Competición Vídeo: Inteligencia artificial aplicada al desarrollo de software

Aurora Ramírez Quesada
Dpto. Informática y Análisis Numérico, Universidad de Córdoba
aramirez@uco.es

***Resumen*—Los profesionales de la medicina, las finanzas o la logística han sabido incorporar los avances que ofrece la inteligencia artificial, convirtiéndola en una herramienta de apoyo para la toma de decisiones o la optimización de sus procesos. Normalmente, la creación de estos procesos inteligentes recae en ingenieros informáticos con conocimientos en estas técnicas. Sin embargo, los propios profesionales de la informática también se enfrentan a situaciones en las que la inteligencia artificial puede servir de apoyo para, por ejemplo, acelerar el tiempo de ejecución de los proyectos o mejorar los productos software. El propósito de este vídeo es presentar de manera divulgativa una de las técnicas de inteligencia artificial más conocidas, los algoritmos evolutivos, y cómo estos pueden aplicarse en el ámbito del desarrollo software.**

I. Introducción

La aplicación de técnicas de inteligencia artificial (IA) al proceso de desarrollo software no se limita a un único tipo de técnica. Si bien se utilizan ya métodos propios del aprendizaje automático o basados en redes bayesianas, la aplicación de técnicas de optimización y búsqueda han despertado un especial interés en los últimos años. La denominada ingeniería del software basada en búsqueda (*Search-based Software Engineering*, SBSE) [1] propone la reformulación de las tareas propias de la ingeniería del software como problemas de optimización, para a continuación abordar su resolución mediante técnicas como las metaheurísticas [2].

Actualmente, es posible encontrar propuestas SBSE para abordar tareas en la mayoría de fases del ciclo de vida del software:

1. Requisitos, con estudios centrados en la selección y priorización de los requisitos a implementar en la siguiente versión del software [3].
2. Análisis y diseño, tanto de sistemas orientados a objetos como basados en servicios [4].
3. Codificación, donde se persigue corregir errores y mejorar aspectos no funcionales del código fuente [5].
4. Pruebas, donde el objetivo es principalmente la generación automática de casos de prueba [6].
5. Mantenimiento, que engloba procesos de modularización y refactorización del código [7].
6. Gestión del proyecto, donde destacan problemas de planificación temporal y de predicción de costes [8].

Desde el punto de vista de las técnicas empleadas, los algoritmos evolutivos son posiblemente los más populares en SBSE [9]. Se trata de técnicas metaheurísticas inspiradas en la teoría de la evolución de Darwin [10]. Un algoritmo evolutivo realiza un proceso de optimización iterativo, el cual parte de una población de individuos generados de forma aleatoria. Cada individuo representa una solución candidata al problema, cuya calidad es medida por la función de *fitness*. En cada iteración, conocida como generación, algunos individuos son seleccionados para producir nuevas soluciones al problema. En concreto, el proceso más común es aplicar un operador de cruce, que combina las características de pares de soluciones buscando converger hacia la solución óptima. A continuación, se ejecuta un operador de mutación que altera pequeñas partes de una solución a fin de introducir diversidad en el proceso.

A la hora de aplicar un algoritmo evolutivo en el contexto de SBSE, es necesaria la definición de los dos elementos específicos del problema: la representación de las soluciones y la función de fitness [1]. En el primer caso, se suelen emplear las estructuras en forma de vector habituales en los algoritmos evolutivos, aunque también se pueden utilizar codificaciones en forma de árbol, adecuados para representar partes de un código fuente, o estructuras compuestas para problemas que necesitan manejar más información. Respecto a la función de *fitness*, las propuestas SBSE suelen utilizar medidas software para medir la calidad de los artefactos software producidos. Por ejemplo, la generación de casos de prueba se basa en criterios de cobertura del código, mientras que medidas de complejidad, cohesión o acoplamiento son frecuentes en las áreas de diseño y mantenimiento.

Se ha elegido esta temática porque permite mostrar a los profesionales de la informática que pueden beneficiarse de las soluciones que aporta la IA, no solo desarrollarlas. En cuanto a la técnica elegida, los algoritmos evolutivos son una de las técnicas pioneras en el ámbito de las metaheurísticas basadas en población, que a menudo resultan muy llamativas y son relativamente fáciles de entender por su símil biológico. Además, como se mencionó anteriormente, son muy populares en SBSE, por lo que es posible encontrar más ejemplos de aplicación.

II. Descripción del vídeo

El vídeo tiene como principal propósito presentar una visión global de las posibilidades que ofrece la IA a los profesionales software como método de apoyo a la decisión. Es por ello por lo que el vídeo comienza planteando diversas situaciones en las que los ingenieros deberían valorar distintas alternativas antes de elegir la mejor solución. De esta forma se busca hacer ver al espectador la dificultad que conlleva el desarrollo de

software y la variedad de casuísticas para las que la IA puede ser útil. Para mantener una separación clara entre el dominio de aplicación y la técnica de IA empleada, se utilizan colores de fondo diferentes como recurso visual.

Una vez realizada la introducción a la temática, el vídeo entra en una fase más teórica que tiene dos objetivos principales: (1) centrar al espectador en las técnicas de búsqueda, como una de las ramas de la IA; y (2) explicar los fundamentos básicos de los algoritmos evolutivos. En este punto, la mayor parte del vídeo se dedica a explicar el proceso iterativo de búsqueda que efectúa un algoritmo evolutivo. Para ello se introduce un pequeño robot que sirve para representar a la IA como una entidad "tangible" encargada de realizar los distintos pasos del ciclo evolutivo: generar soluciones, seleccionar las soluciones "padre" mediante torneo, generar soluciones "hijo" aplicando cruce y mutación, y reemplazar parte de la población para la siguiente generación. Dado el carácter téorico de esta parte del vídeo, se ha optado por utilizar diseños basados en pizarras y esquemas, así como fuentes que simulan la escritura manual.

Tras la explicación del proceso evolutivo de forma genérica, el vídeo aborda su adaptación a problemas concretos del ámbito del desarrollo software. Para ello se realiza un recorrido por el ciclo de vida clásico del software, planteando un problema para alguna de sus fases. En concreto, el vídeo retoma las preguntas que se planteaban los personajes al comienzo del mismo, transformándolas en la definición de los siguientes problemas de búsqueda:

- Planificación de proyectos software, cuyo objetivo es asignar tareas al equipo de desarrollo para minimizar el tiempo y coste del proyecto [11]. Este problema permite mostrar un ejemplo de codificación real.
- Diseño orientado a objetos, donde se pueden aplicar algoritmos evolutivos para encontrar la mejor asignación de propiedades y métodos a clases en base a criterios de cohesión y acomplamiento [12]. En este caso, se opta por una representación con vectores enteros.
- Selección de casos de prueba, que consiste en elegir un subconjunto de casos de prueba que alcancen una cobertura del código determinada [13]. Este problema se puede abordar como un problema de codificación binaria.
- Refactorización de código, donde se persigue encontrar una secuencia de operaciones de refactorización que eliminen el mayor número de defectos de diseño posibles [14]. Este ejemplo permite introducir una codificación basada en listas de distinta longitud.

Además de describir brevemente el objetivo que se persigue con la optimización y cómo se puede representar el problema, el vídeo aborda la definición de la función de *fitness* y un ejemplo de posibles operadores de cruce y mutación. De esta forma se consigue mostrar la flexibilidad de los algoritmos evolutivos y la variedad de problemas a los que se puede aplicar.

Finalmente, también se hace una mención a enfoques evolutivos más avanzados que tienen gran aplicabilidad en SBSE, como son los algoritmos multiobjetivo [15], que pueden optimizar más de una función objetivo, y los modelos interactivos [16], que permiten incorporar la opinión del experto.

III. Herramientas y materiales

Para la elaboración del vídeo se han utilizado herramientas software gratuitas y, en su mayoría, de código abierto. En primer lugar, la edición del vídeo se ha realizado con Powtoon[1]. Se trata de una aplicación web que permite la creación de vídeos con dibujos animados. Aunque se trata de una aplicación de pago, la versión gratuita ofrece la funcionalidad necesaria para la edición de vídeos de corta duración. Además de los recursos gráficos y sonoros proporcionados por la propia herramienta, el usuario puede incorporar sus propias imágenes, vídeos y audios. Powtoon pone a disposición del usuario una gran variedad de plantillas y ejemplos, que facilitan la creación de escenas más complejas. A su vez, los elementos gráficos (objetos, personajes, rótulos, etc.) están disponibles en diferentes estilos, lo que permite adecuar el contenido a diferentes propósitos y audiencias.

Para complementar el catálogo de imágenes de Powtoon, se han utilizado imágenes con licencia *creative common* disponibles en la plataforma Pixabay[2]. En ella pueden descargarse multitud de imágenes en formato PNG o vectorial de forma gratuita. Cuando ha sido necesario, las imágenes han sido editadas mediante herramientas como Inkscape[3] o Gimp[4].

IV. Utilidad didáctica del vídeo

El vídeo presentado a la competición puede ser un buen recurso didáctico por varios motivos. En primer lugar, proporciona una introducción ágil y práctica a las técnicas de búsqueda, que suelen formar parte de asignaturas sobre IA. Además, presenta los fundamentos básicos de la que posiblemente es la técnica más estudiada, los algoritmos evolutivos. Gracias a la presentación de varios ejemplos, el vídeo puede servir de complemento a la hora de mostrar ámbitos de aplicación menos conocidos. Igualmente, el vídeo también deja entrever la necesidad de desarrollar modelos más avanzados con los que abordar problemas complejos del mundo real, lo cual puede despertar el interés por estudiarlas en más detalle.

Cabe destacar también el marcado carácter interdisciplinar del vídeo, que lo hace muy interesante para los estudiantes de las distintas ramas de la informática. En este sentido, los estudiantes verán que aprender los fundamentos de la IA les puede ser de utilidad para distintos perfiles, tanto investigadores como profesionales. Esto es especialmente relevante dada la evolución que está sufriendo la informática y la necesidad que tienen las empresas de contar con perfiles interdisciplinares. Finalmente, el vídeo también puede ser utilizado de forma divulgativa fuera de las aulas, por ejemplo, para favorecer la transferencia de conocimiento con empresas.

[1] https://powtoon.com
[2] https://pixabay.com
[3] http://inkscape.com
[4] https://gimp.com

V. Conclusiones

El vídeo presentado a la competición CAEPIA'18 muestra de forma amena cómo la inteligencia artificial puede resolver problemáticas que surgen a menudo en el proceso de desarrollo software. La intención de este vídeo ha sido presentar a la IA como un método de apoyo a la decisión, capaz de abordar tareas tan diversas como encontrar la mejor planificación de equipos de desarrollo, seleccionar los requisitos que más beneficio reportarán a la empresa, optimizar el diseño de complejos sistemas informáticos o automatizar su entorno de pruebas. Debido a la limitación temporal, el vídeo se centra únicamente en la aplicación de algoritmos evolutivos, sobre los que se explica su proceso iterativo de búsqueda y cómo sus diferentes elementos se deben adaptar al problema a resolver. Para la realización del vídeo se ha utilizado Powtoon, una potente aplicación web para la creación de vídeos mediante dibujos animados.

A modo de valoración personal, la elaboración de este vídeo ha supuesto una gran oportunidad para dar a conocer un dominio de aplicación de la IA que no es tan popular en entornos académicos y, menos aún, entre los profesionales de la informática. También ha supuesto un ejercicio de reflexión sobre la dificultad a la hora de transmitir conceptos avanzados a una audiencia sin conocimientos profundos de IA y en un tiempo muy limitado. Esto ha requerido buscar el lenguaje apropiado, adaptar los contenidos sin perder rigor, recopilar ejemplos prácticos y ser capaz de explicarlos de forma sencilla, etc.

Finalmente, participar en esta competición también ha permitido a su autora aprender a utilizar herramientas software con gran potencial, como es Powtoon. Las habilidades adquiridas pueden ser de gran utilidad en el futuro para la elaboración de materiales didácticos y divulgativos relacionados con esta temática.

Enlace al vídeo

El vídeo se encuentra disponible en el siguiente enlace: https://www.youtube.com/watch?v=vfpY8_AEB6A

Agradecimientos

La autora desea expresar su agradecimiento a José Raúl Romero por sus ideas y comentarios sobre las primeras versiones del vídeo.

Referencias

[1] M. Harman, S. A. Mansouri, and Y. Zhang, "Search Based Software Engineering: Trends, Techniques and Applications," *ACM Computing Surveys*, vol. 45, no. 1, pp. 11:1–61, 2012.

[2] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Information Sciences*, vol. 237, pp. 82–117, 2013.

[3] A. M. Pitangueira, R. S. P. Maciel, and M. Barros, "Software requirements selection and prioritization using SBSE approaches: A systematic review and mapping of the literature," *Journal of Systems and Software*, vol. 103, pp. 267–280, 2015.

[4] O. Räihä, "A survey on search-based software design," *Computer Science Review*, vol. 4, no. 4, pp. 203–249, 2010.

[5] J. Petke, S. O. Haraldsson, M. Harman, W. B. Langdon, D. R. White, and J. R. Woodward, "Genetic Improvement of Software: A Comprehensive Survey," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 3, pp. 415–432, 2018.

[6] P. McMinn, "Search-based software test data generation: A survey," *Software Testing Verification and Reliability*, vol. 14, no. 2, pp. 105–156, 2004.

[7] T. Mariani and S. R. Vergilio, "A systematic review on search-based refactoring," *Information and Software Technology*, vol. 83, pp. 14–34, 2017.

[8] F. Ferruci, M. Harman, and F. Sarro, "Search-Based Software Project Management," in *Software Project Management in a Changing World*, pp. 373–399, Springer-Verlag Berlin Heidelberg, 2014.

[9] M. Harman, "Software Engineering Meets Evolutionary Computation," *IEEE Software*, no. October, pp. 31–39, 2011.

[10] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*. Springer-Verlag Berlin Heidelberg, 2nd ed., 2015.

[11] E. Alba and J. F. Chicano, "Software project management with GAs," *Information Sciences*, vol. 177, no. 11, pp. 2380–2401, 2007.

[12] M. Bowman, L. C. Briand, and Y. Labiche, "Solving the class responsibility assignment problem in object-oriented analysis with multi-objective genetic algorithms," *IEEE Transactions on Software Engineering*, vol. 36, no. 6, pp. 817–837, 2010.

[13] L. S. de Souza, R. B. Prudêncio, F. d. A. Barros, and E. H. d. S. Aranha, "Search based constrained test case selection using execution effort," *Expert Systems with Applications*, vol. 40, no. 12, pp. 4887–4896, 2013.

[14] A. Ouni, M. Kessentini, H. Sahraoui, and M. Boukadoum, "Maintainability defects detection and correction: A multi-objective approach," *Automated Software Engineering*, vol. 20, no. 1, pp. 47–79, 2013.

[15] A. S. Sayyad and H. Ammar, "Pareto-optimal search-based software engineering (POSBSE): A literature survey," in *Proceedings of the 2nd International Workshop on Realizing Artificial Intelligence Synergies in Software Engineering (RAISE)*, pp. 21–27, 2013.

[16] A. Ramírez, J. R. Romero, and C. Simons, "A Systematic Review of Interaction in Search-Based Software Engineering," *IEEE Transactions on Software Engineering*, 2018. In press.