ISBN: 978-987-4998-33-0
DOI:https://doi.org/10.33414/AJEA.4.413.2019

Detección automática de similitudes de código fuente utilizando técnicas de aprendizaje automático

Automatic detection of source code similarities using machine learning techniques

Presentación: 11/10/2019

Doctorando:
Marina Elizabeth Cardenas
Universidad Tecnológica Nacional – Facultad Regional Córdoba

Director/es:
Julio Javier Castillo

Resumen

En la presente propuesta de tesis se plantea el desarrollo de un modelo para detección de similitudes de código fuente para poder determinar la existencia de prácticas de reutilización aplicando técnicas vinculadas a la lingüística computacional, tales como minería de datos sobre texto y procesamiento del lenguaje natural. La identificación de similitudes de código puede servir para varios propósitos, entre los que se puede mencionar el estudio de la evolución del código fuente de un proyecto, detección de prácticas de reutilización, extracción de un fragmento de código para "refactorización" del mismo, seguimiento de defectos para su corrección, entre otros.

Palabras claves: código fuente, similitudes, reutilización, aprendizaje automático, texto, análisis.

Abstract

This thesis proposal proposes the development of a model for detection of source code similarities in order to determine the existence of reuse practices applying techniques related to computational linguistics, such as text data mining and natural language processing. The identification of code similarities have several aims, including the study of the evolution of the source code of a project, detection of reuse practices, extraction of a code fragment for "refactoring" of the project, monitoring of defects for correction, among others.

Introducción

Actualmente, la reutilización de código fuente es una práctica comúnmente utilizada en el proceso de desarrollo de software que conlleva una serie de inconvenientes que van desde la introducción de errores o defectos en el código fuente, lo que incurre en el incremento de tiempo de desarrollo y costos de mantenimiento (Fowler et al., 1999), hasta problemas éticos y legales por infringir derechos de propiedad intelectual. Baker

(1995) afirma en sus estudios que entre el 20 y 30% del código fuente de grandes sistemas de software corresponde a código fuente "clonado" o reutilizado.

La identificación de similitudes de código puede servir para varios propósitos (Smith y Horwitz, 2009), entre los que se puede mencionar el estudio de la evolución del código fuente de un proyecto, detección de prácticas de plagio, detección de prácticas de reutilización, extracción de un fragmento de código para "refactorización" del mismo y seguimiento de defectos para su corrección.

Si bien la problemática de detección de este tipo de prácticas es bastante compleja, existen diversas técnicas enfocadas a su detección automática, éstas son fuertemente dependientes del material de entrenamiento, del lenguaje de programación, y del tipo de código (algorítmico, definición de interfaces, código de acceso a datos, etc) sobre el cual se apliquen las mismas.

Entre algunas de las principales motivaciones por las cuales se realiza la reutilización se pueden mencionar las siguientes (Rattan et al., 2013):

- Limitaciones en las capacidades técnicas de los programadores.
- Restricciones de tiempo en el desarrollo de software.
- Dificultad en la comprensión de sistemas de gran complejidad.
- Limitaciones en los lenguajes de programación.
- Miedo a la introducción de errores por parte de los programadores por el desarrollo de código fuente nuevo.
- Reestructuración del código fuente con limitaciones de tiempo.
- Utilización de soluciones similares en distintos proyectos de software.
- Utilización de plantillas estructurales y funcionales para dar formato al código fuente (algunos paradigmas de programación recomiendan el uso de estas).

Si bien la reutilización de código fuente puede traer ciertas ventajas con respecto a la rápida adaptación del sistema ante los cambiantes requerimientos de los usuarios, en realidad puede implicar ciertas desventajas que ocasionen efectos negativos en el proceso de desarrollo de software, entre los que se destacan:

- Altos costos de mantenimiento.
- Propagación de errores.
- Impacto negativo en el diseño del sistema ya que fomenta el uso de malas prácticas de diseño.
- Incremento innecesario del tamaño del sistema lo que implica una degradación en la performance del mismo.

Como punto de partida, la hipótesis de trabajo de esta tesis postula que:

Dadas las similitudes entre el lenguaje natural escrito (textos no estructurados) y los lenguajes de programación (textos estructurados), es posible aplicar técnicas de procesamiento de lenguaje natural (específicamente de detección de Implicación Textual) al procesamiento de código fuente para determinar la similitud entre diferentes programas desde el punto de vista léxico, sintáctico y semántico.

Asimismo, la utilización de técnicas de aprendizaje automático posibilitará crear un modelo de detección que permita evaluar la similitud de un archivo fuente contra un conjunto de archivos de código fuente.

Dentro de este contexto, se propone la construcción de un sistema, empleando de técnicas de aprendizaje automático supervisado, tales como Redes Neuronales Artificiales (RNA) y Máquinas de Vectores de Soporte (SVM- Support Vector Machines), comúnmente utilizadas para minería de datos (Jadon, 2016). Dichas técnicas serán realimentadas con métricas de código fuente orientadas al procesamiento de lenguaje natural (Flores Sàez, 2017), con el objetivo de poder determinar un valor indicativo del nivel de similitud de un código fuente con respecto a un corpus, de manera tal de brindarle al usuario una medida cuantitativa que lo ayude en la identificación de fragmentos de reutilización de código.

Metodología

Desde el punto de vista metodológico, para el desarrollo de la tesis propuesta se pueden observar cinco etapas claramente diferenciadas: Creación del Corpus, Pre-procesamiento, Entrenamiento, Producción, y Prueba.

- Etapa de Creación de Corpus (material de entrenamiento):

Esta etapa consiste en la creación de material de entrenamiento sobre el cual se aplicarán las técnicas de Aprendizaje Automático Supervisado. Las actividades que se realizan como parte de esta etapa son: generación, tabulación, ordenamiento y etiquetado de los pares de entrenamiento.

- Etapa de Pre-procesamiento:

Esta etapa consiste en la aplicación de múltiples técnicas de lingüística computacional tales como detección de términos relevantes, aplicación de lexers, parsers, etc, sobre el corpus de entrada para construir/generar adecuadamente las features necesarias para el entrenamiento.

- Etapa de Entrenamiento:

Esta etapa consiste en partir de la información del Corpus y tomarla como Trainning Set (conjunto de entrenamiento) para el sistema. Básicamente, el sistema tiene la capacidad de aprender en base a ejemplos. Se construye un modelo en base a los ejemplos de entrenamiento y se buscan relaciones o patrones existentes entre los mismos.

- Etapa de Producción:

Una vez definido el modelo, el mismo podrá tomar como entrada dos documentos y determinar si hay o no similitud entre los mismos.

En la figura 1 se plantea el proceso que utilizará el sistema de detección de similitudes en códigos fuentes expresados en los lenguajes de programación JAVA y Python, en base al corpus elaborado especialmente para el desarrollo del presente trabajo.

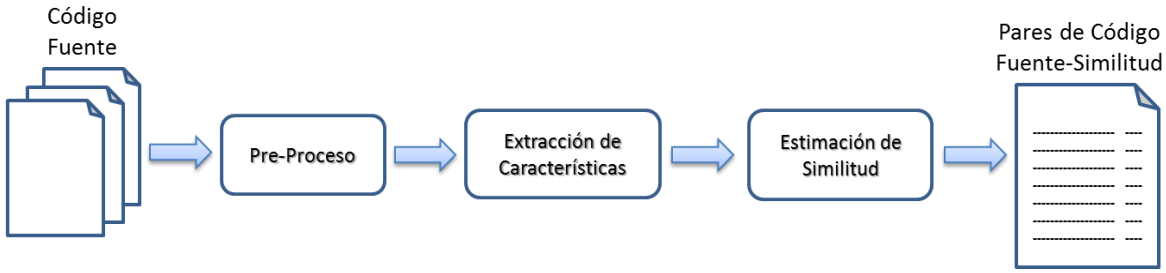

Fig. 1. Proceso de detección de similitudes en códigos fuente.

Se puede observar que, a partir del conjunto de documentos de entrada (archivos de código), se realiza el pre-proceso que consiste en dar formato al documento, su tabulación y eliminación de información redundante y/o superflua. Con el documento pre-procesado se realiza la extracción de las features relevantes, tales como las métricas basadas en técnicas de Reconocimiento de Implicación Textual de lenguaje natural propuestas por Castillo (2015), adaptadas para su utilización con texto estructurado.

Una vez realizada la estimación de similitud, se generará un archivo final con los resultados obtenidos y se los analizará en base a métricas previamente definidas, y a una revisión manual de expertos humanos.

- Etapa de Prueba:

Una manera de probar la efectividad el sistema será en base a la medida de Acc (acurracy) que indica la cantidad de veces que el sistema acierta, en función de la cantidad de veces que el sistema realiza una clasificación. Se analizarán y estudiarán diversas métricas para la evaluación de los resultados propios del área de investigación.

Resultados

El trabajo de tesis se encuentra en desarrollo dentro del proyecto homologado por la Secretaría de Ciencia y Tecnología de la Universidad Tecnológica Nacional, "Modelado para el procesamiento de textos estructurados" con código de identificación UTN4518. Actualmente esta tesis, se encuentra en la etapa de recopilación, tabulación y estudio de la documentación nueva y revisión de la existente en el marco el proyecto de investigación dentro del cual se enmarca. Esta actividad es fundamental para explicar las aportaciones al conocimiento que realiza la tesis al estado del conocimiento actual y como parte de esta actividad también se realiza el estudio y adaptación del sistema de Reconocimiento de Implicación Textual (RTE) Sagan propuesto por Castillo (2015) para la detección de similitudes en código fuente.

Además, se está realizando la recopilación, análisis y tabulación de los datos de entrenamiento del sistema propuesto para poder integrarlos con la arquitectura que define el comportamiento global del sistema.

Conclusiones

El trabajo realizado hasta el momento se ha centrado en el relevamiento de los datos necesarios para la elaboración del material de entrenamiento del sistema, diseño preliminar de la arquitectura del sistema y estudio del estado del arte de la temática planteada. Se prevee en trabajos futuros, profundizar en el mejoramiento de los datos de entrenamiento y refinamiento de la arquitectura, como así también el desarrollo del sistema de detección de similitudes en código fuente y definición del modelo subyascente, para lo cual se realizarán los cálculos necesarios para determinar las features definidas como datos de entrada para el mismo mediante la aplicación de múltiples técnicas de lingüística computacional tales como detección de términos relevantes, aplicación de lexers, parsers, etc, sobre el corpus con el objetivo de construir/generar adecuadamente las features necesarias para el entrenamiento.

Adicionalmente se estudiarán los diferentes mecanismos y algoritmos de visualización de diferencias en archivos de formato estructurado, y diferentes algoritmos para tratar con grandes volúmenes de información.

Referencias

Baker, M. (1995). On Finding Duplication and Near-Duplication in Large Software Systems. In Proceedings of the Second Working Conference on Reverse Engineering (WCRE'95), pp. 86-95, Toronto, Ontario, Canada.

Castillo, J. (2015). Tesis doctoral: Reconocimiento de Implicación Textual y Aplicaciones. Lugar: FaMAF-UNC, Ciudad: Córdoba, País: Argentina. Idioma: Español. Publicación impresa, biblioteca de FaMAF, UNC.

Jadon, S. (2016). Code clones detection using machine learning technique: Support vector machine. 2016 International Conference on Computing, Communication and Automation (ICCCA). IEEE. Noida, India.

Flores Sàez, E. (2017). Detección de reutilización de código fuente monolingüe y translingüe. Procesamiento del Lenguaje Natural. 2017, 58: 163-166.

Fowler, M., Beck, K., Brant, T., Opdyke, W., y Roberts, D. (1999). Refactoring: Improving the Design of Existing Code, Addison-Wesley Longman.

Randy Smith and Susan Horwitz. (2009). Detecting and Measuring Similarity in Code Clones. International Workshop on Software Clones (IWSC'09), pp. 28-34.

Rattan, D., Bhatia, R., Singh, M. (2013). Elsevier. Information and Software Technology 55, pp. 1165–1199.

Resnik P. (1995). Using information content to evaluate semantic similarity. In Proceedings of the 14th International Joint Conference on Artificial Intelligence, pages 448–453, Montreal.