



Universidad Politécnica  
de Madrid

**Escuela Técnica Superior de  
Ingenieros Informáticos**



Máster Universitario en Inteligencia Artificial

Trabajo Fin de Máster

# **Distilling Ontologies from Large Language Models**

Autor(a): Edgar Jesús Huaranga Junco

Tutor(a): Dr. Raúl García Castro, Pablo Calleja Ibañez

Madrid, Julio 2023

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

*Trabajo Fin de Máster*  
*Máster Universitario en Inteligencia Artificial*

*Título: Distilling Ontologies from Large Language Models*

*Julio 2023*

*Autor(a):* Edgar Jesús Huaranga Junco

*Tutor(a):* Dr. Raúl García Castro, Pablo Calleja Ibañez  
Inteligencia artificial  
ETSI Informáticos  
Universidad Politécnica de Madrid

# Resumen

En el campo de informática, una ontología es una descripción formal explícita de conceptos en un dominio específico de conocimiento. El uso de esta forma de representación ha permitido compartir un entendimiento común de cómo se debería estructurar información entre distintas personas, instituciones y agentes trabajando en un campo específico. Sin embargo, el proceso de creación de estas ontologías puede presentar importantes dificultades, desde la selección de conceptos y sus atributos hasta la identificación de la relación que los une. Además, encontrar expertos en el dominio capaces de colaborar en el proceso de creación de la ontología resulta igualmente complicado y, en algunos casos, costoso. Afortunadamente, las tecnologías más recientes, como los grandes modelos de lenguaje ( en inglés, *Large Language Mode* o LLM <sup>1</sup> pueden ser una gran herramienta para enfrentar estos desafíos.

Los LLM son modelos de inteligencia artificial entrenados con grandes volúmenes de texto. Un entrenamiento de tal magnitud permite desarrollar técnicas con la capacidad de detectar patrones y relaciones entre los datos de entrada dentro de un contexto cada vez más amplio. Sin embargo, una de las críticas de mayor peso que recibe esta tecnología es precisamente su falta de conocimiento, una insuficiencia que genera las alucinaciones en las que incurre dando por cierta información falsa o incorrecta. La hipótesis de la que parte esta investigación sostiene que es posible extraer ontologías a partir de los LLM y que luego sus elementos puedan ser sometidos a una validación, bien a cargo de personas, o bien por medio de la comparación con otras fuentes de conocimiento existentes; este proceso podría resultar como una verificación del grado de conocimiento del mundo que realmente posee un LLM.

Este trabajo presenta dos propuestas. En la primera de ellas, se propone el uso de los LLM como asistentes en la creación de ontologías de dominio específico, lo que permitiría superar los desafíos mencionados anteriormente. Al trabajar junto con expertos en el dominio, los LLM pueden favorecer la celeridad en la construcción de ontologías para la extracción de conceptos, atributos y relaciones entre entidades proporcionando una base sólida de términos y relaciones. En segundo lugar, se propone una evaluación de algunos elementos de la ontología que fue extraída de un LLM al realizar una comparación de relaciones y propiedades generadas con una base de conocimiento abierta.

Finalmente, se insiste en que, bien los LLM no pueden reemplazar por completo el trabajo realizado por expertos en dominios específicos, sí pueden mejorar significativamente la eficiencia del proceso de creación de ontologías y reducir la carga de trabajo tanto de los expertos como de los desarrolladores.

---

<sup>1</sup>Parte de las siglas empleadas en el texto proceden del inglés, si bien su desarrollo y explicación se realiza en español



# Abstract

In the field of computer science, an ontology is an explicit formal description of concepts in a specific domain of knowledge. The use of this form of representation has made it possible to share a common understanding of how information should be structured among different people, institutions and agents working in a specific field. However, the process of creating these ontologies can present significant difficulties, from the selection of concepts and their attributes to the identification of the relationship that links them. Moreover, finding domain experts capable of collaborating in the ontology creation process is equally challenging and, in some cases, costly. Fortunately, more recent technologies, such as Large Language Models (LLM), can be a great tool to face these challenges.

LLMs are artificial intelligence models trained on large volumes of text. Training of this magnitude allows the development of techniques with the ability to detect patterns and relationships between input data within an ever-widening context. However, one of the major criticisms of this technology is precisely its lack of knowledge, an insufficiency that generates the hallucinations it incurs by taking false or incorrect information for granted. The hypothesis of this research is that it is possible to extract ontologies from LLMs and that their elements can then be subjected to validation, either by individuals or by comparison with other existing sources of knowledge; this process could result in a verification of the degree of knowledge of the world that an LLM actually possesses.

This paper presents two proposals. The first one proposes the use of LLMs as assistants in the creation of domain-specific ontologies, which would overcome the challenges mentioned above. By working together with domain experts, LLMs can help speed up the construction of ontologies for the extraction of concepts, attributes and relations between entities by providing a solid base of terms and relations. Secondly, an evaluation of some elements of the ontology that was extracted from an LLM is proposed by performing a comparison of generated relations and properties with an open knowledge base.

Finally, it is stressed that while LLMs cannot completely replace the work done by domain-specific experts, they can significantly improve the efficiency of the ontology creation process and reduce the workload of both experts and developers.



# Tabla de contenidos

<b>1. Introducción</b>	<b>1</b>
1.1. Objetivo principal	2
1.1.1. Objetivos específicos	2
1.2. Metodología y plan de trabajo	3
1.3. Estructura del documento	3
<b>2. Estado de la cuestión</b>	<b>5</b>
2.1. Metodologías para la construcción de ontologías	5
2.2. Automatización en la construcción de ontologías	6
2.2.1. <i>Ontology learning</i>	6
2.2.2. <i>Knowledge graph extraction</i>	6
2.3. Extracción de entidades	7
2.4. Extracción de relaciones	9
2.5. <i>Large Language Models</i>	10
2.6. Límites del estado de la cuestión	10
<b>3. Fundamento teórico</b>	<b>11</b>
3.1. ChatGPT	11
3.2. Modelos disponibles	11
3.3. Parámetros	12
3.4. <i>Prompt engineering</i>	13
<b>4. Diseño del estudio</b>	<b>15</b>
4.1. Consideraciones principales	15
4.2. La generación de ontologías	15
4.2.1. Obtención de clases	16
4.2.2. Obtención de atributos por clase	18
4.2.3. Obtención de relación entre dos clases	19
4.2.4. Construcción de la ontología	22
4.3. Evaluación de la cobertura	23
<b>5. Resultados y discusiones</b>	<b>27</b>
5.1. Generación de ontologías a partir de LLM	27
5.1.1. Caso 1: <i>Air Traffic Ontology</i>	27
5.1.2. Caso 2: <i>Tourism Ontology</i>	27
5.2. Cobertura de los atributos generados por LLM	30
5.2.1. Caso 1: <i>International airport</i>	30
5.2.2. Caso 2: <i>Video game</i>	32

5.2.3. Caso 3: <i>Ceremony awards</i> . . . . .	34
<b>6. Conclusiones y trabajo futuro</b>	<b>37</b>
6.1. Conclusiones . . . . .	37
6.2. Lineas de trabajo futuro . . . . .	38
<b>Bibliografía</b>	<b>38</b>



# Capítulo 1

## Introducción

En el campo de inteligencia artificial, se define una ontología como la especificación de un vocabulario representacional para un dominio compartido del discurso[1]. Esto significa que una ontología establece un vocabulario común para investigadores que precisan compartir información dentro de un dominio específico. La característica principal de este vocabulario es que los conceptos básicos del dominio y sus relaciones están en un formato que pueda ser entendido por computadoras.

Una de las motivaciones más comunes para el desarrollo de ontologías se encuentra en compartir el entendimiento que se posee sobre un campo de conocimiento específico y la forma en que se estructura la información dentro de este. La Web Semántica es un ejemplo de este planteamiento, puesto que busca otorgarle mayor significado a los datos y ofrecer flexibilidad para vincular distintas fuentes de datos[2]. En resumen, la Web Semántica no es distinta a la Web tradicional, sino una extensión de esta, donde la información posee significados bien definidos, permitiendo una mejor cooperación entre computadoras y personas [3].

El desarrollo de ontologías no consta de un proceso único. Se han propuesto múltiples metodologías para ello como Methontology[4] u On-To-Knowledge[5], así como herramientas que facilitan su desarrollo como WebProtégé[6]. NeON[7] presenta un glosario de actividades y procesos para el desarrollo de ontologías, donde se puede encontrar el **Knowledge Acquisition for Ontologies** para la extracción de conceptos y propiedades de distintas fuentes. Por otra parte, el **Ontology learning** transforma datos estructurados y no estructurados en estructuras conceptuales, además de otros procesos y actividades mostrados en la figura 1.1.

Desarrollar ontologías supone afrontar dificultades no solo técnicas sino también de tipo social-humano, que pueden generar inconvenientes técnicos y económicos. Como se describe en NeON, el proceso de *Knowledge Acquisition for Ontologies* incluye tareas relacionadas a consultas a expertos, lo que significa que existirá una interacción constante entre este grupo y las personas a cargo del desarrollo de la ontología.

En ese sentido, algunos de esas dificultades pueden ser abordadas desde la automatización mediante técnicas de inteligencia artificial, en concreto desde el campo del procesamiento de lenguaje natural (PLN). Los modelos de lenguaje, por ejemplo, inicialmente utilizaron técnicas estadísticas para determinar la probabilidad de que una determinada palabra continúe una oración. Estos modelos han evolucionado considerablemente y pasaron de ser modelos netamente probabilísticos a ser mo-

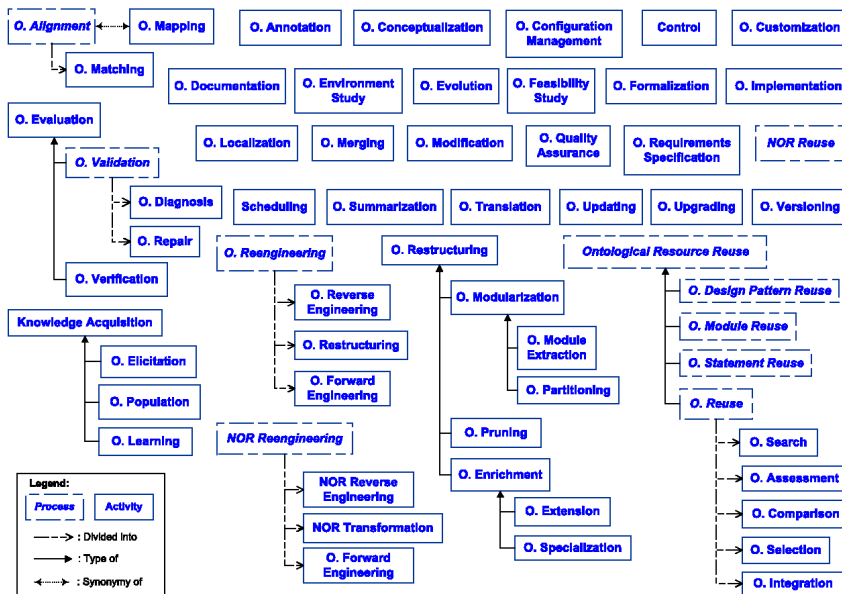


Figura 1.1: NeON. Glosario de procesos y actividades. Fuente: [7]

delos basados en redes neuronales recurrentes (en inglés, *Recurrent neural network* o RNN)[8]. Inicialmente, estos modelos presentaron problemas de coherencia en la generación de textos largos que luego fueron mejorando con enfoques como *Long Short-Term Memory* (LSTM) [9]. Sin embargo, con la presentación de un nuevo tipo de estructura llamada *Transformer* [10], se inicia una revolución en el desarrollo de estos modelos de lenguaje por su capacidad de procesar información en paralelo. Esto no solo permitió acelerar los tiempos de entrenamiento, sino también entrenar los modelos con corpus mucho más grandes.

Como consecuencia del entrenamiento con grandes volúmenes de datos y nuevas estructuras, los LLM han desarrollado la capacidad de realizar tareas que pueden ser de gran ayuda en el desarrollo de ontologías. Por ejemplo, es posible utilizar métodos de reconocimiento de entidades nombradas (en inglés *Named Entity Recognition* o NER) a partir de un LLM para extraer las clases correspondientes a una ontología. Asimismo, las técnicas de *Entity Relation* se han empleado para la extracción de relaciones entre entidades definidas en un contexto específico.

## 1.1. Objetivo principal

El objetivo principal de este trabajo es estudiar cómo un LLM puede ser utilizado como herramienta en el proceso de creación de ontologías para dominio específico. Este objetivo principal puede ser dividido en objetivos específicos para un mejor entendimiento.

### 1.1.1. Objetivos específicos

- Diseño de un procedimiento que permita extraer de los LLM elementos correspondientes a una ontología.
- Implementación de una aplicación web que permita asistir en la creación de

ontologías de dominio específico.

- Estudio de la cobertura de las ontologías generadas a partir del LLM.
- Realizar un mapeo del conocimiento que tiene un LLM sobre un tópico específico.

### **1.2. Metodología y plan de trabajo**

En primer lugar se realiza una revisión del estado del arte en la creación de ontologías, recogiendo los principales desafíos de este proceso que puedan ser abordados por métodos de PLN, específicamente LLM. Además, se revisan los fundamentos teóricos con el fin de comprender ciertos aspectos clave en la ejecución de las tareas y que resultan indispensables para cumplir los objetivos de este trabajo. A continuación, en el apartado de diseño, se plantean procedimientos a partir de la base teórica desarrollada anteriormente, alineados con los objetivos principales. Finalmente, se evalúa y discute cada procedimiento de acuerdo al planteamiento inicial del trabajo.

### **1.3. Estructura del documento**

Con el objetivo de brindar un mejor entendimiento de cómo se ha llevado a cabo este trabajo, en cada apartado se dedican unas líneas a la descripción de su propósito, así como de su importancia en el desarrollo y en la consecución del objetivo final.

En el primer capítulo se explica por qué se decidió realizar este trabajo. El estado de la cuestión pretende brindarle al lector un panorama sobre las decisiones en la metodología y los procesos a seguir. En el segundo capítulo se describen trabajos similares donde algún enfoque de inteligencia artificial ha sido utilizado como soporte para el desarrollo de ontologías. A continuación, se desarrolla el fundamento teórico necesario para el entendimiento del trabajo. Posteriormente, un capítulo se dedica al desarrollo completo del diseño de experimentos y procedimientos para la generación de ontologías a partir de LLM. En este mismo apartado, se incluye una alternativa de evaluación de los datos generados. Finalmente, el último capítulo incluye las conclusiones sobre el trabajo realizado y se describen los posibles campos de desarrollo en el futuro.



## Capítulo 2

# Estado de la cuestión

El desarrollo de ontologías constituye una constante interacción con expertos de dominio, además de la revisión de documentos y recursos con el objetivo de extraer entidades, relaciones y atributos. Normalmente, estas actividades son repetitivas y consumen mucho tiempo por lo que, naturalmente, se han planteado distintas metodologías y mecanismos para automatizar y acelerar estos pasos.

### 2.1. Metodologías para la construcción de ontologías

A pesar de que no exista un camino único para el desarrollo de ontologías, a lo largo del tiempo se han propuesto distintas metodologías que han cosechado un amplio seguimiento. Estas promueven la reutilización de términos e intentan priorizar la creación de ontologías que permitan su anexión a una red de ontologías. Por ejemplo, la metodología NeON[7], por un lado, sugiere escenarios y pautas necesarias para la ejecución de procesos y actividades, por otro, propone un glosario de procesos y actividades involucrados en la construcción de ontologías donde interactúan desarrolladores y expertos de dominio.

Otra metodología más reciente es *Linked Open Terms* (LOT)[11], enfocada en el desarrollo de ontologías y vocabularios para el sector industrial. Esta metodología incluye 4 actividades relacionadas con la especificación de requisitos, implementación, publicación y mantenimiento de la ontología. La figura 2.1 muestra el detalle de cada actividad de esta metodología. Aquí es posible observar la constante interacción entre el desarrollador de ontologías y el experto de dominio en prácticamente todas las etapas.

Además, se han documentado algunos obstáculos al momento de desarrollar ontologías. Existen obstáculos a nivel anotación, razonamiento, nombramiento, lógico y de contenido [12]. Un ejemplo de esto es la creación de múltiples clases haciendo referencia a la misma entidad o la definición de relaciones inversas cuando no necesariamente existen.

## 2.2. Automatización en la construcción de ontologías

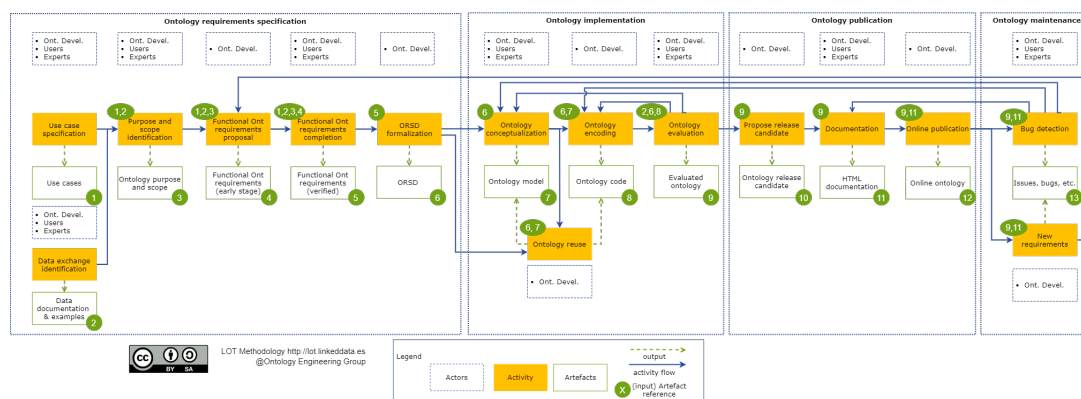


Figura 2.1: Complete LOT Methodology. Fuente: [11]

## 2.2. Automatización en la construcción de ontologías

### 2.2.1. Ontology learning

El proceso de construcción de ontologías se compone de tareas con potencial para ser automatizadas como la extracción de términos, clases y relaciones a partir de tablas, bases de datos, etc. En ese sentido, campos de investigación como el *ontology learning*, están orientados a plantear propuestas a este tipo de desafíos. Este campo ha ido evolucionando al mismo ritmo de los métodos en el procesamiento del lenguaje natural.

Los primeros sistemas de *ontology learning* fueron recogidos en 2003 por Shamsfard [13], donde se analizaron 7 distintos sistemas [14][15][16][17][18][19][20]. La características en común de estos sistemas incluye el aprendizaje de conceptos, relaciones e identificación de taxonomías a partir de textos. Este tipo de características y necesidades se ven reflejados en el *ontology-learning cake* que se muestra en la figura 2.2. Este gráfico permite representar las distintas capas del proceso de aprendizaje de ontologías a partir de datos no estructurados.

En el contexto de web semántica, [21] menciona la importancia de poder que existe entre la extracción de información (entidades, relaciones y conceptos) y recursos de la web semántica como las ontologías. Al mismo tiempo, realiza un listado de las técnicas utilizadas según la tarea objetivo. Por ejemplo, técnicas de reconocimiento de entidades nombradas y *Entity Disambiguation* se utilizan para la extracción de entidades. Para extraer conceptos se recurre a *Term extraction*, *Topic Modeling*, *Topic Labeling*, entre otros. Finalmente, las técnicas más utilizadas para la extracción de relaciones son técnicas del ámbito de *Relation Extraction*.

### 2.2.2. Knowledge graph extraction

Un campo de investigación con tareas similares a las del desarrollo de ontologías es el de la construcción de grafos de conocimiento o *knowledge graphs* (KG). Entre las similitudes que guardan ambos campos se cuentan la identificación de entidades y de sus relaciones, y la capacidad de exponerlas en grafos. Por ese motivo, también se incluyen trabajos relacionados con la construcción de grafos de conocimiento y se aborda el rol de LLM en esa tarea. Por ejemplo, el trabajo de [23] propone una estrategia para construir KG a partir de modelos de lenguaje preentrenados donde la extrac-

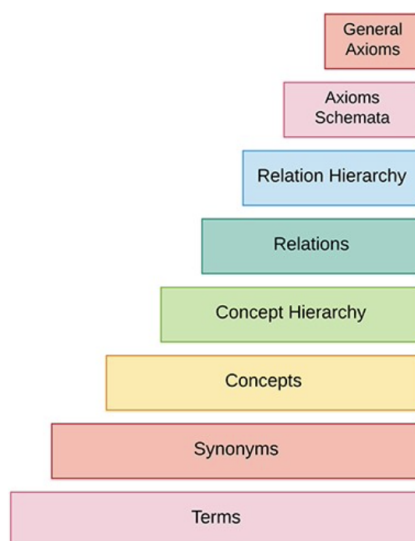


Figura 2.2: Ontology-learning layer cake. Fuente: [22]

ción de la relación entre dos entidades no es solo sintáctica sino también contextual. COMET[24] propone la construcción de KG a partir del conocimiento existente en formato de tuplas y un posterior aprendizaje en un modelo GPT[25]. Adicionalmente, el resultado del proceso de creación de grafos de conocimiento puede ser utilizado como una representación simbólica (BertNet [26]) del modelo de lenguaje revelando su capacidad cognitiva. Lo interesante de este trabajo es el proceso de creación de *prompts*, como se muestra en la imagen 2.4

Por otro lado, [27] evalúa la capacidad de los más recientes LLM, como ChatGPT, GPT-4 y otros modelos de la familia de OpenAI para generar KG. A diferencia de este, en CodeKGC [28] se utiliza un *code language model* para generar tripletas con un formato descrito inicialmente. Este último método aprovecha todos los datos estructurados (bases de datos, código fuente, etc) que fueron utilizados para el entrenamiento del LLM. Otro enfoque, denominado *Symbolic Knowledge Distillation (SKD)*[29] se basa en los LLM como GPT-3 con el objetivo de mejorar el *commonsense knowledge* de modelos más pequeños generando *commonsense KGs*, cuyos resultados son mejores en comparación con grafos de conocimiento generados por humanos.

### 2.3. Extracción de entidades

Este proceso busca identificar y extraer entidades a partir de datos no estructurados como textos, páginas web o publicaciones en redes sociales. Las entidades reciben el tratamiento de clases en las ontologías generadas.

El reconocimiento de entidades nombradas constituye un subcampo del PLN que se utiliza para extraer y clasificar entidades en textos. Esta capacidad ha permitido construir *lightweight ontologies*[30] aplicando técnicas de NER [31] haciendo uso de modelos de lenguaje derivados de *Bidirectional Encoder Representations from Transformers* (BERT)[32]. Además, es posible clasificar NER en tres subcategorías: *flat*

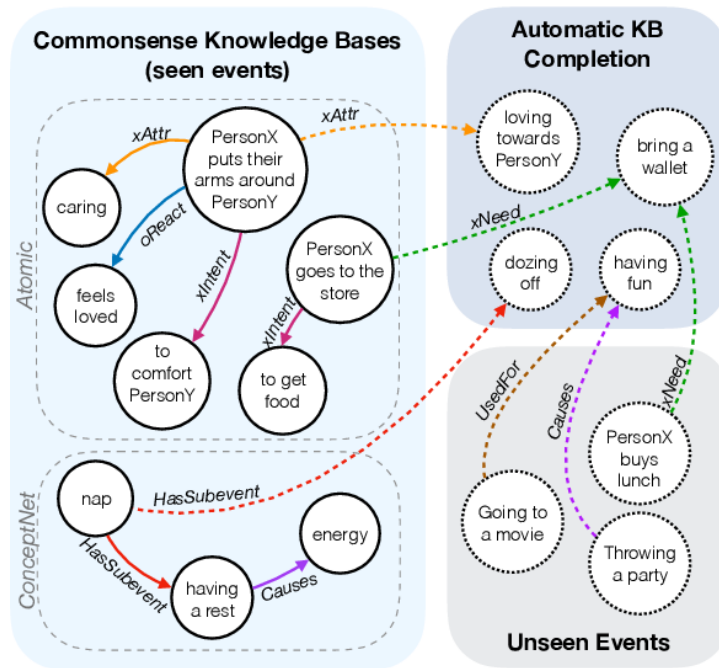


Figura 2.3: COMET toma conocimiento existente, ConceptNet por ejemplo, para luego generar un completado automático de un KG. Fuente: [24]

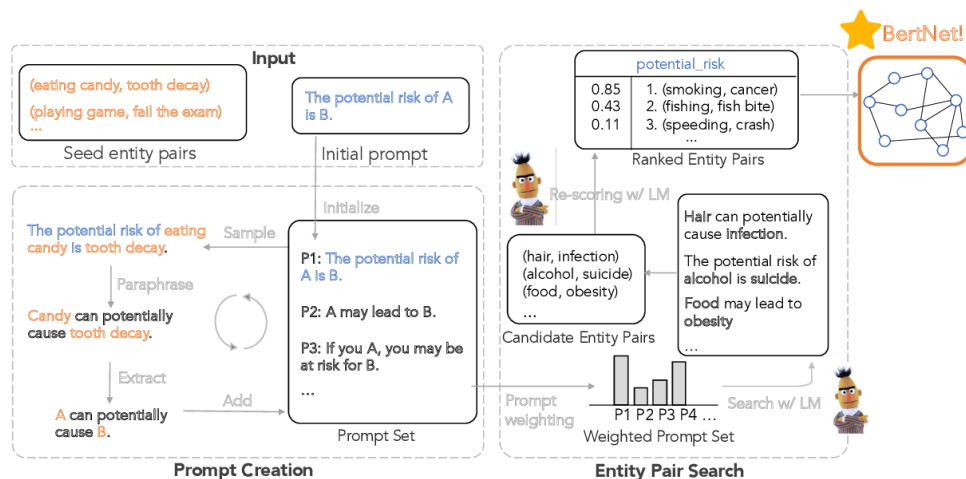


Figura 2.4: Resumen del proceso de extracción de grafos de conocimiento a partir de un input mínimo de parte del usuario. Fuente [26]



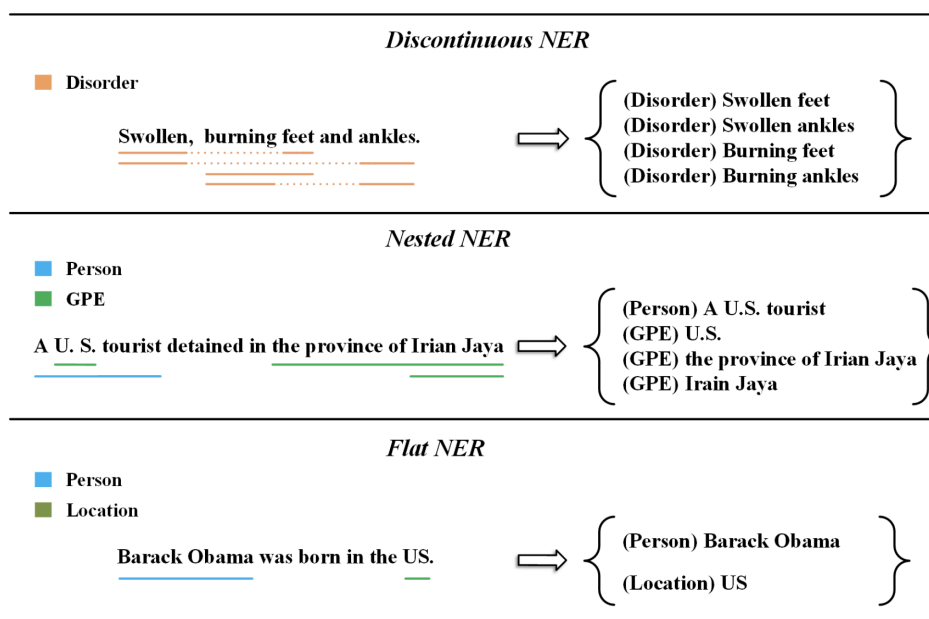


Figura 2.5: Ejemplos de *flat*, *nested*, *discontinuous* NER. Fuente: [37]

*NER*, *nested NER* y *discontinuous NER* cuyas características se muestran en la imagen 2.5. En los últimos casos, sobre entidades superpuestas y discontinuas, BERT también se emplea como modelo base[33] para generar representaciones de palabras con un contexto más profundo. Por otro lado, propuestas como [34] abordan los 3 distintos tipos de NER haciendo uso de un modelo preentrenado *Seq2Seq*[35] y BART [36], cuya arquitectura es similar a la de modelos como BERT[32] y GPT[25].

## 2.4. Extracción de relaciones

La extracción de relaciones consiste del proceso de identificar la relación semántica entre dos entidades. Dentro del contexto del desarrollo de ontologías, esta función es importante en la definición de la relación entre dos clases.

En este campo existen muchos modelos de lenguaje preentrenados con datos de campos especializados y que son derivados de BERT. Normalmente estos modelos no solo son para extracción de relaciones, sino también para identificar entidades. Seguidamente se describen algunos ejemplos:

- **BioBERT**[38], **SciBERT** [39]: Modelos preentrenado con texto especializado en el campo de biomedicina y científico, respectivamente.
- **Matching the Blanks: Distributional Similarity for Relation Learning** [40] presenta un primer enfoque sobre cómo BERT puede ser utilizado para tareas de extracción de relaciones tomando como base el método para descubrir reglas de inferencia a partir de texto de DIRT[41].
- **LayoutLM**[42] extrae y estructura el texto de documentos escaneados. Las palabras son enriquecidas con etiquetado de sus entidades semánticas y la relación con otras entidades.

- **Joint entity recognition and relation extraction as a multi-head selection problem**[43] presenta un buen rendimiento en tareas de extraer entidades y sus relaciones en contextos de noticias, campo biomédico, bienes raíces, etc., sin depender de etiquetados manuales ni herramientas externas de PLN.
- **LUKE(Language Understanding with Knowledge-based Embedding)**[44] es un modelo basado en transformers donde no solo las palabras son tokenizadas sino también las entidades. Esto le permite modelar la relación entre dos entidades de forma directa.

## 2.5. Large Language Models

En secciones anteriores se ha hablado de los LLM, modelos de aprendizaje profundo que pueden realizar tareas como identificar, predecir y generar texto a partir del conocimiento obtenido en su fase de entrenamiento a partir de grandes volúmenes de texto. Los LLM son modelos que basan su estructura en Transformers[10].

Los LLM poseen la capacidad de realizar tareas como NER o RE a partir de la información que han aprendido en la fase de entrenamiento. Sin embargo, los LLM requieren mucho poder computacional para poder ejecutar inferencia sobre este. Es por ello que no es sencillo construir un LLM en una computadora local o un clúster con capacidades limitadas, sino donde al menos se tenga hardware disponible como GPU. A diferencia de algunos modelos que fueron liberados para su uso en el ámbito científico como LLaMa[45].

## 2.6. Límites del estado de la cuestión

Según se ha podido ver, existen tareas dentro del proceso de creación de ontologías que pueden ser automatizadas. Sin embargo, muchas de estas propuestas sientan sus bases sobre textos con un formato preestablecido o hacen uso de datos semi-estructurados mostrando limitaciones al momento de interactuar con el lenguaje natural[46]. En ese sentido, la idea de este trabajo es aprovechar las capacidades de los LLM para interpretar textos no estructurados y poder realizar tareas como la extracción de entidades y de relaciones.

## Capítulo 3

# Fundamento teórico

Este capítulo describe los fundamentos que sostienen las bases necesarias para implementar propuestas que permitan la creación de ontologías. El contenido consiste principalmente en las distintas versiones que OpenAI dispone para realizar consultas sobre alguno de sus LLM disponibles.

### 3.1. ChatGPT

ChatGPT es una herramienta web desarrollada por la empresa OpenAI que permite una interacción conversacional. Una característica relevante de ChatGPT es su capacidad de generar texto similar al que podría ser generado por una persona. Este producto está desarrollado teniendo como base al modelo de lenguaje GPT-3[47]. Las instrucciones que el modelo emplea como *input* se denominan *prompts*, de esta manera la interacción humano-modelo se realiza netamente a través de intercambio de cadenas de texto.

ChatGPT no fue el primer producto desarrollado por OpenAI. Previamente, la empresa ya había presentado otros modelos como GPT y GPT-2 en 2018 y 2019, respectivamente. La diferencia con respecto a ChatGPT y GPT-3 radica en que, como en la mayoría de modelos de inteligencia artificial, cuanto mayor sea la cantidad de datos y mayor sea el poder de cómputo utilizados para el entrenamiento, los resultados obtenidos serán más precisos. La figura 3.1 muestra la diferencia en el tamaño de los tókenes utilizados para el preentrenamiento del modelo. De acuerdo a la documentación de OpenAI, es posible utilizar como referencia que un token corresponde aproximadamente a 4 caracteres de un texto. Es importante mencionar que el tamaño del modelo no es el único factor que influye en la experiencia de uso de ChatGPT sino también una técnica introducida llamada *reinforcement learning from human feedback* (RLHF), que permitió el desarrollo de nuevos modelos de lenguaje llamados InstructGPTs en los cuales se involucran a personas o moderadores en el proceso de generación del texto.

### 3.2. Modelos disponibles

A escala comercial, ChatGPT se presenta como un producto sin la posibilidad de jugar con parámetros adicionales durante la interacción con el modelo, pero no es el

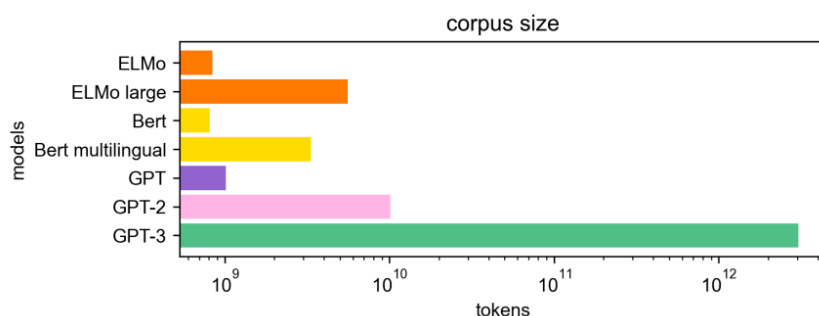


Figura 3.1: Tamaño del corpus utilizado para entrenar distintos modelos de lenguaje  
Fuente: hellofuture.orange.com

único disponible. A través del Playground<sup>1</sup> y del API de OpenAI es posible acceder a los distintos modelos desarrollados por la empresa. Estos modelos pueden ser conversacionales, como el modelo más reciente GPT-4, o de tipo de generación de texto, como el modelo text-davinci-003. Asimismo, se encuentran los modelos detrás de la generación de imágenes como DALL.E o Whisper, que permiten generar la transcripción de ficheros de audio.

OpenAI pone a disposición modelos con características distintas. Los modelos más relevantes son los siguientes:

- **Completion & chat models.** Estos modelos poseen la capacidad de entender y generar texto en lenguaje natural. Los modelos más potentes incorporan, además, la función de entender código.
- **Embedding models.** Estos modelos reciben palabras como entrada que convierten en representaciones numéricas (*embeddings*) y que posteriormente son aplicadas en tareas de búsqueda, *clustering*, recomendaciones, detección de anomalías o clasificación.
- **Audio models.** Estos modelos poseen la capacidad de generar transcripciones y traducciones a partir de un archivo de audio.
- **Fine tuning models.** Estos modelos se encuentran disponibles para realizar *fine-tuning* a partir de un conjunto de ejemplos detallados en un archivo.

En la tabla 3.2 se muestran los identificadores de cada modelo disponible en cada tipo descrito.

### 3.3. Parámetros

El API de OpenAI requiere de algunos parámetros para que el modelo pueda ejecutar la tarea asignada. En el caso de modelos de generación de texto los parámetros más relevantes para este trabajo son los siguientes:

- **temperature.** En el contexto de los modelos de lenguaje de OpenAI la temperatura hace referencia a la aleatoriedad del texto generado. Este valor varía entre

<sup>1</sup><https://platform.openai.com/playground>

## Fundamento teórico

---

ENDPOINT	MODEL NAME
/v1/chat/completions	gpt-4, gpt-4-0613, gpt-4-32k, gpt-4-32k-0613, gpt-3.5-turbo, gpt-3.5-turbo-0613, gpt-3.5-turbo-16k, gpt-3.5-turbo-16k-0613
/v1/completions	text-davinci-003, text-davinci-002, text-curie-001, text-babbage-001, text-ada-001
/v1/edits	text-davinci-edit-001, code-davinci-edit-001
/v1/audio/transcriptions	whisper-1
/v1/audio/translations	whisper-1
/v1/fine-tunes	davinci, curie, babbage, ada
/v1/embeddings	text-embedding-ada-002, text-search-ada-doc-001
/v1/moderations	text-moderation-stable, text-moderation-latest

Figura 3.2: Lista de tipos de modelos disponibles en la plataforma de OpenAI Fuente: [platform.openai.com](https://platform.openai.com)

cero (0) y dos (2). Mientras el valor sea más cercano a cero, menor es la aleatoriedad del texto generado.

- **max tokens.** Se refiere al máximo número de tokens utilizados, es decir, la suma de tokens utilizados entre la instrucción (*prompt*) y el resultado. Cada modelo disponible establece un máximo número de tokens posible.
- **stop.** Se trata de un conjunto de hasta 4 caracteres que permite detener la generación de más tokens.

Del mismo modo, la API no solo establece límites en la cantidad de consultas a su servicio por minuto (*request per minute* RPM) sino también del máximo número de tokens por minuto (*tokens per minute* TPM). Estos límites están detallados en la página web<sup>2</sup> de OpenAI. Durante la escritura de este trabajo, la mayoría de los modelos generadores de texto tienen establecido 3000 RPM y 250 000 TPM. Los modelos más grandes y recientes como GPT-3.5-turbo y sus variantes ofrecen 3500 RPM y 90 000 o 180 000 TPM.

### 3.4. Prompt engineering

El término *prompt* hace referencia al proceso de construcción de instrucciones que resultan en el cumplimiento más efectivo de una tarea[48]. *Prompt engineering* se define como el conjunto de instrucciones recibidas por un LLM para refinar y establecer reglas específicas sobre un resultado más eficiente en la realización de tareas.

OpenAI ofrece las siguientes recomendaciones al momento de generar instrucciones para LLM.

- Utilizar los modelos más recientes.
- Priorizar las instrucciones al inicio del *prompt* y separar el *prompt* del contexto con caracteres como ###.

---

<sup>2</sup><https://platform.openai.com/account/rate-limits>

- Ser específicos, descriptivos y detallistas respecto al contexto, la salida, el tamaño, el formato, estilo, etc.
- Dar ejemplos de posibles salidas.
- Iniciar con un ejemplo de *prompt zero-shot* e ir gradualmente añadiendo detalles.
- Reducir las descripciones imprecisas.
- Priorizar las instrucciones sobre las restricciones.
- En el caso de generación de código, utilizar "palabras clave" de inicio para orientar el modelo hacia un patrón determinado.

## Capítulo 4

# Diseño del estudio

De acuerdo con lo desarrollado en capítulos anteriores, es posible plantear el hecho de utilizar los más recientes LLM para la extracción de entidades, atributos y relaciones que permitan la creación de ontologías en distintos ámbitos. Es por ello que en este capítulo se describe el procedimiento para la extracción de conocimiento de forma estructurada intentando mantener el contexto de la ontología objetivo. Al finalizar este apartado, se plantean escenarios que permitan realizar una evaluación de la coherencia de los distintos elementos de la ontología generada.

### 4.1. Consideraciones principales

Debido a que este trabajo busca hallar la forma en la que los últimos LLM cuentan con la capacidad de generar ontologías, es necesario el uso de recursos considerables de hardware para poder ejecutarlos o intentar algún *fine-tuning*. Por ese motivo, en este proyecto se hace uso del servicio de API (*Application programming interface*) que facilita OpenAI. De esta manera, el tiempo de respuesta se reduce y el coste de ejecución disminuye, en comparación con los recursos necesarios para ejecutar algún open-source LLM.

Todas las pruebas para este trabajo se realizan por medio de Flask(Python), un entorno que permite generar una interfaz web. Por otro lado, todas las consultas se manejan a partir de librerías de consultas HTTP en consonancia con otras herramientas como la API REQUEST PARALLEL PROCESSOR del repositorio cookbook de OpenAI. Esta herramienta permite un manejo en paralelo de las consultas a la API a fin de evitar sobrepasar el límite de consultas establecido, manejar errores, etc.

### 4.2. La generación de ontologías

El proceso para la generación de una ontología a partir de consultas a la API de los modelos de OpenAI consta de los siguientes pasos:

1. **Solicitud del usuario.** La persona que busca generar la ontología debe especificar dos palabras a modo de entrada. **Ontology context**, que es utilizada para delimitar el contexto de la ontología, así como una clase raíz o **root class** sobre la que se pretende construir la ontología. Estas dos palabras son ingresadas a través de un formulario web.

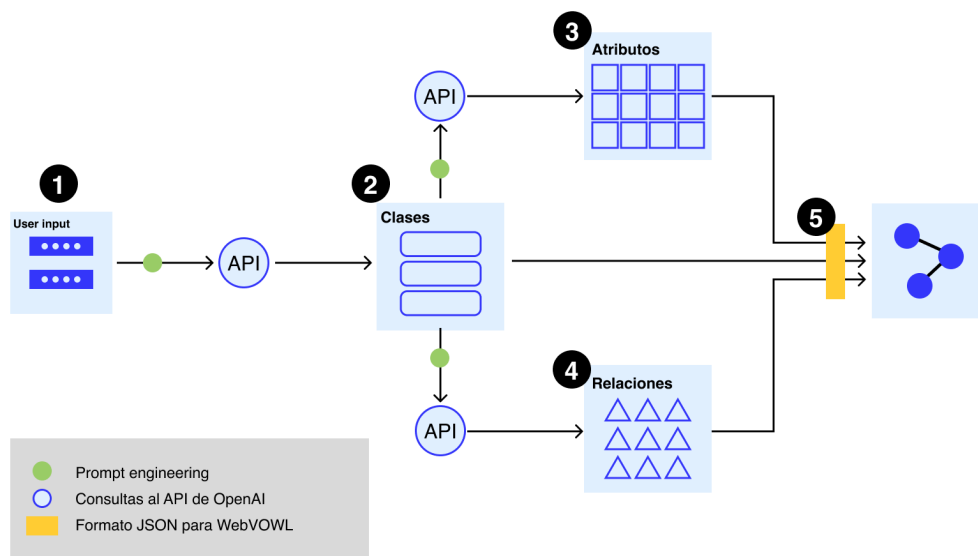


Figura 4.1: Proceso completo para la generación de ontologías Fuente: Generación propia

2. Las dos palabras del paso previo se emplean para generar una instrucción (prompt) más detallada con el objetivo de que el modelo de lenguaje pueda responder con una lista de las clases asociadas al **root class** en un **ontology context**.
3. La instrucción generada se envía a la API de OpenAI especificando el modelo y la *temperature* que se van a utilizar. Se recibe una lista de palabras que son las clases asociadas a la clase **root class**.
4. Cada palabra recibida de la API, es utilizada para generar una nueva instrucción diseñada para extraer una lista de atributos asociados a esa clase dentro del **ontology context**.
5. Cada clase recibida del API en el paso 3, en conjunto con la clase **root class** son utilizadas para generar una instrucción que permita identificar la relación entre ambas entidades, en ambos sentidos.
6. Los pasos del 1 al 5 pueden ser ejecutados nuevamente con distintos valores de *temperature* y modelos de lenguaje.

La figura 4.1 muestra el flujo completo de intercambio de información con la API de OpenAI que también se describe en el algoritmo 1.

### 4.2.1. Obtención de clases

Generar un *prompt* para el modelo de lenguaje que permita la obtención de clases asociadas a una **root class** en el contexto de **ontology context** requiere que el usuario sea lo más específico posible en cuanto a cómo desea la respuesta. Esto se debe a la necesidad que el resultado de la consulta sea procesado por otra porción de código para generar más instrucciones. En resumen, la respuesta del modelo de lenguaje debe ser *computer-friendly*.



---

### Algorithm 1 Ontology construction Algorithm

---

```

1: procedure ONTOLOGY CONSTRUCTION
2:   ontologyContext  $\leftarrow$  input()
3:   rootClass  $\leftarrow$  input()
4:   firstPrompt  $\leftarrow$  generatePrompt(ontologyContext, rootClass)
5:   relatedClasses  $\leftarrow$  queryLLM(firstPrompt)
6:   for each class  $\in$  relatedClasses do
7:     propertiesPrompt  $\leftarrow$  generatePropertiesPrompt(ontologyContext, class)
8:     relationsPrompt  $\leftarrow$  generateRelationsPrompt(ontologyContext, class, rootClass)
9:     properties  $\leftarrow$  queryLLM(propertiesPrompt)
10:    relations  $\leftarrow$  queryLLM(relationsPrompt)
11:  end for
12:  buildOntologyGraph(relatedClasses, properties, relations)
13: end procedure

```

---

```

1 def generate_prompt(ontology, root_node):
2   return "In the context of {}. List 4 classes that most likely have a relationship with {}"
   .format(ontology, root_node)

```

Listing 4.1: Función para generar una instrucción para obtener clases a partir del root node y el context

Una primera propuesta de una función dirigida a generar la instrucción puede ser con el formato de 4.1. Sin embargo, como se puede observar en la tabla 4.1, este tipo de instrucción genera respuestas de palabras enumeradas, separadas por saltos de línea sin seguir un patrón específico. Los resultados también permiten realizar la siguiente observación: en algunos casos la clase se compone de dos palabras. Ambos detalles, el formato del resultado y la extensión de cada clase, son importantes para el proceso de construcción de las próximas instrucciones.

Aunque este texto podría moldearse para tener un formato muy similar o igual a una lista en Python, es mejor aprovechar la capacidad del modelo de lenguaje a fin de que la respuesta ya incluya ese formato. Por otro lado, también es necesario hacer una modificación a la instrucción con el objetivo obtener únicamente una palabra para cada clase.

Instrucción	Respuesta
In the context of <b>Tourism</b> . List 4 classes that most likely have a relationship with <b>Museum</b>	1. Tourists 2. Educators 3. Researchers 4. Curators
In the context of <b>Building</b> . List 4 classes that most likely have a relationship with <b>Construction</b> .	1. Architects 2. Engineers 3. Landscapers 4. Contractors.
In the context of <b>Crystallographic</b> . List 4 classes that most likely have a relationship with <b>Cell</b>	-Space group -Crystal system -Point group -Lattice type

Cuadro 4.1: Respuestas obtenidas para **temperature** 0.7 en el modelo **davinci-003**

```

1 def generate_prompt(ontology, root_node):
2     return "In the context of a {} ontology. List 4 classes that have a relation with {}, one
           word each, separated by commas\n".format(ontology, root_node)

```

Listing 4.2: Función para generar una instrucción para obtener clases a partir del root node y el context. Modificado

El código que se muestra en 4.2 incluye dos modificaciones importantes respecto a la primera propuesta. El primero es "**one word each**" y el segundo es "**separated by commas**". Ambas instrucciones adicionales permiten sortear los posibles inconvenientes para cuando se desee obtener un resultado tipo lista que permita generar las siguientes instrucciones. Las respuestas del modelo de lenguaje ante este nuevo tipo de instrucción, con los mismos valores de **root class** y **ontology context**, se muestran en la tabla 4.2.

Instrucción	Respuesta
In the context of <b>Tourism</b> . List 4 classes that most likely have a relationship with <b>Museum</b>	Artwork, Culture, History, Location
In the context of <b>Building</b> . List 4 classes that most likely have a relationship with <b>Construction</b> .	Architects, Contractors, Materials, Regulations
In the context of <b>Crystallographic</b> . List 4 classes that most likely have a relationship with <b>Cell</b>	UnitCell, CrystalSystem, Space-Group, SymmetryElement

Cuadro 4.2: Respuestas obtenidas para **temperature** 0.7 en el modelo **davinci-003**. Instrucción mejorada

### 4.2.2. Obtención de atributos por clase

Se considera atributos de una clase a la lista de palabras que describen sus características dependiendo del contexto en el que se está buscando generar la ontología. Del paso anterior se recoge una lista de palabras que representan clases que tienen alguna relación con **root class**. A partir de ello, es posible generar una instrucción para cada palabra con el objetivo de obtener una lista, una vez más, de los atributos de cada clase.

De forma similar al paso anterior, es posible iniciar la tarea con instrucciones básicas y sin mucho detalle para intentar extraer los atributos de cada clase. El código en 4.3 muestra cómo el **root node** sigue siendo utilizado para indicar al modelo de lenguaje que no pierda el contexto de la tarea objetivo. En la tabla 4.3 se muestran las respuestas dadas para el caso de "**root node=Building**" y "**ontology context = Construction**".

```

1 def generate_prompt(ontology, root_node):
2     return "In the context of a {} ontology. Return a list of 4 attributes associated with a
           {} class\n".format(ontology, class_name)

```

Listing 4.3: Función para generar una instrucción para obtener los atributos de una clase en una ontología

Instrucción	Respuesta
In the context of a <b>Construction</b> ontology. Return a list of 4 attributes associated with a <b>Architects</b> class.	1. Name 2. Title 3. Address 4. Phone number
In the context of a <b>Construction</b> ontology. Return a list of 4 attributes associated with a <b>Contractors</b> class.	1. Contractors class 2. Contractors class has a name 3. Contractors class has a specialization 4. Contractors class has a location
In the context of a <b>Construction</b> ontology. Return a list of 4 attributes associated with a <b>Materials</b> class.	hardness, density, thermal conductivity, modulus of elasticity
In the context of a <b>Construction</b> ontology. Return a list of 4 attributes associated with a <b>Regulations</b> class.	1. The type of regulation 2. The governing body that created the regulation 3. The date the regulation was enacted 4. The date the regulation was last updated

Cuadro 4.3: Respuestas obtenidas para **temperature** 0.7 con el modelo **davinci-003**. Instrucción de obtención de atributos.

Nuevamente, el formato de respuesta supone un inconveniente para el objetivo final de la generación de la ontología. Idealmente, cada atributo debe ser una palabra y las palabras deben aparecer en un formato que permita identificar el inicio y el fin de cada una.

```

1 def generate_prompt(ontology, root_node):
2     return "In the context of a {} ontology. Return a list of 4 attributes associated with a
           {} class, one word each, separated by commas\n".format(ontology, class_name)

```

Listing 4.4: Función para generar una instrucción para obtener los atributos de una clase en una ontología. Modificado

La función descrita en 4.4 añade indicaciones sobre la longitud de la palabra para cada atributo, además del tipo de delimitación para identificar el inicio y fin de cada uno de estos. Se mantiene el caso de uso donde **root node=Building** y **ontology context = Construction**. La tabla 4.4 muestra las respuestas a partir de esta modificación en la instrucción. Este nuevo formato es mucho más sencillo de manejar e incorporar en una ontología.

### 4.2.3. Obtención de relación entre dos clases

La tarea de extraer la relación entre dos clases o entidades obtenidas anteriormente presenta inconvenientes parecidos a los que se vieron para la extracción de clases y en la extracción de atributos. De manera intuitiva, se podría realizar una pregunta directa como lo muestra la función 4.5. Desafortunadamente, como se muestra en la tabla 4.5. Las respuestas son extensas y existe texto redundante. A diferencia de las instrucciones para obtener clases o atributos, las relaciones son un poco más complejas porque no siempre están compuestas por una sola palabra.

```

1 def generate_prompt(ontology, root_node):

```

Instrucción	Respuesta
In the context of a <b>Building</b> ontology. Return a list of 4 attributes associated with a <b>Architects</b> class, one word each, separated by commas	Name, Specialty, Clients, Experience
In the context of a <b>Building</b> ontology. Return a list of 4 attributes associated with a <b>Contractors</b> class, one word each, separated by commas	Color, Shape, Weight, Durability
In the context of a <b>Building</b> ontology. Return a list of 4 attributes associated with a <b>Materials</b> class, one word each, separated by commas	Compliance, Safety, Zoning, Inspection
In the context of a <b>Building</b> ontology. Return a list of 4 attributes associated with a <b>Regulations</b> class, one word each, separated by commas	Type, Location, Cost, Quality

Cuadro 4.4: Respuestas obtenidas para **temperature** 0.7 con el modelo **davinci-003**. Instrucción de obtención de atributos modificada.

```
2 return "In the context of a {} Ontology. Define a relationship between {} and {}".format(
    ontology, class1, class2)
```

Listing 4.5: Función para generar una instrucción para obtener los atributos de una clase en una ontología. Modificado

Como se ha mencionado anteriormente, las relaciones entre dos clases no necesariamente están formadas por una sola palabra, sino que pueden llegar a ser hasta tres. Con esa observación, es posible modificar la instrucción según el código 4.6 donde se ha añadido la restricción "**three words max**". Esto permite obtener resultados más cortos y sencillos como los que se muestran en la tabla 4.6. Las relaciones que se han podido extraer con esta instrucción mejorada son mucho mejores y concisas que las iniciales, al punto que pueden ser consideradas para su uso en la construcción de ontologías.

```
1 def generate_prompt(ontology, root_node):
2     return "In the context of a {} Ontology. Define, in three words max, a relationship
    between {} and {}".format(ontology, class1, class2),
```

Listing 4.6: Función para generar una instrucción para obtener los atributos de una clase en una ontología. Modificado

El proceso de extracción de relaciones es únicamente con la clase raíz **root class** y en ambas direcciones. Esto quiere decir que se extrae la relación entre la clase raíz sobre la clase descubierta en el primer paso, y viceversa.

En suma, se han aplicado algunas de las recomendaciones de OpenAI para la creación de *prompts*. En este ejemplo, se ha añadido extensión y formato al texto salida del LLM. Las figuras 4.2, 4.3, 4.4 muestran la diferencia entre los *prompt* iniciales y los *prompts* finales, más específicos. La instrucción final es la que se utiliza para la interacción con la API de OpenAI.

**Prompts - Clases**



-  In the context of a **Tourism** ontology.  
List 4 classes that most likely have a relationship with **Museum**
-  In the context of a **Tourist** ontology.  
List 4 classes that have a relation with **Museum**, one word each, separated by commas  
**+extensión**      **+formato**

Figura 4.2: Proceso de la generación de *prompts* para la extraer clases para la construcción de una ontología Fuente: Generación propia

**Prompts - Atributos**



-  In the context of a **Tourism** ontology.  
Return a list of 4 attributes associated with a **Museum** class
-  In the context of a **Tourist** ontology.  
Return a list of 4 attributes associated with a **Museum** class, one word each, separated by commas  
**+extensión**      **+formato**

Figura 4.3: Proceso de la generación de *prompts* para la extraer clases atributos de una clase para la construcción de una ontología Fuente: Generación propia

**Prompts - Relaciones**



-  In the context of a **Building** Ontology.  
Define a relationship between **Construction** and **Architects**
-  In the context of a **Building** Ontology.  
Define, in three words max, a relationship between **Construction** and **Architects**  
**+extensión**

Figura 4.4: Proceso de la generación de *prompts* para la extraer la relación entre clases para la construcción de una ontología Fuente: Generación propia

Instrucción	Respuesta
In the context of a <b>Building</b> Ontology. Define a relationship between <b>Construction</b> and <b>Architects</b>	The relationship between Construction and Architects is that Architects <b>design</b> the plans for a building, and Construction is responsible for bringing those plans to life by building the structure.
In the context of a <b>Building</b> Ontology. Define a relationship between <b>Construction</b> and <b>Contractors</b>	The relationship between Construction and Contractors is that Construction <b>is responsible for overseeing</b> the entire building process, while Contractors are responsible for completing specific tasks such as plumbing, electrical work, and other specialized services.
In the context of a <b>Building</b> Ontology. Define a relationship between <b>Construction</b> and <b>Materials</b>	The relationship between Construction and Materials is that Construction <b>is responsible for procuring</b> the necessary materials and supplies needed to build a structure, while Materials are the physical components that are used in the construction process.
In the context of a <b>Building</b> Ontology. Define a relationship between <b>Construction</b> and <b>Regulations</b>	The relationship between Construction and Regulations is that Construction <b>is responsible for adhering</b> to the relevant regulations and laws governing the construction process, while Regulations are the rules and regulations that must be followed in order to ensure a safe and compliant building.

Cuadro 4.5: Respuestas obtenidas para **temperature** 0.7 con el modelo **davinci-003**. Instrucción de obtención de relación entre dos clases.

#### 4.2.4. Construcción de la ontología

Una vez establecido el formato que debe tener cada instrucción para extraer clases, atributos y relaciones, es posible construir ontologías de dominio específico. A partir del formato de respuesta de cada instrucción, es posible desencadenar otro conjunto de instrucciones para continuar con el proceso de extracción de información.

Todas las instrucciones son almacenadas en archivos dentro de una carpeta nombrada por un identificador único por cada solicitud. Esto permite a la librería API REQUEST PARALLEL PROCESSOR ejecutar cada instrucción de acuerdo a los parámetros necesarios sin exceder los RPM ni los TPM establecidos por OpenAI. De forma complementaria, cada instrucción contiene información adicional como metadata para poder identificar los valores que permitieron generar cada instrucción.

Una vez que la última solicitud a la API sobre la relación entre dos clases ha termina-

Instrucción	Respuesta
In the context of a <b>Building</b> Ontology. Define, in three words max, a relationship between <b>Construction</b> and <b>Architects</b>	Designed by
In the context of a <b>Building</b> Ontology. Define, in three words max, a relationship between <b>Construction</b> and <b>Contractors</b>	Hires For
In the context of a <b>Building</b> Ontology. Define, in three words max, a relationship between <b>Construction</b> and <b>Materials</b>	Uses materials
In the context of a <b>Building</b> Ontology. Define, in three words max, a relationship between <b>Construction</b> and <b>Regulations</b>	Governs Compliance

Cuadro 4.6: Respuestas obtenidas para **temperature** 0.7 con el modelo **davinci-003**. Instrucción de obtención de relación entre dos clases. Mejorado

do, el contenido de la carpeta de la ontología pasa a incluir archivos con el detalle de cada instrucción y la respuesta recibida del modelo de lenguaje. Estos archivos son utilizados para generar otro con formato Visual Notation for OWL Ontologies (VOWL) para que pueda ser visualizado por el usuario en un navegador web con WebVOWL<sup>1</sup>. La imagen 4.5 muestra una ontología generada a partir de **root class=Hotel** y **ontology context = Tourism** con **temperature=0.7** utilizando el modelo de texto **text-davinci-003**.

### 4.3. Evaluación de la cobertura

Anteriormente, se ha mencionado que una de las críticas más comunes hacia los LLM es su falta de conocimiento, mostrando en muchas ocasiones alucinaciones al momento de generar texto. En esta sección se propone un procedimiento que permite comparar la similitud de los atributos generados por el LLM versus la información disponible en un repositorio de conocimiento abierto como Wikidata. Esto nos permitirá analizar la cobertura que cubre un LLM con distintos parámetros de modelo y temperatura.

El dataset de *Few-Shot Relation Classification Dataset* (FewRel)[49] contiene 70 000 oraciones donde personas colaboradoras han realizado anotaciones de 100 relaciones entre entidades de Wikidata. Este conjunto de datos no solo contiene la relación, sino también los identificadores en Wikidata correspondientes a cada entidad. Este identificador permite conocer, a través de una relación "**instance of**", la instancia a la que pertenece cada entidad parte de la relación. Luego, es posible extraer por medio de un SPARQL QUERY las propiedades de estas entidades. Por otro lado, para generar atributos desde el LLM, es necesario llevar a cabo consultas, esta vez produciendo variaciones en los parámetros **model** y **temperature**. Para esta prueba se

<sup>1</sup><https://github.com/VisualDataWeb/WebVOWL>

### 4.3. Evaluación de la cobertura

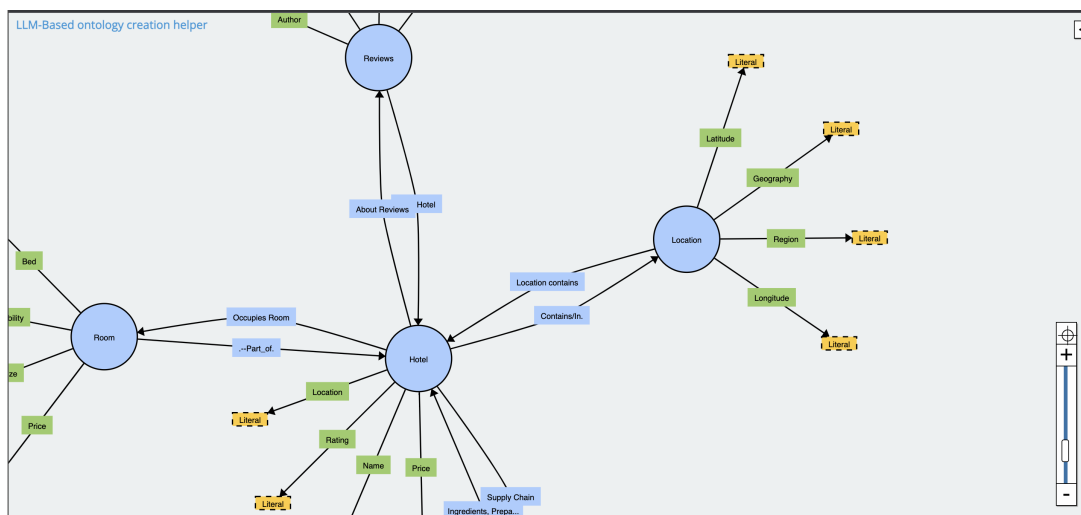


Figura 4.5: Ontología generada a partir de **root class=Hotel** y **ontology context = Tourism** Fuente: Generación propia

utilizaron los modelos descritos en la tabla 4.7. Una vez obtenidos los atributos, sean de Wikidata o a través del LLM, tiene lugar su comparación y análisis.

Modelo	Características	Max tokens
gpt-4	Versión mejorada a GPT-3.5. Optimizado para conversaciones. Con un conocimiento general más amplio y mejor precisión a modelos anteriores	8,192
gpt-3.5-turbo	Modelo optimizado para conversaciones, y más barato que el modelo text-davinci-003	4,096
gpt-3.5-turbo-16k	Mismas capacidades que gpt-3.5-turbo con 4 veces el tamaño de contexto	16,384
text-davinci-003	Parte de los modelos GPT-3.5. Con capacidad para entender lenguaje natural o código. El finetuning de este modelo se hizo con Reinforcement Learning	4,097
text-davinci-002	Parte de los modelos GPT-3.5. Con capacidad para entender lenguaje natural o código. El finetuning de este modelo se hizo con métodos supervisados	4,097
davinci, curie, babbage	Son los modelos originales GPT-3 lanzados en 2019. Con capacidad para entender y generar lenguaje natural	2,049

Cuadro 4.7: Lista de modelos de OpenAI utilizados para evaluación de entidades y sus características. Fuente: www.openai.com

El dataset utilizado para esta prueba tuvo que ser modificado debido a que al aplicar la relación "**instance of**" existen entidades duplicadas y entidades sin objetos conectados a través de "**properties for this type**". El dataset de entrenamiento (*train*) contiene 44 000 filas que representan 2642 instancias. De estas 2642 instancias,



## **Diseño del estudio**

---

únicamente 342 contienen información sobre sus propiedades.

Las consultas sobre propiedades al LLM utilizan como parámetros a 8 distintos modelos, jugando con temperaturas en el rango  $[0 - 2]$  en intervalos de 0,05. Una vez obtenidas las propiedades, se analizan tres escenarios: el primero corresponde a la identificación de las coincidencias entre los atributos generados por el LLM y los atributos existentes de acuerdo con Wikidata; el segundo, busca analizar la frecuencia con la que los atributos son generados por el LLM independientemente del parámetro que reciba; el tercer escenario revisa la diferencia entre los distintos modelos utilizados para la consulta.



## Capítulo 5

# Resultados y discusiones

Este apartado describe algunos casos de uso con el objetivo de volver sobre planteamientos ya expuestos para comprenderlos mejor y profundizar en ellos. La primera parte consiste en dos casos de uso para la generación de ontologías para dos dominios específicos: "**Air Traffic**" y "**Tourism**". La segunda parte consiste en un análisis de la cobertura de atributos específicos de tres escenarios comparándola con información disponible en Wikidata. Los datos que se analizan se corresponden a "**international airport**", "**video games**" y "**ceremony awards**".

### 5.1. Generación de ontologías a partir de LLM

Según lo descrito en el capítulo anterior, y siguiendo el procedimiento que sugiere la figura 4.1, es posible construir una ontología básica a partir de dos parámetros iniciales otorgados por el usuario.

#### 5.1.1. Caso 1: Air Traffic Ontology

El primer caso que se simula consiste en la creación de una ontología de **Air Traffic** donde la clase raíz es "**Airport**". A través de un formulario web se introducen dichos valores, como se muestra en la imagen 5.1. En primera instancia, se generan las clases relacionadas con la clase raíz dentro del contexto dado. Posteriormente, de forma paralela, se realiza la consulta al LLM sobre las propiedades y relaciones entre cada una de estas clases y la clase raíz.

La figura 5.2 muestra los distintos elementos generados para la construcción de una ontología de **Air Traffic** con clase raíz "**Airport**". Finalmente, cuando se le proporciona el formato adecuado WebVOWL, la ontología se exporta en formato TTL para que pueda ser utilizada con otras herramientas. Estos elementos generados se detallan mejor en la figura 5.3. La figura 5.4 muestra el grafo completo con clases, atributos y relaciones entre la clase raíz y las otras generadas a partir de las consultas al LLM.

#### 5.1.2. Caso 2: Tourism Ontology

En este caso se busca generar una ontología de **Tourism** donde la clase raíz es "**Hotel**". De la misma forma que en el primer caso, los valores iniciales se introducen

## 5.1. Generación de ontologías a partir de LLM

### Distilling Ontologies from Large Language Models

Generate

Figura 5.1: Interfaz de usuario para el ingreso de Contexto y la clase raíz. Fuente: Generación propia

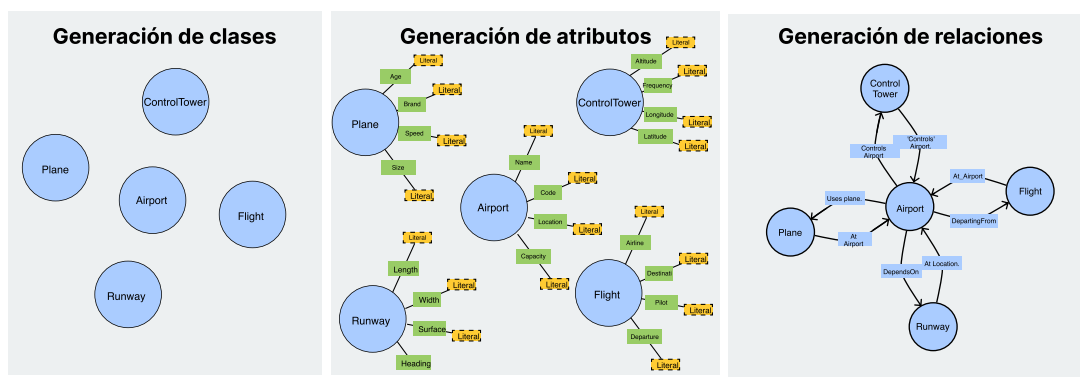


Figura 5.2: Proceso de generación de los elementos para la ontología de "Air Traffic". Modelo **text-davinci-003** temperatura **0.7**. Fuente: Generación propia

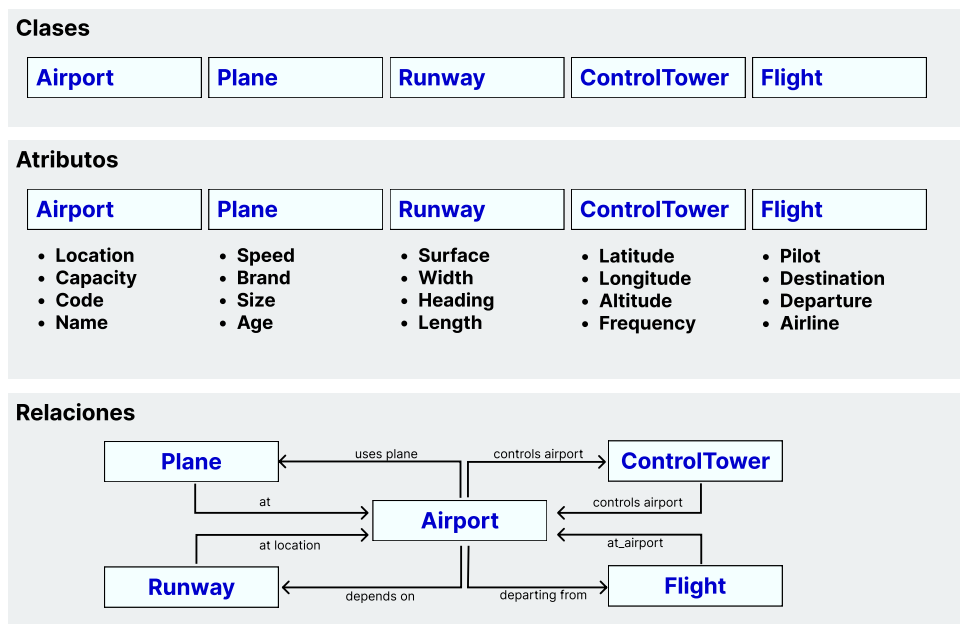


Figura 5.3: Clases, atributos y relaciones generadas para la ontología "Air Traffic". Modelo **text-davinci-003** temperatura **0.7**. Fuente: Generación propia

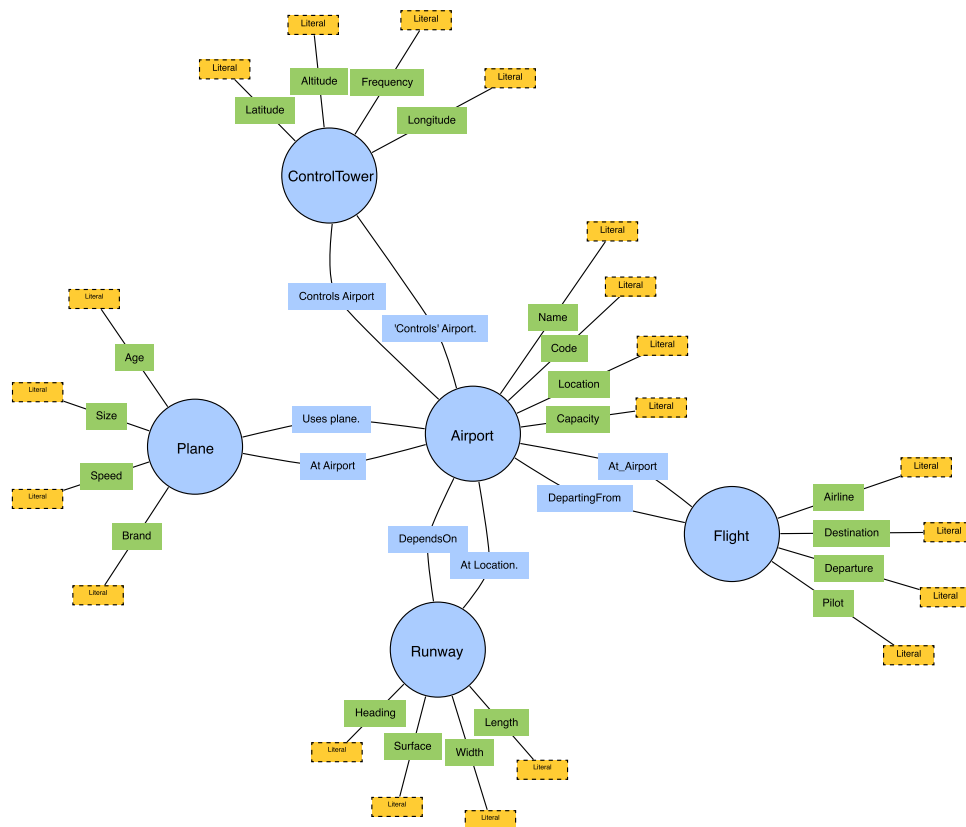


Figura 5.4: Clases, atributos y relaciones de la ontología "*Air Traffic*". Modelo **text-davinci-003** temperatura **0.7**. Fuente: Generación propia

## 5.2. Cobertura de los atributos generados por LLM

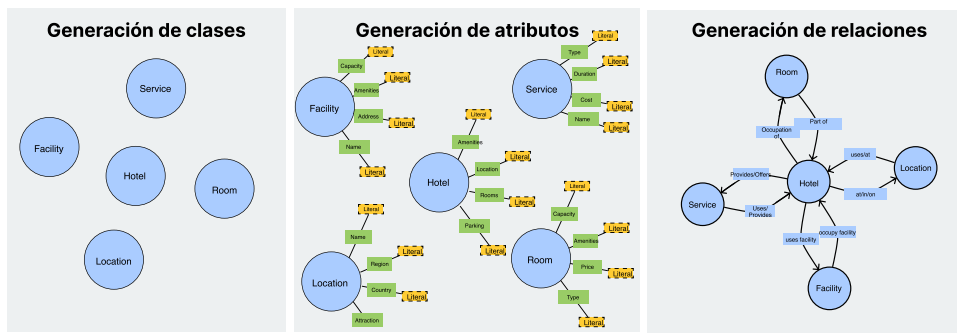


Figura 5.5: Proceso de generación de los elementos para la ontología de "**Tourism**". Modelo **text-davinci-003** temperatura **0.7**. Fuente: Generación propia

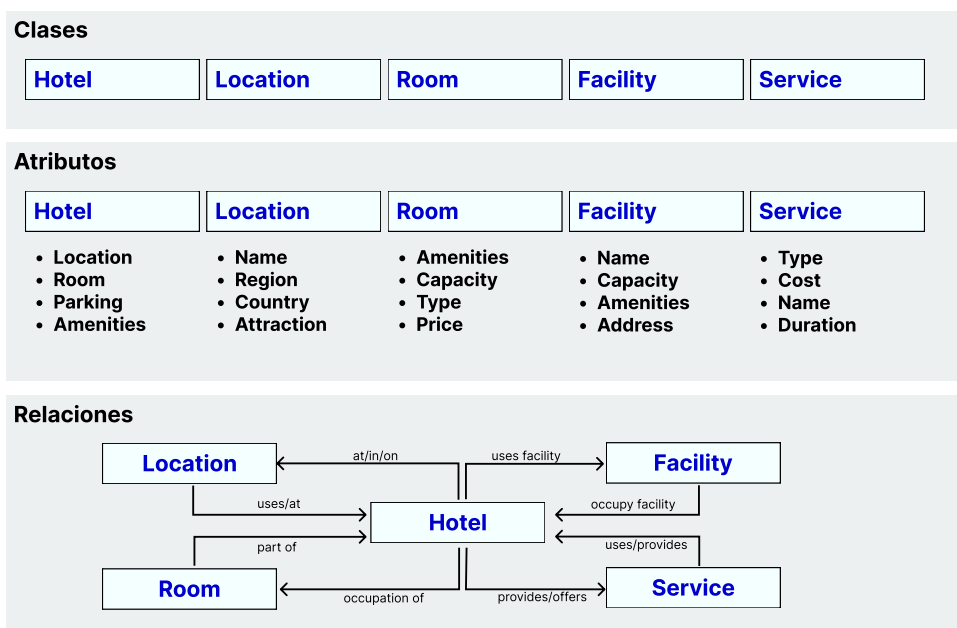


Figura 5.6: Clases, atributos y relaciones generadas para la ontología "**Tourism**". Modelo **text-davinci-003** temperatura **0.7**. Fuente: Generación propia

desde el formulario web, para posteriormente recorrer el proceso de construcción descrito en la figura 4.1.

La figura 5.5 muestra en cada paso cómo se están generando elementos para la construcción de la ontología de "**Tourism**" con clase raíz "**Hotel**". Estos elementos son mejor descritos en la figura 5.6. El grafo completo de la ontología y todos sus elementos se muestran en la figura 5.7.

## 5.2. Cobertura de los atributos generados por LLM

### 5.2.1. Caso 1: *International airport*

Una de las 346 instancias disponibles para comprobar su alcance y distribución es "**international airport**" que tiene como identificador Q644371 en Wikidata. Con el objetivo de comprobar el alcance de esta instancia, es posible hacer consultas al LLM

## Resultados y discusiones

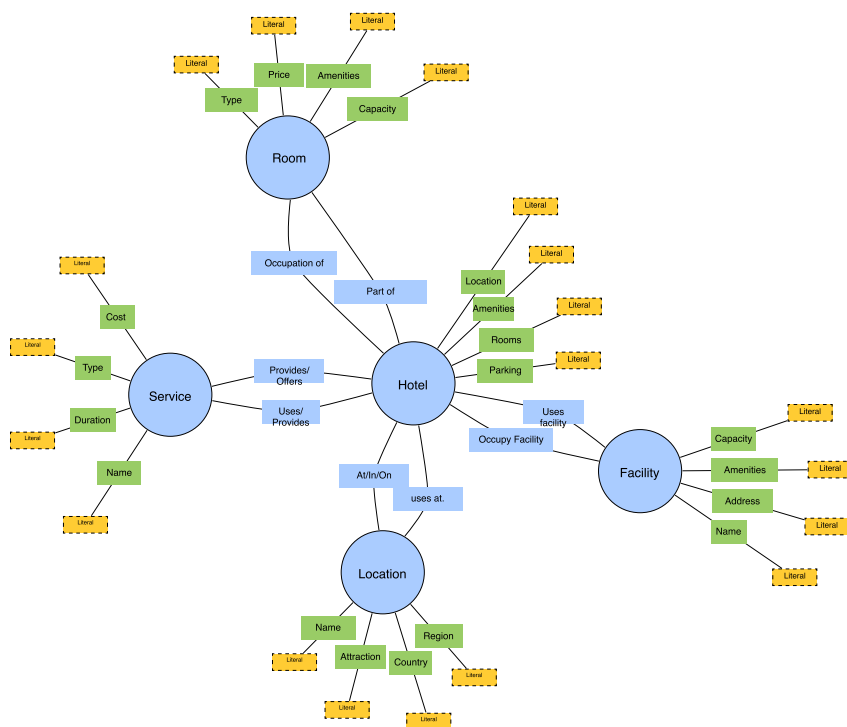


Figura 5.7: Clases, atributos y relaciones de la ontología "**Tourism**". Modelo **text-davinci-003** temperatura **0.7**. Fuente: Generación propia

en el contexto de "**Air Traffic**" con la clase raíz **international airport**.

Según la información de Wikidata, existen 11 atributos asociados a la clase "**international airport**". Mientras que, de acuerdo a las respuestas del LLM con los parámetros utilizados, se han generado un conjunto de hasta 114 propiedades distintas. Al realizar una comparación por cada atributo, es posible observar que los atributos con más coincidencias en los distintos modelos han sido los de **location**, **runway** y códigos **ICAO** e **IATA**.

Por otro lado, si analizamos la figura 5.9 que muestra la distribución de los atributos obtenidos a partir de las consultas realizadas al modelo. se observa que existen claramente atributos con mayor presencia que otros como "**location**" y "**runway**".

Modelo	Entity: International airport vs Airport Class										
	P01	P02	P03	P04	P05	P06	P07	P08	P09	P10	P11
gpt-4		x	x		x			x		x	x
gpt-3.5-turbo		x	x		x				x	x	x
gpt-3.5-turbo-16k		x	x		x			x		x	x
text-davinci-003		x	x					x	x	x	x
text-davinci-002	x	x	x		x				x	x	x
davinci											

Propiedades en wikidata
P01 scheduled service destination
P02 runway
P03 coordinate location
P04 place served by transport hub
P05 country
P06 image
P07 owned by
P08 located in the administrative territorial entity
P09 named after
P10 ICAO airport code
P11 IATA airport code

Figura 5.8: Evaluación de la cobertura de la clase "**International airport**" versus etiquetas de Wikidata Fuente: Generación propia

## 5.2. Cobertura de los atributos generados por LLM

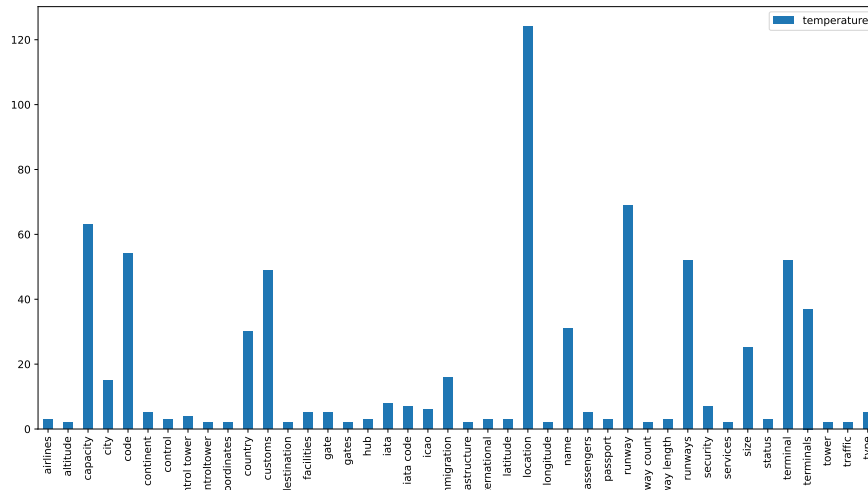


Figura 5.9: Frecuencia de atributos para la clases **International airport** originada por las consultas al LLM Fuente: Generación propia

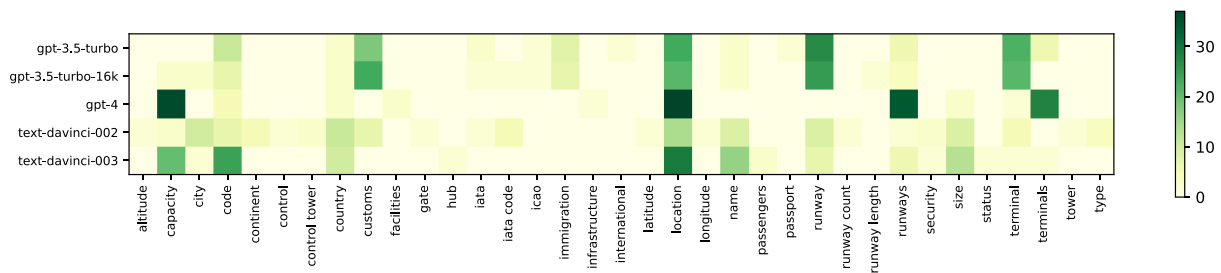


Figura 5.10: Mapa de calor por cada atributos de "**International airport**" originada por las consultas al LLM, diferenciada por tipo de modelo Fuente: Generación propia

La figura 5.10 muestra este tipo de distribución para atributos con más de una (01) aparición. Aquí es posible observar que existen atributos persistentes independientes del modelo y de la temperatura utilizada para la consulta al LLM. Otra característica interesante de la consulta al LLM se aprecia en la figura 5.11 , que muestra la dispersión de cada atributo por cada modelo utilizado para las consultas. Lo que puede verse es que existe mucha mayor dispersión en el conjunto general de atributos por los modelos "más pequeños" como **text-davinci-002** a diferencia del **GPT-4** donde, independiente del parámetro de temperatura, intenta mantener un conjunto pequeño de atributos como respuesta.

### 5.2.2. Caso 2: Video game

Otra instancia disponible para comparar el alcance de las consultas al LLM es la de "**video game**" con identificador Q7889 en Wikidata. Para ello, se realiza una consulta al LLM con los parámetros de contexto "**Store**" sobre la clase "**video game**". De acuerdo con Wikidata, esta entidad dispone de 31 atributos asociados a "**video game**"; mientras que las respuestas a las consultas al LLM generaron hasta 77 atributos entre los distintos modelos y temperaturas utilizadas. La imagen 5.12 representa la



## Resultados y discusiones

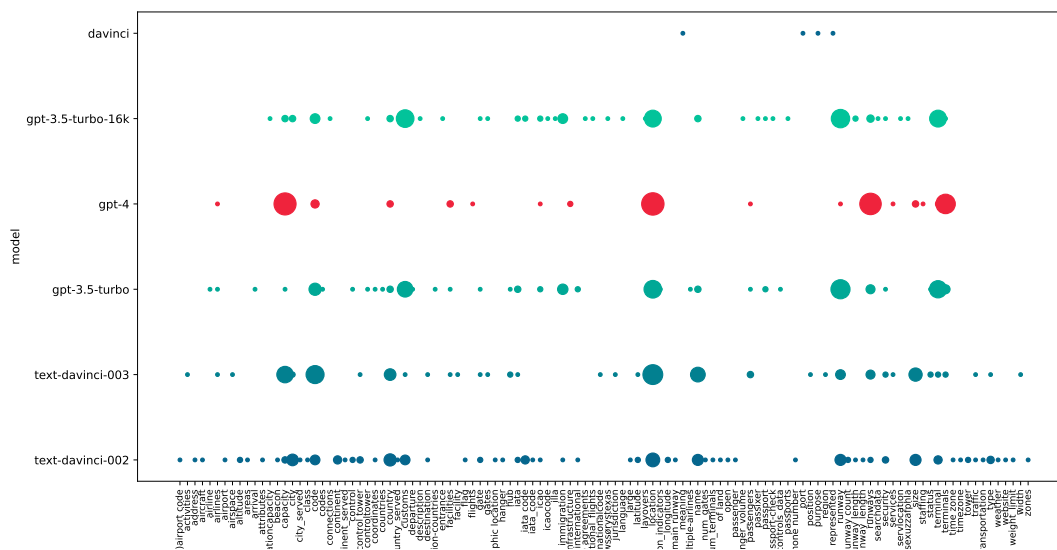


Figura 5.11: Gráfico de dispersión por cada atributos de **international airport** originada por las consultas al LLM, diferenciada por tipo de modelo Fuente: Generación propia

cobertura de estos 77 atributos sobre aquellos que existen en Wikidata. Se observa que las de mayor incidencia son las de "**platform**", "**release date**", algún tipo de "**rating**", "**genre**" y "**developer**".

La distribución de estos atributos, independiente del modelo puede, que se observa en la figura 5.13 muestra que existen atributos con una mayor presencia como "**genre**" o "**platform**" sobre el resto de atributos.

La figura 5.14 presenta un gráfico de aquellos atributos con más de una (01) aparición en las consultas al LLM. Esta imagen indica una persistencia de algunos atributos en todos los modelos utilizados sin importar el parámetro de temperatura utilizado para la consulta. Al igual que el caso anterior, si se observa el gráfico 5.15 de dispersión, se aprecia qué modelos generan atributos más aleatorios y qué modelos generan

Entity: Video games vs Video Game Class																																	
Modelo	P01	P02	P03	P04	P05	P06	P07	P08	P09	P10	P11	P12	P13	P14	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24	P25	P26	P27	P28	P29	P30	P31		
gpt-4	x				x		x	x	x	x	x																				x	x	x
gpt-3.5-turbo	x				x		x	x	x	x	x																				x	x	x
gpt-3.5-turbo-16k	x				x		x	x	x	x	x																				x	x	x
text-davinci-003	x				x		x	x	x	x	x																				x	x	x
text-davinci-002	x				x		x	x	x	x	x																				x	x	x

Propiedades en wikidata								
P01	platform	P11	GSRR rating	P21	IGN game ID			
P02	game mode	P12	Melancolic ID	P22	GameSpot game ID			
P03	input device	P13	Steam application ID	P23	Nintendo GameID (GameCube/Wii)			
P04	country of origin	P14	MobyGames game ID	P24	Microsoft Store product ID			
P05	publication date	P15	MAME ROM name	P25	North America PlayStation Store ID			
P06	Wine AppDB ID	P16	PlayStation ID	P26	Europe PlayStation Store ID			
P07	CERO rating	P17	OpenCritic ID	P27	Igromania ID			
P08	ESRB rating	P18	Nintendo of America Game Store ID	P28	director			
P09	PEGI rating	P19	Guardiana ID	P29	publisher			
P10	USK rating	P20	Giant Bomb ID	P30	genre			
				P31	developer			

Figura 5.12: Evaluación de la cobertura de la clase "**video game**" versus etiquetas de Wikidata Fuente: Generación propia

## 5.2. Cobertura de los atributos generados por LLM

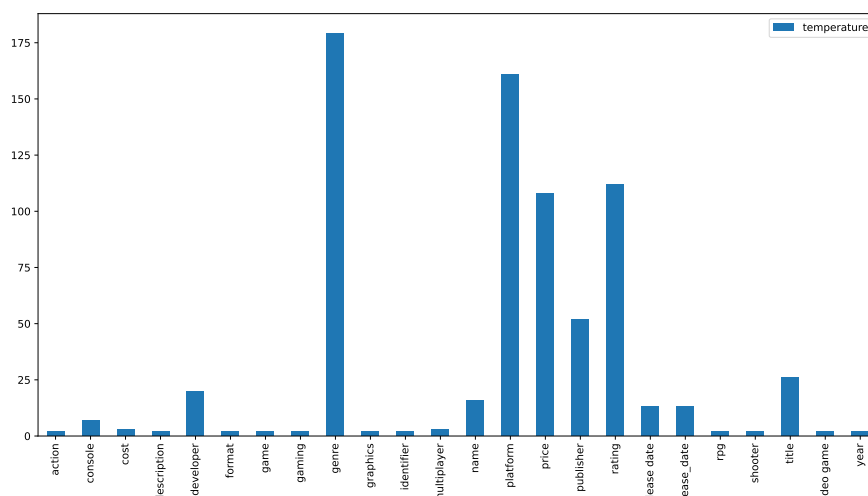


Figura 5.13: Frecuencia de los atributos para la clase "**video game**" originada por las consultas al LLM. Fuente: Generación propia

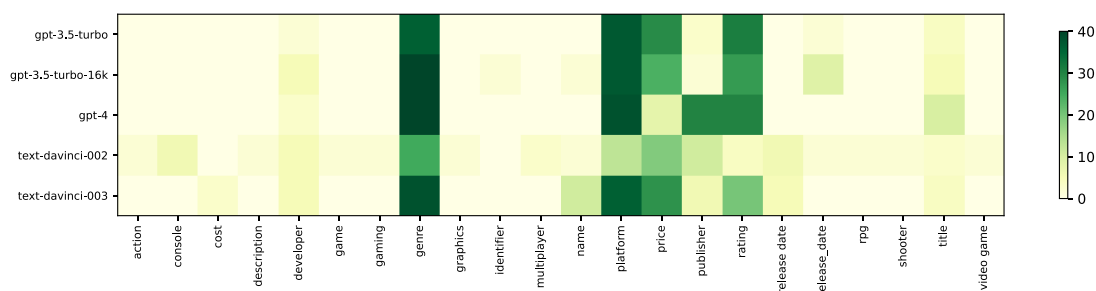


Figura 5.14: Mapa de calor por cada atributo de "**video game**" originada por las consultas al LLM diferenciada por cada modelo. Fuente: Generación propia

atributos más estables con la variación del parámetro de temperatura.

### 5.2.3. Caso 3: Ceremony awards

La instancia "**ceremony awards**" con identificador Q4504495 en Wikidata cuenta con 12 atributos. Estos atributos son comparados con los 78 atributos generados por el LLM con parámetros "**events**" como contexto y "**award ceremony**" como clase raíz. La imagen 5.16 muestra la cobertura de estos 78 atributos que incluye principalmente a "**location**", "**point in time(date)**", "**organizer**" y "**winner**".

Independiente del modelo y la temperatura utilizada, la figura 5.17 indica la frecuencia de atributos con más de una aparición entre las respuestas del LLM. En este gráfico resaltan atributos como "**date**", "**location**" y "**winner**" sobre el resto de atributos.

Las figuras 5.18 muestran la persistencia de ciertos atributos en todos los modelos utilizados. En cambio, la figura 5.19 presenta cómo algunos modelos generan atributos de forma más dispersa que otros independiente del valor temperatura en la consulta al LLM.

## Resultados y discusiones

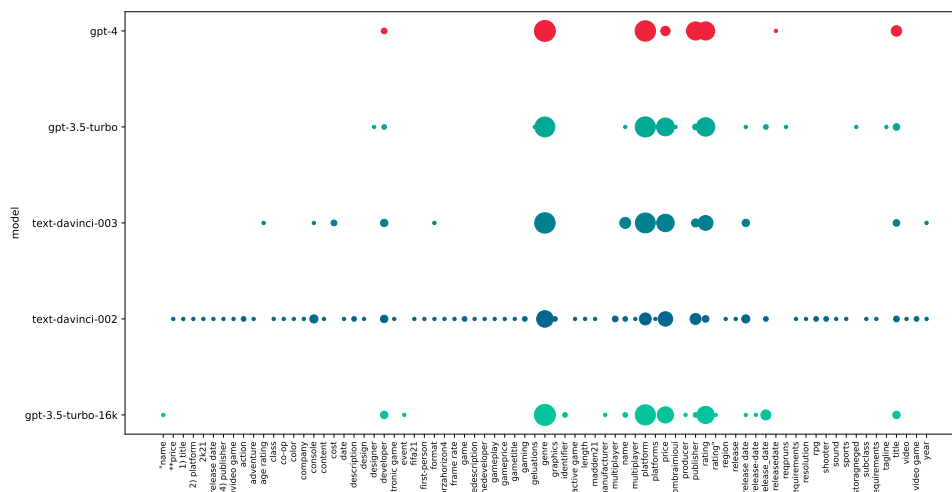


Figura 5.15: Mapa de dispersión por cada atributo de "**video game**" originada por las consultas al LLM diferenciada por cada modelo. Fuente: Generación propia

Entity: ceremony award vs Ceremony Award Class												
Modelo	P01	P02	P03	P04	P05	P06	P07	P08	P09	P10	P11	P12
gpt-4	x			x	x	x		x				
gpt-3.5-turbo	x	x		x	x	x		x				
gpt-3.5-turbo-16k	x	x		x	x			x				
text-davinci-003	x			x	x			x				
text-davinci-002	x			x	x			x				

Propiedades en wikidata
P01 location
P02 presenter
P03 edition number
P04 point in time
P05 organizer
P06 number of participants
P07 facet of
P08 winner
P09 awarded for period
P10 follows
P11 followed by
P12 part of the series

Figura 5.16: Evaluación de la cobertura de la clase "**ceremony awards**" versus etiquetas de Wikidata. Fuente: Generación propia

## 5.2. Cobertura de los atributos generados por LLM

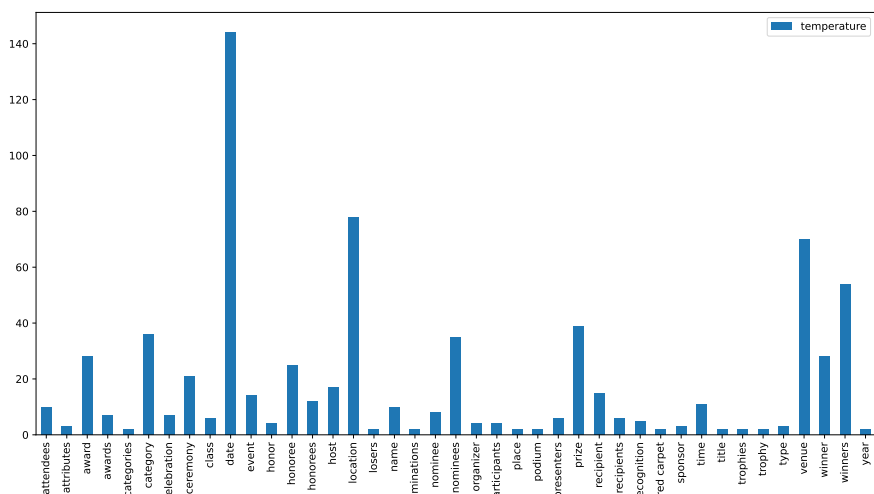


Figura 5.17: Frecuencia de atributos para la clase "**ceremony awards**" generadas por las consultas al LLM. Fuente: Generación propia

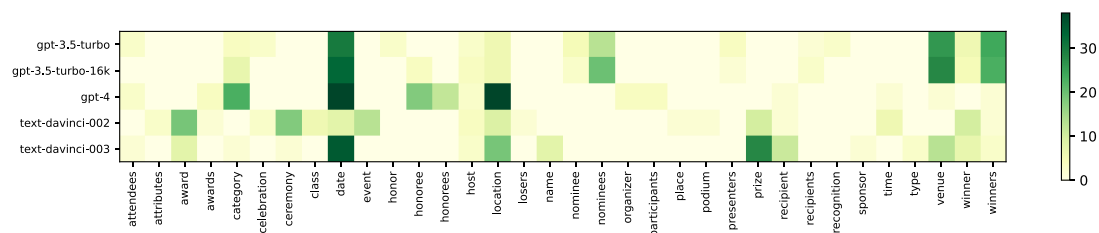


Figura 5.18: Mapa de calor por cada atributo de "**ceremony awards**" originada por las consultas al LLM diferenciada por cada modelo. Fuente: Generación propia

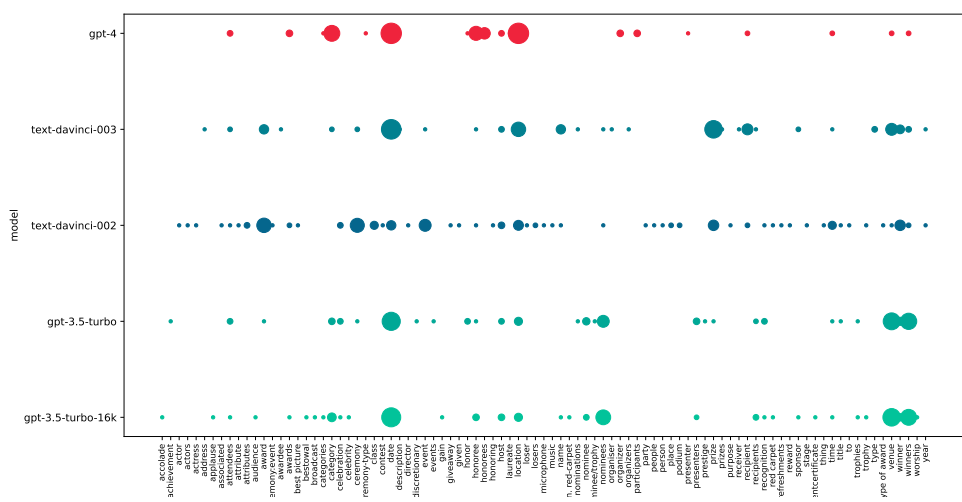


Figura 5.19: Mapa de dispersión por cada atributo de "**ceremony awards**" originada por las consultas al LLM diferenciada por cada modelo. Fuente: Generación propia

## Capítulo 6

# Conclusiones y trabajo futuro

Los apartados previos han descrito las características del proceso de construcción de ontologías y que este contiene tareas que pueden ser automatizadas. Ante esta situación, se plantea el uso de LLM como una posible herramienta para acelerar su construcción.

Asimismo, se han evaluado los resultados obtenidos por un LLM llevando a cabo una comparación con información existente en una base de conocimiento como Wikidata.

### 6.1. Conclusiones

Como uno de los resultados del trabajo se puede concluir que es posible implementar una herramienta capaz de generar ontologías haciendo uso de las características de un LLM como la extracción de clases, propiedades y relaciones a partir de instrucciones generadas de forma semi-automática. Además, resulta evidente que existen LLM con la capacidad de entender instrucciones que soliciten un formato específico de salida. Esta característica permite que las salidas a alguna instrucción al modelo puedan conectarse automáticamente a otras instrucciones más, generando una cadena de instrucciones con un fin específico sin mayor interacción por parte de la persona, como por ejemplo, interconectar el flujo de construcción de ontologías.

Sin embargo, al igual que cualquier otro método de automatización, la evaluación de este proceso puede resultar compleja debido a que no existe un *gold standard* al que se apunta o un conjunto de reglas definida para analizar si una ontología es buena o mala. Por ese motivo, la evaluación realizada en este trabajo a distintos elementos generados también nos ha servido para observar qué tanto conoce el LLM sobre algunos conceptos del mundo en un contexto específico. Lo que se ha podido observar es que existen atributos que son generados con mayor frecuencia por los LLM ante el mismo prompt. Estos atributos usualmente son aquellos que coinciden con información de Wikidata.

Otro factor interesante a observar es el comportamiento del LLM en sus distintas versiones. Lo que se ha podido observar es que, a pesar que la temperatura sea definida como un parámetro de aleatoriedad en los modelos de lenguaje, es posible notar que las respuestas de los modelos más grandes corresponden a un conjunto más pequeño de atributos. Mientras que modelos más pequeños generan un conjunto

más grande y aleatorio frente al mismo *prompt*. Además que estos últimos tienen mayor complicación al generar un formato de salida solicitado en el *prompt*.

Aunque este trabajo no pretende ser una herramienta determinante para la generación de ontologías, los procesos que se proponen pueden ser de gran ayuda. No solo a desarrolladores expertos cuando aborden un dominio desconocido sino también para quienes se inician en el desarrollo de ontologías y necesitan algunas referencias para iniciar su camino.

## 6.2. Líneas de trabajo futuro

Al momento plantear la propuesta de este trabajo, uno de los problemas fue la incertidumbre sobre si ChatGPT o los modelos disponibles a través de su API seguirían disponibles en los próximos meses. Otro inconveniente fue la naturaleza no-abierta de estos modelos desarrollados OpenAI, que aunque existan algunos detalles en los artículos, no es el modelo final publicado o disponible en el API. Lo que pasó al final fue que ChatGPT generó un movimiento de desarrollo de LLM desde distintos frentes. Tal vez el más importante para la comunidad científica fue la publicación de LLaMA que puso a disposición los pesos del modelo. Esto generó múltiples modelos derivados de LLaMA.

A manera de continuar con esta línea de trabajo, es posible señalar lo siguiente:

- Utilizar otros modelos de lenguaje como Bard para realizar las mismas pruebas.
- Utilizar los LLM para popular otras partes de la ontología como el etiquetado por idiomas o la descripción de las clases.
- Automatizar la evaluación realizada para cada atributo y extenderla a las relaciones entre entidades.
- Realizar un *finetuning* del LLM para obtener respuestas de un dominio más especializado.

# Bibliografía

- [1] T. R. Gruber, “A translation approach to portable ontology specifications,” *Knowledge Acquisition*, vol. 5, pp. 199–220, 1993.
- [2] M. R. Saeed, C. Chelmiss, and V. K. Prasanna, “Automatic integration and querying of semantic rich heterogeneous data: Laying the foundations for semantic web of things,” in *Managing the Web of Things*, 2017.
- [3] T. Berners-Lee, J. A. Hendler, and O. Lassila, “The semantic web in scientific american,” 2001.
- [4] M. Fernández-López, A. Gómez-Pérez, and N. Juristo, “Methontology: From ontological art towards ontological engineering,” in *AAAI Conference on Artificial Intelligence*, 1997.
- [5] Y. Sure-Vetter, S. Staab, and R. Studer, “On-to-knowledge methodology (otkm),” in *Handbook on Ontologies*, 2004.
- [6] T. Tudorache, C. Nyulas, N. Noy, and M. A. Musen, “Webprotégé: A collaborative ontology editor and knowledge acquisition tool for the web,” *Semantic web*, vol. 4 1, pp. 89–99, 2013.
- [7] M. C. Suárez-Figueroa, A. Gómez-Pérez, and M. Fernández-López, “The neon methodology framework: A scenario-based methodology for ontology development,” *Appl. Ontology*, vol. 10, pp. 107–145, 2015.
- [8] J. J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities.,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79 8, pp. 2554–8, 1982.
- [9] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, pp. 1735–1780, 1997.
- [10] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *NIPS*, 2017.
- [11] M. Poveda-Villalón, A. Fernández-Izquierdo, M. Fernández-López, and R. García-Castro, “Lot: An industrial oriented ontology engineering framework,” *Eng. Appl. Artif. Intell.*, vol. 111, p. 104755, 2022.
- [12] M. Poveda, M. C. Suárez-Figueroa, and A. Gómez-Pérez, “Common pitfalls in ontology development,” in *Conferencia de la Asociación Española para la Inteligencia Artificial*, 2009.

- 
- [13] M. Shamsfard and A. A. Barforoush, "The state of the art in ontology learning: a framework for comparison," *The Knowledge Engineering Review*, vol. 18, pp. 293–316, 2003.
- [14] D. Faure and C. Nédellec, "Knowledge acquisition of predicate argument structures from technical texts using machine learning: The system asium," in *International Conference Knowledge Engineering and Knowledge Management*, 1999.
- [15] T. Morita, Y. Shigeta, N. Sugiura, N. Fukuta, N. Izumi, and T. Yamaguchi, "Doodle-owl: Owl-based semi-automatic ontology development environment," in *Workshop on Evaluation of Ontology-based Tools*, 2004.
- [16] M. Shamsfard and A. A. Barforoush, "Learning ontologies from natural language texts," *Int. J. Hum. Comput. Stud.*, vol. 60, pp. 17–63, 2004.
- [17] G. de Chalendar and B. Grau, "Svetlan' a system to classify nouns in context," 2000.
- [18] U. Hahn and M. Romacker, "Syndikate-generating text knowledge bases from natural language texts," *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*, vol. 5, pp. 918–923 vol.5, 1999.
- [19] A. Maedche and S. Staab, "The text-to-onto ontology learning environment," 2000.
- [20] M. W. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. M. Mitchell, K. Nigam, and S. Slattery, "Learning to extract symbolic knowledge from the world wide web," in *AAAI/IAAI*, 1998.
- [21] J.-L. Martínez-Rodríguez, A. Hogan, and I. Lopez-Arevalo, "Information extraction meets the semantic web: A survey," *Semantic Web*, vol. 11, pp. 255–335, 2020.
- [22] P. Buitelaar, P. Cimiano, and B. Magnini, "Ontology learning from text: An overview," 2005.
- [23] A. Kumar, A. Pandey, R. Gadia, and M. Mishra, "Building knowledge graph using pre-trained language model for learning entity-aware relationships," *2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON)*, pp. 310–315, 2020.
- [24] A. Bosselut, H. Rashkin, M. Sap, C. Malaviya, A. Celikyilmaz, and Y. Choi, "Comet: Commonsense transformers for automatic knowledge graph construction," in *Annual Meeting of the Association for Computational Linguistics*, 2019.
- [25] A. Radford and K. Narasimhan, "Improving language understanding by generative pre-training," 2018.
- [26] S. Hao, B. Tan, K. Tang, H. Zhang, E. P. Xing, and Z. Hu, "Bertnet: Harvesting knowledge graphs from pretrained language models," *ArXiv*, vol. abs/2206.14268, 2022.
- [27] Y. Zhu, X. Wang, J. Chen, S. Qiao, Y. Ou, Y. Yao, S. Deng, H. Chen, and N. Zhang, "Llms for knowledge graph construction and reasoning: Recent capabilities and future opportunities," *ArXiv*, vol. abs/2305.13168, 2023.



- [28] Z. Bi, J. Chen, Y. Jiang, F. Xiong, W. Guo, H. Chen, and N. Zhang, “Codekgc: Code language model for generative knowledge graph construction,” *ArXiv*, vol. abs/2304.09048, 2023.
- [29] P. West, C. Bhagavatula, J. Hessel, J. D. Hwang, L. Jiang, R. L. Bras, X. Lu, S. Welleck, and Y. Choi, “Symbolic knowledge distillation: from general language models to commonsense models,” in *North American Chapter of the Association for Computational Linguistics*, 2021.
- [30] F. Giunchiglia and I. Zaihrayeu, “Lightweight ontologies,” 2007.
- [31] M. Koho, R. Leal, E. Ikkala, M. Tamper, H. Rantala, and E. Hyvönen, “Building lightweight ontologies for faceted search with named entity recognition: Case warmemoirsampo,” in *TEXT2KG 2022 & MK 2022 First International Workshop on Knowledge Graph Generation From Text and First International Workshop on Modular Knowledge*, CEUR-WS. org, 2022.
- [32] J. Devlin, M.-W. Chang, K. Lee, and K. N. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2018.
- [33] F. Li, Z. Lin, M. Zhang, and D. Ji, “A span-based model for joint overlapped and discontinuous named entity recognition,” *ArXiv*, vol. abs/2106.14373, 2021.
- [34] H. Yan, T. Gui, J. Dai, Q. Guo, Z. Zhang, and X. Qiu, “A unified generative framework for various ner subtasks,” *ArXiv*, vol. abs/2106.01223, 2021.
- [35] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Proc. NIPS*, (Montreal, CA), 2014.
- [36] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, “BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, (Online), pp. 7871–7880, Association for Computational Linguistics, July 2020.
- [37] Y. He and B. Tang, “Setgner: General named entity recognition as entity set generation,” in *Conference on Empirical Methods in Natural Language Processing*, 2022.
- [38] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, “Biobert: a pre-trained biomedical language representation model for biomedical text mining,” *Bioinformatics*, vol. 36, pp. 1234 – 1240, 2019.
- [39] I. Beltagy, K. Lo, and A. Cohan, “Scibert: A pretrained language model for scientific text,” in *Conference on Empirical Methods in Natural Language Processing*, 2019.
- [40] L. B. Soares, N. FitzGerald, J. Ling, and T. Kwiatkowski, “Matching the blanks: Distributional similarity for relation learning,” in *Annual Meeting of the Association for Computational Linguistics*, 2019.
- [41] D. Lin and P. Pantel, “Dirt @sbt@discovery of inference rules from text,” in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, (New York, NY, USA), p. 323–328, Association for Computing Machinery, 2001.

- 
- [42] Y. Xu, M. Li, L. Cui, S. Huang, F. Wei, and M. Zhou, “Layoutlm: Pre-training of text and layout for document image understanding,” *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.
- [43] G. Bekoulis, J. Deleu, T. Demeester, and C. Develder, “Joint entity recognition and relation extraction as a multi-head selection problem,” *ArXiv*, vol. abs/1804.07847, 2018.
- [44] I. Yamada, A. Asai, H. Shindo, H. Takeda, and Y. Matsumoto, “Luke: Deep contextualized entity representations with entity-aware self-attention,” in *Conference on Empirical Methods in Natural Language Processing*, 2020.
- [45] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, *et al.*, “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.
- [46] G. Petrucci, C. Ghidini, and M. Rospocher, “Ontology learning in the deep,” in *International Conference Knowledge Engineering and Knowledge Management*, 2016.
- [47] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. J. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” *ArXiv*, vol. abs/2005.14165, 2020.
- [48] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, “Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing,” *ACM Computing Surveys*, vol. 55, pp. 1 – 35, 2021.
- [49] X. Han, H. Zhu, P. Yu, Z. Wang, Y. Yao, Z. Liu, and M. Sun, “Fewrel: A large-scale supervised few-shot relation classification dataset with state-of-the-art evaluation,” in *Conference on Empirical Methods in Natural Language Processing*, 2018.