CADbot: Grandes modelos de lenguaje para la generación 3D en sintaxis CAD

S. Corley sano.corley@gmail.com[1,2] and E. Iarussi emmanuel.iarussi@utdt.edu[1,3]

[1] Universidad de Torcuato Di Tella, Argentina
[2] Universidad de Buenos Aires, Argentina
[3] Consejo Nacional de Investigaciones Científicas y Técnicas, Argentina

Abstract. La síntesis de objetos 3D a partir de lenguaje natural es un campo con grandes avances en la literatura reciente. La proliferación de los grandes modelos de lenguaje (LLMs) permitió su uso para esta tarea. En este trabajo presentamos un trabajo en desarrollo con el que proponemos generar objetos CAD en sintaxis OpenSCAD, un lenguaje que permite generar mallas 3D mediante descripciones textuales.

Keywords: LLM · OpenSCAD · CAD

1 Introducción

Recientemente han surgido numerosas aplicaciones de los grandes de modelos de lenguajes (LLMs por sus siglas en inglés, *Large Language Models*). Sus casos de uso además no se restringen a tareas sobre lenguaje natural, sino que con frecuencia se utilizan para diferentes tareas incluyendo la generación de código, interpretación de imágenes, y otros. [6,7]

Diferentes autores atacaron el problema de la interpretación o generación de objetos 3D mediante el uso de LLMs. [3,**?**] Estos trabajos típicamente utilizan representaciones entrenadas de los objetos 3D analizados, lo que puede ser costoso y difícil de resolver. Este trabajo considera la generación de objetos 3D como un problema de programación clásico, dónde entrenamos un LLM con la sintaxis y semántica propias de OpenSCAD[4], un lenguaje que permite generar objetos mediante descripciones textuales.

OpenSCAD es ampliamente usado para crear objetos 3D sólidos mediante descripciones textuales. Estas descripciones son compiladas y usadas para generar mallas poligonales mediante Geometría Constructiva de Sólidos (CSG). Para las descripciones, se tienen *primitivas* (como cubos, esferas, etc.) que se pueden combinar mediante operadores o *funciones* para generar una gran cantidad de formas. OpenSCAD tiene una sintaxis específica que los usuarios deben respetar al diseñar objetos.

Diferentes modelos, incluyendo GPT-4, cuentan con un dominio básico del lenguaje OpenSCAD como resultado de su extenso pre-entrenamiento. Sin embargo, a partir de estudios preliminares pudimos determinar que su calidad de

[4] https://openscad.org/

generación es muy pobre. Nuestro objetivo es crear un *chatbot* conversacional que pueda asistir al usuario en la generación de objetos 3D en lenguaje OpenSCAD.

2 Metodología y generación del dataset de entrenamiento

Para lograr nuestro objetivo, entrenamos un modelo de la familia LLaMA [6]. Hasta el momento no existen conjuntos de datos compuestos por código OpenSCAD emparejados con descripciones en lenguaje natural, y mucho menos en un formato similar al esperado por un *chabot*. Siendo un dataset con estas características esencial para nuestra tarea, este trabajo se centra en la generación de estos datos.

Partimos de una recopilación de objetos 3D obtenidos de internet, ya sea compartidos por usuarios o publicados en instructivos y blogs. Luego utilizamos GPT en tres instancias para limpiar y dar formato a la colección de modelos en OpenSCAD conseguidos. El objetivo de estas instancias fue convertir datos crudos en información utilizable para entrenar nuestro modelo de lenguaje, además de inyectar en el dataset información que permita al LLM aprender representaciones sobre el espacio tridimensional y la forma de los objetos.

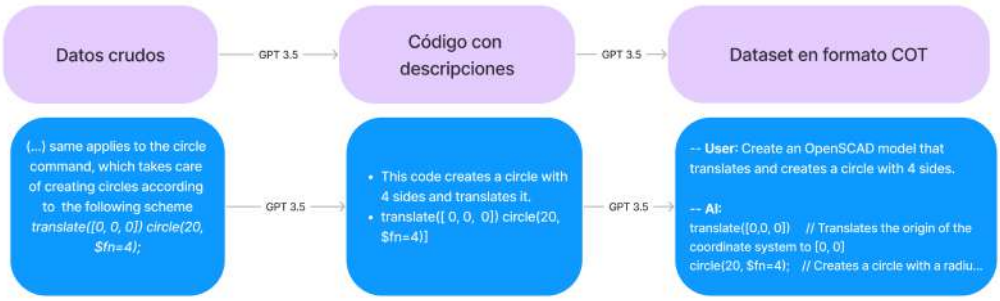

Fig. 1. Primeras dos instancias de la generación del dataset (en la parte superior de la figura), con un ejemplo ilustrativo (en la parte inferior).

Primera instancia. La primera tarea en la construcción del dataset a partir del texto obtenido, consistió en separar el código OpenSCAD de las descripciones. Por ejemplo, de un instructivo de cómo crear un modelo de una silla se extrajo el código de la silla y su descripción.

Segunda instancia. Esta instancia consistió en convertir los pares obtenidos en la instancia anterior en información que pueda ser utilizada para realizar un *instruction tuning* con *chain of thought*[5]. En otras palabras, cada par compuesto por código en OpenSCAD y su descripción en lenguaje natural fue utilizado

para obtener una instrucción que pudiera haber sido empleada para generar dicho código, junto con su versión en formato *chain of thought* (ver Figura 1). El dataset obtenido al finalizar esta instancia consiste de 415 tuplas.

Tercer instancia. Esta instancia tiene como objetivo generar pares de preguntas-respuestas que incorporen información sobre el espacio que ocupan los objetos 3D. Para esto modificamos el *pipeline* utilizado en LLAVA [7] para generar preguntas sobre los objetos usando solamente sus descripciones y GPT-4. Esto consistió en, dado el código de un objeto 3D y su descripción (obtenidos en las instancias 1 y 2 de pre-procesamiento), generar una serie de preguntas y respuestas sobre el objeto, su código y el espacio que ocupa. Nos centramos solamente en pares de preguntas-respuestas. Nuestra hipótesis es que estos datos pueden inyectar información sobre las relaciones espaciales entre los objetos.

3 Experimentos y Resultados

Metricas. El desarrollo de métricas que nos permitan cuantificar la *performance* de nuestro modelo es una parte del trabajo aún en desarrollo. Por ahora utilizamos dos métricas simples. La primera consiste en verificar el porcentaje de código generado que resulta compilable. La segunda consiste en observar la similitud entre el código OpenSCAD generado y referencias que obtuvimos al preprocesar el dataset. La métrica usada para la comparación de texto fue BLEU[1]. Para las siguientes etapas esperamos desarrollar una métrica que utilice la información visual de los objetos 3D.

Configuración experimental. Entrenamos una versión de LLaMA-2 conversacional usando Lora[2] durante 3 épocas. Utilizamos el algoritmo de optimización *AdamW* con un *learning rate* de $2e^{-5}$. El entrenamiento fue realizado sobre una tarea de *next word prediction*, usando *cross entropy* como función de pérdida, calculada sobre las respuestas generadas para cada consulta. La experimentación consistió en *promptear* al modelo para generar 60 objetos de OpenSCAD, obtenidos durante el pre-procesamiento del dataset y no utilizados para entrenar.

Table 1. CADbot V1 es el modelo *fine-tuneado* en el dataset de la instancia 2 del pre-procesamiento, y CADbot V2 es el resultado del fine-tuning en el dataset completo.

Modelo	Código compilable	BLEU score
Modelo base	1%	0.030
CADbot V1	72.5%	0.074
CADbot V2	76.6%	0.057

Resultados. Se entrenaron dos versiones del modelo: una combinando datos de las instancias dos y tres del pre-procesamiento, que incluye pares de instrucciones y preguntas sobre código de OpenSCAD; y otra utilizando solo datos de la

segunda instancia de pre-procesamiento, solo con pares de instrucciones. La tabla 1 muestra el impacto positivo del entrenamiento con nuestros datos, mejorando en la cantidad de código compilable. La diferencia es mayor al entrenar usando dataset generado en la instancia 2, lo que sugiere que el dataset completo es más rico. En la figura 2 se puede observar un ejemplo de la generación del chatbot.

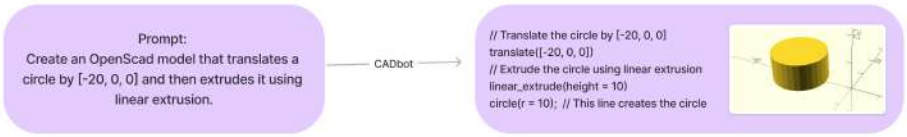

Fig. 2. Ejemplo de un modelo generado por CADbot durante la experimentación.

4 Conclusiones

Presentamos un modelo en desarrollo con el que proponemos generar objetos CAD en sintaxis OpenSCAD. El principal desafío consiste en la construcción de un conjunto de datos que pueda ser útil para entrenar un modelo de lenguaje con estas características. Nuestra forma de pre-procesar el dataset obtenido es apropiada y permite obtener buenos resultados, incluso con un dataset de tamaño limitado. A futuro, desarrollaremos métricas de evaluación que permitan integrar la información visual los objetos 3D generados.

References

1. Papineni, K., Roukos, S., Ward, T., & Zhu, W. J. (2002, July). Bleu: a method for automatic evaluation of machine translation. In Proceedings of the 40th annual meeting of the Association for Computational Linguistics (pp. 311-318).
2. Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., ... & Chen, W. (2021). Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685.
3. Xu, R., Wang, X., Wang, T., Chen, Y., Pang, J., and Lin, D. (2023). Pointllm: Empowering large language models to understand point clouds. arXiv preprint arXiv:2308.16911.
4. Siddiqui, Y., Alliegro, A., Artemov, A., Tommasi, T., Sirigatti, D., Rosov, V., ... & Nießner, M. (2023). Meshgpt: Generating triangle meshes with decoder-only transformers. arXiv preprint arXiv:2311.15475.
5. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., ... & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems, 35, 24824-24837.
6. Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., ... & Scialom, T. (2023). Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288.
7. Liu, H., Li, C., Wu, Q., & Lee, Y. J. (2024). Visual instruction tuning. Advances in neural information processing systems, 36