acotado entre $[0, 1]$ que será la similitud entre ambos códigos fuente.

3.2. Modelo aplicando LSI

Para interpretar mejor el modelo anterior, hemos realizado una comparación de su desempeño utilizando LSI, el cual encuentra relaciones "latentes" entre las dimensiones y reduciendo la dimensionalidad de la representación vectorial del documento, véase, por ejemplo en [7]. En nuestro experimento utilizamos valores para el modelo final de 10, 20, 30, 40, 50, hasta 400 dimensiones. Posteriormente se calcula el coseno de los vectores resultantes para determinar la similitud entre cada par de programas.

4. Experimentos

En esta sección se va a detallar la experimentación realizada con el modelo basado en n-gramas con y sin aplicación de LSI. La eficacia de ambos modelos se calculará utilizando la métrica *Mean Average Precision (MAP)*, la cual tiene en cuenta el ranking de similitudes obtenido. Se han realizado experimentos con y sin incrustación de código de funciones. La incrustación de código consiste en eliminar la llamada a funciones y sustituir por el cuerpo de éstas. Debido al reducido número de funciones en esta colección, los resultados de incrustar no son significativos, por lo que solo se presentan los resultados sin incrustación.

En la tabla 1 se muestra los resultados al aplicar LSI por cada dimensión (K) a la que se ha reducido el espacio vectorial. Los mejores resultados se han obtenido utilizando trigramas de caracteres y considerando 80 dimensiones. A partir de 80 dimensiones se obtiene el mismo valor de MAP, por lo que los códigos estarán mejor representados con las 80 dimensiones más significativas. Debido a la longitud de los códigos fuente, el número de dimensiones considerando n-gramas de palabras es muy reducido. Los resultados son ligeramente inferiores respecto al aplicar n-gramas de caracteres.

Tabla 1. *Mean Average Precision* para diferentes dimensiones (K) aplicando LSI sin incrustación para unigramas, bigramas y trigramas de caracteres y palabras.

	Caracteres			Palabras		
K	Unigramas	Bigramas	Trigramas	Unigramas	Bigramas	Trigramas
10	0,804	0,820	0,823	0,613	0,711	0,691
20	0,804	0,822	0,823	0,662	0,738	0,697
...	...	...	...	...	...	...
70	0,819	0,831	0,833	0,734	0,763	0,750
80	0,822	0,830	**0,836**	0,746	0,789	0,796
...	...	...	...	...	...	...
400	0,822	0,831	**0,836**	0,746	0,789	0,796

En la tabla 2 se muestran los resultados con n-gramas sin reducir las dimensiones con LSI. Los trigramas de caracteres han obtenido resultados ligeramente mejores al igual que cuando se ha aplicado LSI. Además, realizando un test estadístico t-Student comparando los resultados tanto con sólo n-gramas como aplicando LSI, las diferencias en los resultados han resultado ser significativas (p-value=0,045). Por lo tanto, la aplicación de n-gramas de caracteres ha obtenido buenos resultados pero mejorándose al aplicar LSI.

Tabla 2. *Mean Average Precision* para el modelo basado en espacios vectoriales para unigramas, bigramas y trigramas de caracteres y palabras.

	Caracteres			Palabras		
	Unigramas	Bigramas	Trigramas	Unigramas	Bigramas	Trigramas
MAP	0,712	0,753	**0,774**	0,772	0,766	0,756

5. Conclusiones y trabajo futuro

La aplicación de modelos de RI a la detección de reutilización ha demostrado ser eficaz en un entorno real del ámbito académico. Ambos modelos han mostrado un mejor desempeño utilizando trigramas de caracteres. Aplicando LSI se ha conseguido una mejora de los resultados significativa desde un punto de vista estadístico. LSI ha sido capaz de obtener información latente al reducir las dimensiones del espacio vectorial utilizando las dimensiones más representativas. Debido a los buenos resultados obtenidos con LSI, se planea extender como trabajo futuro su aplicación a la detección de reutilización translingüe, es decir, entre distintos lenguajes de programación.

Agradecimientos

Este trabajo es el resultado de la colaboración en el marco del proyecto WIQEI IRSES (Grant No. 269180 del FP 7 Marie Curie People). El trabajo ha sido financiado a través de los proyectos DIANA-APPLICATIONS-Finding Hidden Knowledge in Texts: Applications (TIN2012-38603-C02-01), SIP-IPN 20131441 y 20144274.

Referencias

1. Chuda, D., Navrat, P., Kovacova, B., Humay, P.: The Issue of (Software) Plagiarism: A Student View. IEEE Transactions on Education, 55(1), pp. 22–28 (2012)
2. Ohmann, T., Imad R.: Efficient clustering-based source code plagiarism detection using PIY. Knowledge and Information Systems, pp. 1–28 (2014)
3. Prechelt, L., Malpohl, G., Philippsen M.: Finding plagiarisms among a set of programs with JPlag. J. of Universal Computer Science, 8(11), pp. 1016–1038 (2002)
4. Arwin, C., Tahaghoghi, S.: Plagiarism detection across programming languages. In Proc. 29th Australasian Computer Science Conference, 48, pp. 277–286 (2006)
5. Fleiss, J.: Measuring nominal scale agreement among many raters. Psychol. Bull., 76, pp. 378–382 (1971)
6. Sidorov, G.: Construcción no lineal de n-gramas en la lingüística computacional: n-gramas sintácticos, filtrados y generalizados. México, 166 p. (2013) URL: `http://www.cic.ipn.mx/~sidorov/`
7. Deerwester, S., Dumais, S., Landauer, T., Furnas, G., Harshman, R.: Indexing by Latent Semantic Analysis. J. American Society for Information Science, 41(6), pp. 391–407 (1990)