Tecnologías Semánticas para el desarrollo de Agentes Inteligentes: Generación de comentarios a partir de código fuente.

Cristian Vincenzini and Sandra Roger

email: cristian.vincenzini@est.fi.uncoma.edu.ar
roger@fi.uncoma.edu.ar

Grupo de Investigación en Lenguajes e Inteligencia Artificial
Departamento de Teoría de la Computación - Facultad de Informática
UNIVERSIDAD NACIONAL DEL COMAHUE

Resumen

La generación de comentarios de código es la tarea de generar una descripción de lenguaje natural de alto nivel para un método o función de código determinado.

Este trabajo, explora el uso de tecnologías de generación automática de texto explicativos a partir de código fuente. A partir de esta exploración, se pretende implementar un módulo que provea asistencia al desarrollador en la generación de un texto descriptivo, sobre una sección de código a su elección.

Para lograr este objetivo, se hará uso de técnicas de aprendizaje automático que permiten entrenar un modelo a partir de un corpus adecuado. Se empleará la técnica de transferencia de conocimiento en un modelo basado en atención, que permite aprovechar el aprendizaje previo de otros modelos con pequeños conjuntos de datos.

Para la evaluación de la calidad y utilidad de los comentarios generados se utilizarán diferentes lenguajes de programación correspondientes a distintos paradigmas.

Palabras Clave: Generación de Lenguaje Natural, Aprendizaje Automático, Código, Comentarios.

Contexto

Este trabajo está parcialmente financiado por la UNCo, en el marco del nuevo proyecto de investigación *Tecnologías Semánticas para el desarrollo de Agentes Inteligentes*. El proyecto de investigación tiene una duración de cuatro años y ha comenzado en 2022 y se desarrolla en forma colaborativa con docentes-investigadores de la UNS.

1. Introducción

El objetivo de la generación de lenguaje natural (GLN) es producir textos que sean indistinguibles del lenguaje natural hablado o escrito por una persona, de manera tal que el resultado final sea comprensible para cualquiera que reconozca el mismo lenguaje. La GLN a partir de código atañe a proveer de manera automatizada una descripción resumida de un fragmento de código [1]. Estas descripciones, también conocidas como comentarios, suelen ser utilizados para explicar el propósito y el funcionamiento de las diferentes partes del código fuente de un programa, y suelen ser escritos en un lenguaje natural legible para los humanos. Este proceso se realiza mediante el uso de algoritmos de procesamiento de lenguaje natural, algoritmos que permiten identificar la estructura gramatical y el significado de las palabras en un texto.

Desafortunadamente, gran cantidad del código fuente existente no se encuentra comentado, o bien existe pero no describe correctamente la funcionalidad que pretende explicar.

Es importante tener en cuenta que la generación de comentarios a partir de código fuente es una tarea compleja: problemas como la ambigüedad semántica, la falta de contexto o una sintaxis compleja del código analizado invariablemente impactan y reducen la calidad del resultado obtenido.

Existen diferentes enfoques para la generación de comentarios a partir de código fuente [2, 3, 4]. Un enfoque para esta tarea es aplicando técnicas de minería de datos para analizar el código y extraer información relevante del mismo. En la actualidad sin embargo es más común utilizar grandes modelos de lenguaje, que no son más que modelos estadísticos entrenados con millones de datos para realizar una tarea.

Una de las características de las arquitecturas de estos sistemas es que permiten refinar su aprendizaje para realizar una tarea específica, mecanismo que se conoce como *transferencia de aprendizaje*. Mediante esta técnica es posible utilizar estos grandes modelos ya entrenados y adaptarlos, obteniendo resultados aceptables en un tiempo considerablemente inferior al de entrenar un modelo desde cero.

Para este trabajo utilizaremos un *Transformer* conocido como T5 (2020) [5]. Este modelo permite procesar múltiples tareas relacionadas a la GLN. La tarea a desarrollar en esta línea de investigación es la generación de comentarios a partir de código fuente. Para ello tomaremos un modelo ya entrenado que será refinado sobre un conjunto de datos propio y posteriormente analizaremos los resultados obtenidos.

2. Línea de Investigación y Desarrollo

El proyecto de investigación *Tecnologías Semánticas para el desarrollo de Agentes Inteligentes* tiene como objetivo general, generar conocimiento especializado en el área de agentes inteligentes que accedan, procesen y recuperen información mediante tecnologías semánticas.

En este sentido, se desarrolla una línea de investigación enfocada en la generación de lenguaje natural. Específicamente, se busca generar comentarios descriptivos de funciones utilizando técnicas noveles en el campo del aprendizaje profundo.

El modelo de lenguaje utilizado en esta investigación inicial se basa en la arquitectura de transformadores de codificación-decodificación de CodeTrans [6]. En dicho trabajo se utilizaron tres tamaños del modelo T5 para realizar los entrenamientos: *small*, *base* y *large* (con 60, 220 y 770 millones de parámetros respectivamente). Estos modelos fueron entrenados para diferentes tareas en varios lenguajes de programación (Java, Python, Ruby, GO, entre otros) utilizando transferencia de aprendizaje.

La transferencia de aprendizaje consiste en dos etapas. Una etapa inicial de entrenamiento auto-supervisado donde se utilizan datos sin etiquetar y una segunda etapa, conocida como *fine-tuning* donde el modelo se entrena para una tarea específica utilizando datos etiquetados. Otra técnica utilizada en CodeTrans es el entrenamiento multi-tarea, la cual consiste en entrenar un modelo en múltiples tareas, utilizando datos etiquetados y sin etiquetar. Esta metodología permite, además, realizar un *fine-tuning* posterior a través de la transferencia de aprendizaje.

En una primera instancia en este trabajo tomamos la arquitectura T5 ya entrenada con los datos de CodeTrans, tomando los modelos realizados por transferencia de aprendizaje (TF) y multi-tarea (MT) para los tres tamaños considerados: *small, base* y *large*, y realizamos *fine-tuning* sobre cada uno de ellos, utilizando sets de datos propios para realizar diversos experimentos.

Estos conjuntos de datos propios están formado por dos corpus construidos: uno relacionado al lenguaje de programación GO y otro al lenguaje de programación PROLOG. Se están desarrollando pruebas y validaciones del trabajo de estos modelos sobre estos lenguajes a fin de formular hipótesis de investigación futura.

3. Resultados Obtenidos y Trabajos Futuros

Se estudiaron, analizaron y evaluaron diferentes enfoques y arquitecturas disponibles en investigaciones actuales. En base a este estudio se diseño e implementó una arquitectura a utilizar [7]. Este diseño está realizado sobre el trabajo de CodeTrans [6].

En base a esto y para comparar resultados se construyeron dos corpus de datos sobre dos lenguajes de programación. Se trabajó sobre un lenguaje que haya sido utilizado en dicho trabajo y que sirva como base, y sobre un lenguaje nuevo que no haya sido utilizado en dicha investigación. Ambos corpus contienen 120 líneas. La estructura de los datos consiste en una lista de tuplas, donde cada tupla está compuesta por una función y su correspondiente comentario en inglés que describe su funcionalidad.

Para nuestro primer conjunto de datos, elegimos el lenguaje de programación imperativo GO[1], el cual fue desarrollado por Google en el año 2009. Utilizando principalmente GitHub Copilot[2], se extrajeron las funciones y sus respectivos comentarios.

Para el siguiente conjunto de datos, seleccionamos PROLOG, un lenguaje lógico que utiliza predicados como elementos de ejecución. Los datos fueron obtenidos de diversas fuentes, pero durante la selección se evitaron predicados provenientes de módulos o paquetes de terceros, así como la utilización de sintaxis de gramáticas de cláusulas definidas (DCG).

En una primera instancia se llevaron a cabo experimentos para evaluar las tareas de *Transfer Learning* (TF) y de *Multi-Task Learning* con *Fine-Tuning* (MT-TF), debido a que en ambas tareas se emplea la utilización de un *Fine-Tune*. En cada tarea, se realizaron experimentos para los modelos de tres tamaños distintos: *small*, *base* y *large*.

Un primer set de experimentos fueron realizados sobre el lenguaje GO para comparar cómo se comportaba nuestro *finetuning*, el cual fue construido utilizando el corpus descripto anteriormente con respecto a los modelos de CodeTrans de [6] utilizando nuestro *dataset-test*.

Por otro lado, se hicieron evaluaciones de estas mismas tareas en los modelos pequeños y bases, como así también los modelos grandes en el lenguaje PROLOG. Como mencionamos anteriormente, CodeTrans no cuenta con este lenguaje. Al igual que con el lenguaje GO, se construyó un corpus tanto para realizar el *fine-tuning* como el *dataset* utilizado en el testeo. El objetivo fue probar cómo se comportaban los modelos para un lenguaje no contemplado previamente. Además, elegimos PROLOG por ser un lenguaje de programación declarativo, a diferencia de los otros lenguajes utilizados. Para realizar la comprobación se se utilizaron dos métricas: BLEU [8] y ROUGE [9]. Dentro de las variantes de la medida ROUGE nos concentramos en la ROUGE-L, con la ventaja de que puede capturar la estructura del nivel de oración de una manera natural.

Según los datos obtenidos y observaciones preliminares se pueden pensar que los comentarios de los lenguaje declarativo tienen cierta semi-estructuración donde se describen, muchas veces, las características de los parámetros que intervienen haciendo uso de los nombres de las variables y de los predicados. Además, en general, la longitud de los comentarios son relativamente más cortos. Esto hace pensar que las medidas de evaluación automáticas que generalmente se utilizan en estas tareas, no sean del todo adecuadas.

4. Formación de Recursos Humanos

Durante la realización de esta investigación se espera lograr, la culminación de 2 tesis de grado dirigidas y/o codirigidas por los integrantes del proyecto.

Así también, se espera que durante el desarrollo del proyecto, los integrantes del mismo estén en proceso de elaboración de su tesis de posgrado puedan consolidar su formación en investigación, y que el trabajo realizado contribuya

[1] https://go.dev/

[2] https://github.com/features/copilot

a su graduación.

Finalmente, es constante la búsqueda hacia la consolidación como investigadores de los miembros más recientes del grupo.

Referencias

[1] Alexander LeClair, Sakib Haque, Lingfei Wu, and Collin McMillan. Improved code summarization via a graph neural network. In *Proceedings of the 28th International Conference on Program Comprehension*, pages 184–195, 2020.

[2] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. Summarizing source code using a neural attention model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2073–2083, 2016.

[3] Yanlin Wang, Ensheng Shi, Lun Du, Xiaodi Yang, Yuxuan Hu, Shi Han, Hongyu Zhang, and Dongmei Zhang. Cocosum: Contextual code summarization with multi-relational graph neural network. *arXiv preprint arXiv:2107.01933*, 2021.

[4] Sonia Haiduc, Jairo Aponte, Laura Moreno, and Andrian Marcus. On the use of automated text summarization techniques for summarizing source code. In *2010 17th Working Conference on Reverse Engineering*, pages 35–44. IEEE, 2010.

[5] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

[6] Ahmed Elnaggar, Wei Ding, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Silvia Severini, Florian Matthes, and Burkhard Rost. Codetrans: Towards cracking the language of silicon's code through self-supervised deep learning and high performance computing. *arXiv preprint arXiv:2104.02443*, 2021.

[7] Cristian Vincenzini and Sandra Roger. Generación de comentarios a partir de código fuente utilizando transformers. *XXVIII Congreso Argentino de Ciencias de la Computación*, pages 973–977, 2022.

[8] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[9] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.