Generando modelos de características mediante Large Language Models manteniendo la coherencia sintáctica y semántica

José A.Galindo[1], Antonio J. Dominguez[1], Jules White[2], and David Benavides[1]

[1] Universidad de Sevilla, Sevilla, Spain
{jagalindo,afernandez7,benavides}@us.es
[2] Vandervilt University, NashVille, USA
jules.white@vanderbilt.edu

Resumen Los modelos de características representan aspectos comunes y variables de las lineas de producto software. El análisis automatizado de los modelos de características ha permitido probar, mantener y mejorar las líneas de productos de software. Para probar el análisis de los modelos de características suele ser necesario basarse en un gran número de modelos lo más realistas posible. Existen diferentes propuestas para generar modelos sintéticos de características; sin embargo, los métodos existentes no tienen en cuenta la semántica de los conceptos del dominio. Este artículo propone el uso de Large language models (LLM), como Codex o GPT-3, para generar variantes realistas de modelos que preserven la coherencia semántica al tiempo que mantienen la validez sintáctica.

Keywords: large language models · universal variability language · automated analysis

1. Introducción y motivación

Los modelos de características son usados para representar aspectos comunes y variables de sistemas software que necesitan lidiar con variabilidad para funcionar de manera correcta. Algunos ejemplos son el núcleo de Linux[5] o gestores de contenido CMSs[9]. Estos modelos pueden contener más de 7000 características, lo que hace que el análisis manual sea laborioso y propenso a errores. Para abordar este problema, se introdujo el análisis automatizado de modelos de características hace 30 años[3], esto llevó al desarrollo de diversas aplicaciones y técnicas que permitieron pruebas, mantenimiento y mejora de líneas de productos de software y sistemas de variabilidad intensiva.

Los Large Language Models (LLMs) han surgido como una herramienta para procesar y analizar datos de lenguaje natural, con aplicaciones en traducción de idiomas, resumen de texto y generación de lenguaje[1]. Recientemente, se ha desarrollado el Lenguaje de Variabilidad Universal (UVL) como un nuevo lenguaje específico de dominio para serializar modelos de características[8]. En este artículo, los autores proponen utilizar LLMs para generar modelos de características realistas que preserven la semántica de los conceptos utilizados en el

modelo. Este enfoque tiene como objetivo crear modelos de características que imiten el realismo en términos de estructura y elementos de dominio.

2. Antecedentes

UVL (Universal Variability Language) El Lenguaje Universal de Variabilidad (UVL) es un lenguaje para gestionar la variabilidad en líneas de productos de software [8]. Este lenguaje busca simplificar la representación, divulgación y gestión de modelos de características y proporciona elementos para modelar relaciones entre características.

Modelos de Lenguaje de Gran Escala (LLMs) Los LLMs son redes neuronales pre-entrenadas con grandes cantidades de texto que han demostrado capacidades para tareas distintas a las que fueron diseñadas [6]. El tamaño de estos modelos mejora la precisión y revela nuevas propiedades en la entrada o "promnt" [4]. Los LLMs se pueden utilizar para tareas de generación de código entre otras, como se ve en OpenAI Codex [1].

Análisis de similitud semántica con LLMs Los "embeddings" son representaciones matemáticas de oraciones de un lenguaje en un espacio vectorial de alta dimensión [7]. Se pueden examinar la similitud de los "embeddings" utilizando métricas de distancia, como la similitud del coseno lo que nos puede servir para comparar la cercanía de los elementos del dominio de dos oraciones.

3. Solución propuesta

El proceso de generación de nuevos modelos de características que mantengan el dominio de las nuevas características requiere: 1) preparar el conjunto de entrenamiento para el LLM, 2) obtener propuestas del LLM, 3) verificar la validez sintáctica de los modelos. Nosotros usamos el analizador UVL existente en Flama [8] y finalmente, 4) realizar similitud del coseno para comparar los nombres de las características con respecto a las del modelo original.

Paso 1: Ajuste de la entrada Este proceso requiere refinar la entrada del LLM para mejorar la calidad de la salida del modelo. Normalmente esto se puede conseguir proporcionando contexto adicional y aclarando el formato de respuesta deseado. Este proceso requiere experimentación, refinamiento iterativo y comprensión de las fortalezas y limitaciones del modelo de inteligencia artificial.

Paso 2: Inferencia El proceso de inferencia implica elegir estrategias de decodificación adecuadas y ajustar los hiper-parámetros para obtener mejores resultados en la generación de texto. La parametrización de la inferencia para los métodos de muestreo estocástico requiere ajustar hiperparámetros como la temperatura, top-k, top-p (muestreo del núcleo) y penalización de repetición.

Paso 3: Validación sintáctica Para la validación sintáctica, se utiliza la implementación en Python de ANTLR de Flama [2] para verificar si la instancia DSL se adhiere a la sintaxis del lenguaje UVL.

Paso 4: Validación semántica El paso de validación semántica utiliza la similitud del coseno para medir la similitud entre dos conjuntos de características. El valor de similitud del coseno varía de -1 (completamente disímiles) a 1 (idénticos). Los valores más cercanos a cero indican que las dos frases son menos similares en significado. La similitud del coseno se calcula entre los conjuntos de características de diferentes variantes de modelos, lo que ayuda a determinar su similitud en dirección.

4. Resultados preliminares

En este estudio, los autores descargaron inicialmente 1215 modelos de características del sitio web de SPLOT y los convirtieron al formato FaMa XML. Después de filtrar los modelos no reales y aquellos con nombres de características significativos, se quedaron con 30 modelos que tradujeron al DSL UVL. El proceso de experimentación se muestra en la Figura 1.

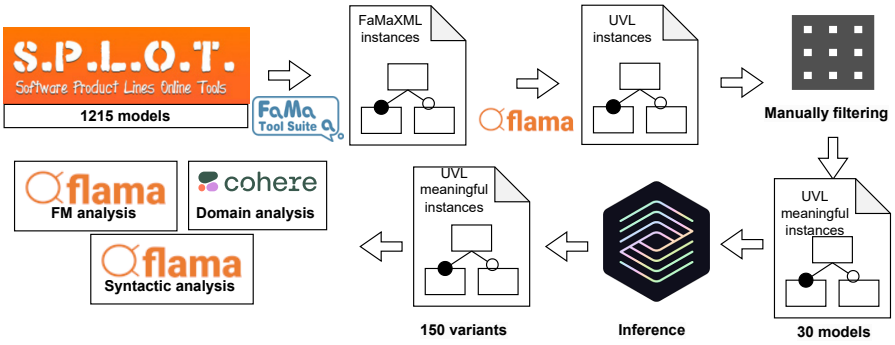

Figura 1. Proceso de experimentación

En nuestros experimentos usamos Codex [10] para generar cinco instancias o variantes por cada modelo usado como promnt, lo que resultó en un total de 150 modelos. Luego, analizamos estos modelos tanto sintáctica como semánticamente. El análisis sintáctico y las transformaciones de formato se realizó utilizando Flama [2], mientras que la plataforma Cohere se utilizó para el análisis semántico, comparando los dominios de los modelos originales con las variantes sintácticamente válidas generadas. En estos experimentos observamos que el 90 % de los modelos eran válidos sintácticamente, que había diferencias en los mismos y que las características introducidas eran del mismo dominio que las del modelo usado como entrada del LLM.

Agradecimientos

Este trabajo ha sido financiado por el proyecto *Data-pl*(PID2022-138486OB-I00), la red TASOVA PLUS (RED2022-134337-T) financiada por el Ministerio de ciencia e innovación(FEDER)y el proyecto *METAMORFOSIS* (FEDER_US-1381375) financiado por la Junta de Andalucía.

Referencias

1. Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H.P.d.O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al.: Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374 (2021)
2. Galindo, J.A., Benavides, D.: A python framework for the automated analysis of feature models: A first step to integrate community efforts. In: SPLC '20: 24th ACM International Systems and Software Product Line Conference, Montreal, Quebec, Canada, October 19-23, 2020, Volume B. pp. 52–55. ACM (2020). https://doi.org/10.1145/3382026.3425773, https://doi.org/10.1145/3382026.3425773
3. Galindo, J.A., Benavides, D., Trinidad, P., Gutiérrez-Fernández, A.M., Ruiz-Cortés, A.: Automated analysis of feature models: Quo vadis? Computing **101**(5), 387–433 (2019). https://doi.org/10.1007/s00607-018-0646-1, https://doi.org/10.1007/s00607-018-0646-1
4. Kaplan, J., McCandlish, S., Henighan, T., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., Amodei, D.: Scaling laws for neural language models (2020)
5. Lotufo, R., She, S., Berger, T., Czarnecki, K., Wasowski, A.: Evolution of the linux kernel variability model. Software Product Lines: Going Beyond pp. 136–150 (2010)
6. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer (2020)
7. Reimers, N., Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks (2019)
8. Sundermann, C., Feichtinger, K., Galindo, J.A., Benavides, D., Rabiser, R., Krieter, S., Thüm, T.: Tutorial on the universal variability language. In: SPLC '22: 26th ACM International Systems and Software Product Line Conference, Graz, Austria, September 12 - 16, 2022, Volume A. p. 260. ACM (2022). https://doi.org/10.1145/3546932.3547024, https://doi.org/10.1145/3546932.3547024
9. Sánchez, A., Segura, S., Ruiz-Cortés, A.: The drupal framework: A case study to evaluate variability testing techniques. In: International Workshop on Variability Modeling of Software-Intensive Systems (VAMOS) (2014), http://www.scopus.com/inward/record.url?eid=2-s2.0-84897649985&partnerID=40&md5=de6353198a27406792a498089482854a, cited By (since 1996)0
10. Xu, F.F., Alon, U., Neubig, G., Hellendoorn, V.J.: A systematic evaluation of large language models of code. In: Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming. pp. 1–10 (2022)