Automatic Enrichment of Semantic 3D City Models using Large Language Models

Khaoula Kanna*, Thomas H. Kolbe*

*Chair of Geoinformatics, Technical University of Munich, 80333 Munich, Germany - (khaoula.kanna, thomas.kolbe)@tum.de

Keywords: 3D City Models, CityGML 3.0, Urban Digital Twins, Semantic Enrichment, 3DCityDB, Large Language Models

Abstract

Semantic 3D city models have become an essential component of city planning and digital twin applications. While standards like CityGML have enabled the structured representation of buildings and infrastructure, publicly available CityGML datasets often lack critical semantic attributes such as construction year, usage type, refurbishment status or sometimes outdated building function. These gaps hinder the application of 3D models in areas like energy demand analysis or infrastructure planning. Meanwhile, much of the missing data can be found in alternative sources such as municipal records, OpenStreetMap, or other APIs. Yet, integrating this heterogeneous and often unstructured information into the CityGML schema remains a complex task that requires geospatial expertise and good knowledge of the CityGML data model. In this paper, we explore the use of Large Language Models (LLMs) to automatically extract and map relevant information from sources like PDFs, APIs and VGI (Volunteered Geographic Information) platforms such as OpenStreetMap into CityGML, using spatial databases such as 3DCityDB to store and manage the enriched semantic data for both building and street use cases. We propose a framework based on two LLM agents, one for data enrichment and one for querying, which will enable non-experts to enrich and interact with 3D city models more effectively. Our approach aims to reduce reliance on domain-specific knowledge and make the usage of semantic 3D city models accessible to everyone.

1. Introduction

In recent years, semantic 3D city models have been widely used in the domains of smart city development and urban digital twins. Their adoption is largely driven by the increasing availability of these models across various cities and countries, often accessible via open access websites. This accessibility has made it easier for urban planners, architects, and researchers to employ 3D city models for a wide range of applications, including energy analysis, urban planning, and environmental simulations. One of the most frequently used standards for representing these models is CityGML (Gröger et al., 2012), a standard by the Open Geospatial Consortium (OGC) for representing and exchanging semantic 3D city models. CityGML enables the representation of detailed building attributes, transportation networks, and other city elements. Managing, analysing and handling CityGML datasets can be done using the 3DCityDB (Yao et al., 2018), an open-source software suite for spatial relational database management systems (SRDBMS) such as Oracle Spatial and PostGIS. Many cities worldwide use 3DCityDB for managing their semantic 3D city models. Compared to alternatives such as CJDB (a lightweight database based on CityJSON encoding (Powałka et al., 2024)), 3DCityDB offers several advantages in terms of semantic richness and spatial analysis capabilities. Since CJDB inherits the limitations of CityJSON, it lacks support for CityGML 3.0 features such as the Dynamizer module and new transportation classes (e.g., Section, Intersection). Moreover, although CJDB is built on PostGIS like 3DCityDB, it does not utilize 3D spatial datatypes, indexes, predicates or functions (e.g., for volume or surface area calculation) which are essential for advanced geospatial operations. Alternatively, web-based approaches such as the OGC FeatureAPI or WFS allow access to CityGML data without a database, but they do not support spatial aggregation or complex querying. In contrast, 3DCityDB provides robust spatial SQL support, including filtering, aggregation, and efficient data management for digital twin applications. Moreover, the new version 5 of the 3DCityDB was recently released (3DCityDB Steering Group, 2025), offering full support of CityGML 3.0 (Kutzner et al., 2020) with fewer tables, and facilitating complex GIS modelling and analysis tasks for smart city and digital twin applications.

Many applications rely on rich semantic information that is frequently absent in publicly available 3D city models. Attributes such as the year of construction, refurbishment status, and building usage are often missing or incomplete. In addition, many existing 3D city models are not regularly updated, which limits their accuracy and usability for dynamic urban analysis and decision-making. Interestingly, much of this missing information can be found in other sources, such as municipal documents, cadastral datasets, or volunteered geographic information (VGI) platforms like OpenStreetMap (OSM), WikiMapia, Wikimedia, or Mapillary. These sources offer huge amounts of data that could significantly enhance the semantic richness of city models. However, integrating this heterogeneous and sometimes unstructured information into standardized formats such as CityGML remains a complex and manual task and requires a deep understanding of the CityGML data model. Previous work in semantic enrichment of 3D city models have shown its potential for urban applications. While some studies have relied on manual integration (Kumke et al., 2007), others have automated this integration using techniques such as mapping algorithms ((Smart et al., 2011), (Ogawa et al., 2024)), Knowledge Graphs (Ding et al., 2024), or ontologies (Falquet et al., 2009). Commercial software such as Feature Manipulation Engine (FME) or ArcGIS have also been used for data integration. However, even these automated approaches require domain experts with a deep understanding of the CityGML data model to correctly map external data into the appropriate schema elements.

Meanwhile, recent advancements in Large Language Models (LLMs) have shown impressive capabilities in understanding, interpreting, and generating structured outputs from heregenous data, such as PDF documents and APIs. Moreover, these models have shown great ability in generating and debugging SQL queries easily, which reduces the need for specialized SQL programming knowledge (Hong et al., 2025). Thus, the use of LLMs with relational databases such as the 3DCityDB can be very promising. The LLM can learn the schema of the

3DCityDB together with the data model of CityGML 3.0 and automatically update the corresponding CityGML 3.0 attributes such as *dateOfConstruction* or *usage/function* of a building.

In this work, we explore the use of LLMs for the automated enrichment of CityGML with different data sources (such as PDF documents and APIs) using the 3DCityDB as a bridge. The goal is to answer the following research question: "How accurate can LLMs identify and map attributes from heterogeneous data sources into the 3DCityDB and eventually CityGML 3.0?". In our approach, we use pre-trained LLMs to extract building related information from heterogeneous data sources and determine where in the CityGML data model each attribute should be inserted. The proposed system reduces manual intervention, lowers the expertise barrier for working with semantic 3D city models (and CityGML), and opens new possibilities for dynamic and scalable semantic enrichment of city models.

2. Related Work

Over many years, considerable efforts have been made toward the creation of semantic 3D city models for digital twin applications, based on standards such as CityGML. Since its version 2.0, CityGML has grown popularity with researchers, city planners and companies, making it nowadays widely used around the world. Moreover, the major new version of CityGML 3.0 provides various new features, such as a new space concept, the support of time-dependent properties, the possibility to manage multiple versions of cities and an improved representation of traffic infrastructure (Kutzner et al., 2020).

Available and open-source CityGML datasets do often include few amounts of information, such as geometry, height, roof shape. While this information is relevant for some digital twin applications such as visualization, surface/volume calculations and solar potential analysis, complex applications such as heat demand estimation and traffic simulation always require more input data. To overcome this limitation, researchers have focused on enriching CityGML models with semantic information since its early years. In 2007, Kumke et al. (Kumke et al., 2007), have enriched a 3D city model with infrared textures to get information about the thermal surface together with factual data from municipal surveying office, and have extended the XML schema of CityGML. In 2011, Smart et al. (Smart et al., 2011) have shown how available open source Web information such as OpenStreetMap and Wikipedia can be used for automated enrichment of 3D city models using a fuzzy matching algorithm. In a more recent work (Ahmadian & Pahlavani, 2022), a formal concept analysis (FCA)-based methodology was proposed to integrate and align semantic tags from OpenStreetMap (OSM) with the *AbstractBuilding* concept in CityGML. The study demonstrated that key OSM attributes such as class, usage/function, and address can be semantically mapped to CityGML, highlighting the potential of VGI as a complementary source for enriching standardized city models. In 2024, a CityGML-KG framework was proposed in (Ding et al., 2024) to expose 3DCityDB data as a Knowledge Graph, enabling ontology-based querying and integration with external sources like OpenStreetMap. The approach supports expressive queries across heterogeneous geospatial datasets. However, the approach is limited to CityGML 2.0, struggles with complex geometries such as polyhedral surfaces and geometry collections polyhedral surfaces and geometry collections. Besides the semantic enrichment, a recent study in Japan focused on geometric enrichment of CityGML by matching building measurement data such as 2D polygons, aerial images, and 3D point clouds, to the Japanese national 3D city model (Ogawa et al., 2024). The matching was based primarily on geometric properties, achieving high coverage rates (e.g., 93.6% match from aerial images). The approach enabled the automatic enrichment of textured LOD1/LOD2 models for selected buildings using open data sources. However, the method faces limitations in handling misaligned polygons and relies heavily on coverage quality of the measurement method, particularly for 3D point cloud data.

In recent years, generative AI and in particular Large Language Models (LLMs) have gained significant attention in the geospatial domain due to their wide range of capabilities, including question answering, code and text generation, data analysis, and especially data enrichment. While a number of activities were focused on the use of LLMs for question answering in geospatial contexts such as ((Mooney et al., 2023),(Jiang & Yang, 2024),(Li & Ning, 2023)), their application for geospatial data enrichment remains relatively unexplored. One notable exception is the work by (Juhász et al., 2023), where GPT-3.5-turbo was used to suggest appropriate OpenStreetMap (OSM) tags for roads based on descriptions of Mapillary street-level photographs. The study demonstrated that combining LLMs with detailed context and prompt engineering significantly improves the accuracy of mapping suggestions. Beyond the geospatial domain, several studies have investigated LLMs for enriching structured data. For instance, in (Kasneci & Kasneci, 2024), the authors explored how embeddings from LLMs such as RoBERTa and GPT-2 can be used to enrich tabular data by generating additional features that improve the performance of machine learning models. Similarly, in the cultural heritage domain, (Mountantonakis et al., 2024) introduced GPToLODS+, a framework that combines ChatGPT with large-scale RDF Knowledge Graphs and entity recognition tools to support semantic enrichment, fact validation, and question answering over textual inputs. While these studies target domains outside of GIS, their methodologies demonstrate the strong potential of LLMs to semantically align and enrich structured datasets from heterogeneous and unstructured sources. To the best of our knowledge, no prior work has explored the use of LLMs for the automated enrichment of CityGML data to support urban digital twin applications.
In this paper, we explore the usage of LLMs for automated enrichment of CityGML 3.0 from heterogenous data sources, such as PDFs and OSM to support urban digital twin applications using the 3DCityDB geospatial database.

3. Methodology

This section presents the architecture of our framework. The system consists of two main components: A Data Enrichment Agent and a Query Agent, each implemented as an LLM-powered assistant capable of interacting with the 3DCityDB database.

3.1 The data enrichment agent:

The Data Enrichment Agent is responsible for identifying relevant building related attributes from various heterogeneously structured data sources and updating the 3DCityDB accordingly. The overall architecture is illustrated in Figure 1 and is composed of four main components: data sources, prompt/instructions, the LLM agent, and the output.

Data sources may include PDF documents, JSON or Word files, Volunteered Geographic Information (VGI) platforms such as OpenStreetMap and Mapillary, as well as external APIs like OpenWeather. Table 1 lists examples of data that can be retrieved from these sources.

The LLM deals with accessing each data source separately by using specific tools. On the one hand, to extract data from PDF or Word documents, we use the Retrieval-Augmented Generation or RAG for short. RAG is a framework for accessing external documents or raw data (Lewis et al., 2020). On the other hand, retrieving data from external APIs needs Function/Tool Calling, which allows LLMs to call functions and considering the response.

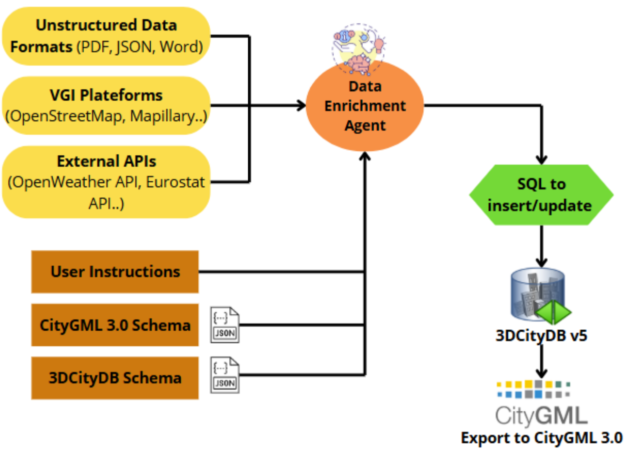


Figure 1. The Framework for the Data Enrichment agent

Data Source	Data Format	Data that can be retrieved
Building Energy certificate	PDF	Year of construction, Number of floors, Number of apartments, Energy efficiency class, Annual heating demand, CO2 emissions, Refurbishment measures, Recommendations for refurbishment
OpenStreetMap	JSON (API)	Building class, building function, Maximum road speed, Road material, Latitude, longitude, Building name, Extra tags (email, phone, website…)

Table 1. Example of different data sources and the data that can be retrieved from them.

It is also important to think about the mapping logic between the 3DCityDB where the CityGML dataset is stored, and the data sources. One way is to directly use the addresses of these buildings from the *ADDRESS* table of the 3DCityDB. Matching using the addresses may not always work, as the definition of a building footprint in CityGML data which is based on cadastre data may differ from other data sources such as OSM. However, in this paper, our use case includes buildings that match directly with the OSM buildings. In the case where addresses are missing, fall-back strategies such as bounding boxes or coordinate-based matching would have to be applied.

To enrich the database, the agent would have to understand both the CityGML 3.0 data model and the 3DCityDB v5 schema. This is achieved by embedding a concise but informative schema description in the prompt. Given the limited context length of LLMs, we focused on a subset of CityGML modules: Construction, Building and Transportation. These modules are sufficient for many urban applications but can be extended to include others, such as Bridge, Vegetation, or Dynamizer, based on the use case. The 3DCityDB schema was first exported from the database, then we enhanced the database table descriptions and added examples of typical SQL queries, creating a few-shot in-context learning environment. Furthermore, external documents, such as the German cadastral code list for building functions, can also be integrated using RAG to support semantic alignment. When applying this framework to other countries, for example Japan, Japanese code lists could be added instead.

Once this foundational knowledge is provided, the user interacts with the agent by uploading files or issuing requests. For instance, if the user uploads a folder containing energy certificates, the agent automatically reads the documents, identifies available attributes, and asks the user to confirm which ones should be inserted. Based on the CityGML 3.0 schema, the agent decides where the data belongs. For example, *dateOfConstruction* is an attribute of the Construction module and must be inserted into the *PROPERTY* table under the *con* namespace (ID = 8).
Before inserting a property, the agent must determine to which building the attribute belongs to. This is done by identifying the correct *feature_id*, which uniquely represents a building in the *FEATURE* table. The *feature_id* is retrieved by joining the *FEATURE*, *PROPERTY*, and *ADDRESS* tables. According to the 3DCityDB schema, addresses are stored as separate features and linked to buildings via the *PROPERTY* table, where the name is *'address'* and the address ID is stored in the *val_address_id* column. The LLM generates queries such as:

```
SELECT f.id AS feature_id
FROM feature f
JOIN property p ON f.id = p.feature_id
JOIN address a ON p.val_address_id = a.id
WHERE p.name = 'address'
  AND a.street = 'Hugo-Wolf-Straße'
  AND a.house_number = '68'
  AND f.objectclass_id = 901; -- 901 corresponds to Building
```

Once the correct *feature_id* is identified, the semantic attribute can be inserted using a query such as:

```
INSERT INTO property (feature_id, name, namespace_id, val_timestamp)
VALUES (12345, 'dateOfConstruction', 8, '1985-01-01');
```

Although a unique ID for each property should be specified here, PostgreSQL automatically assigns one if omitted, by incrementing the last existing *property_id* in the table.

Of course, the agent is not able to insert or update data on its own, it needs Function/Tool calling to do so. The first function the agent needs is the run_sql(query: str), with the corresponding SQL statement created by the agent to insert/update a specific attribute. The other functions the agent needs are for accessing the APIs, such as OSM. In these cases, the user must tell the agent for which objects the new attributes should be added. This could be for the entire dataset in the database or only for selected buildings or streets. The agent goes on and calls the function for the specific object, inserts or updates the attributes, before moving on to the next object in

the database. Moreover, other functions can also be added with other APIs depending on the use case.
Finally, the CityGML 3.0 data can be exported from the 3DCityDB and used for further data analysis or urban digital twin applications. An example could be the heat demand analysis or traffic simulation.

3.2 The Data Query agent

Beyond using the CityGML file in further data analyses, another user might want to retrieve some information directly from the 3DCityDB and do data aggregation with them. The Data Query agent is an agent that uses the understanding of the 3DCityDB schema and generates SQL queries based on the user request. Figure 2 shows the logic behind the query agent.
One might argue that the Data Enrichment Agent could also handle database queries, since it already possesses the necessary knowledge. However, this is very dangerous as the data enrichment agent has the right to read and write in the database, which should only be used by trusted administrators. In contrast, the data query agent is designed for broader accessibility, targeting users like citizens, researchers, and city planners who should have read-only access to the database to ensure data integrity.
Based on this, and similar to the data enrichment agent, the query agent will be provided a similar prompt with the description of the 3DCityDB. To perform its task, the agent is granted access to a single tool function: run_sql_query(query), which executes the SQL query generated from the user's request and returns the results in a human readable format.

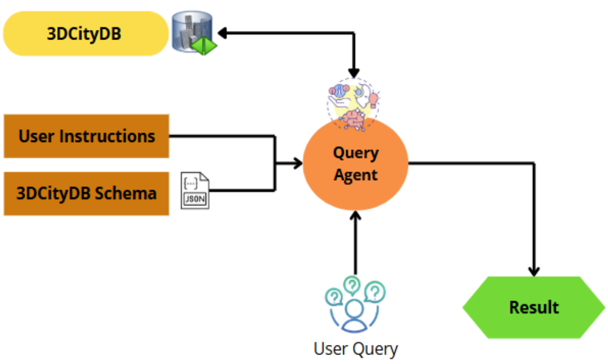
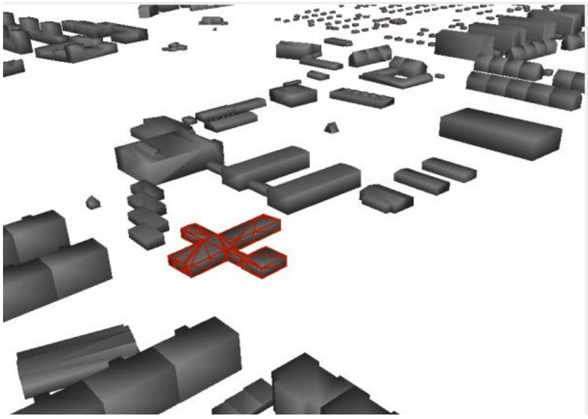


Figure 2. The Framework of the Query Agent

4. Implementation

To evaluate the feasibility of the proposed semantic enrichment framework, we implemented and tested it on two use cases: one for buildings and one for streets. For the buildings, we choose a district 3D city model. The building dataset used in this case study covers a district in Munich. The original data was downloaded from the Open Data Portal of Bavaria in CityGML 2.0 format and imported into the 3DCityDB v5. However, the dataset only includes basic geometric and topographic information such as footprints, roof shapes, and building heights and building function, but lacks crucial semantic attributes needed for urban analysis such as year of construction, number of floors and building usage. The building function refers to the intended purpose for which a building was originally constructed, whereas the building usage describes its current use. For example, an industrial building (function: industry) may currently be used for educational purposes (usage: education). Unfortunately, these two attributes are often confused. An example is shown in Figure 4, illustrating a kindergarten building. As indicated by the highlighted attributes, the initial building function provided by the mapping agency is 31001_9998, which corresponds to 'Not to be specified according to sources' in the German CityGML code list. However, since the building is currently used as a kindergarten, the appropriate building usage should be 31001_3065, as defined in the same code list. Moreover, attributes such as year of construction are also missing. To enrich and update the semantic content of this dataset, we used various data sources:
- Energy certificates, provided by the city of Munich for each building provided in PDF format, which contain information such as year of construction, number of apartments, energy class, annual heating demand, and CO_2 emissions.
- OpenStreetMap (OSM) as a Volunteered Geographic Information (VGI) source for attributes like building class, building function, and auxiliary tags (e.g., contact information or names).
Each of these sources was accessed via a corresponding tool-enabled function call. The agent processes the user's input (e.g., a folder path for PDFs or selected OSM tags) and uses the appropriate tool (RAG for PDFs, Function Calling for APIs) to extract and interpret the data. The assistant then generates a valid SQL INSERT or UPDATE query based on the CityGML 3.0 schema knowledge and the 3DCityDB table structure, as described in Section 3.
The LLM used in this work was GPT-4o (OpenAI et al., 2024). The interaction with the data enrichment agent follows a structured process, through a user-agent communication. The user initiates the enrichment by uploading a folder containing energy certificates and submits a request to update the CityGML model stored in the 3DCityDB. After receiving this input, the agent analyses the first document and gives the user a list of the attributes and information included in the document. The agent asks the user what kind of attributes are relevant to be added to the database. Upon the user’s response, the agent starts mapping the attributes to their corresponding namespaces in CityGML. For example, the year of construction should be added to the Construction namespace with the attribute *dateOfConstruction*, the number of storeys should be added to the Building namespace with the attribute *storeysAboveGround*. On the other hand, all the attributes that have no corresponding mapping in CityGML 3.0, such as CO2 emissions, yearly energy demand, should be added to the Generics namespace as generic attributes. The agent suggests the generic attributes names and asks the user if the names are okay as they are. The user can suggest other names if required. Finally, if the user is okay with the mapping of the agent, the agent proceeds with inserting the attributes in the 3DCityDB. Similarly, the OSM data was used to update the building functions in the 3DCityDB, as well as adding useful attributes such as contact data or a website. The agent uses the *ADDRESS* table in the 3DCityDB to retrieve all the building addresses, and sends a request to the OSM Nominatim API using the street name and house number for each single building. The agent analyses the JSON response and retrieves the important information to be inserted in the database. And as the agent is capable of understanding multiple languages, it can interpret the JSON file easily. As mentioned in Section 3, the agent is also provided with an XML code list for building functions and usages from the German cadastre. For example, the type “kindergarten” in the Nominatim data can be mapped to the usage code 31001_3065 according to the German code list. Moreover, additional attributes can also be added such as contact email, opening hours, and website as generic



Property	Value
Feature Type	Building
Coordinate System	EPSG:25832
Dimension	3D
Number of Vertices	436
Min Extents	690782.76, 5342401.21, 498.13
Max Extents	690829.92, 5342446.65, 502.798
Attributes (27)	
citygml_creationDate (string: UTF-16LE)	2021-03-18
citygml_feature_role (string: UTF-16LE)	cityObjectMember
citygml_function (string: UTF-16LE)	31001_9998
citygml_level_of_detail{0} (string: UTF-16LE)	2
citygml_measured_height (string: UTF-16LE)	4.664
citygml_measured_height_units (strin...	urn:adv:uom:m
citygml_roof_type (string: UTF-16LE)	1000
citygml_storeys_above_ground (strin...	1
citygml_target_uri (string: UTF-16LE)	http://www.opengis.net/citygml/building/1.0
DatenquelleBodenhoehe (string: UTF-16LE)	1100
DatenquelleDachhoehe (string: UTF-16LE)	5000
DatenquelleLage (string: UTF-16LE)	1000
externalReference{0}.externalObject.na...	DEBYvAAAAABS436P
externalReference{0}.informationSyste...	http://repository.gdi-de.org/schemas/adv/citygml/fdv/art.htm#_9100
fme_feature_type (string: UTF-8)	Building
fme_geometry (string: UTF-8)	fme_aggregate
fme_type (string: UTF-8)	fme_solid
Gemeindeschluessel (string: UTF-16LE)	09162000
Geometrietyp2DReferenz (string: UTF-16LE)	3000

Figure 3. The initial attributes of the kindergarten building as provided by the official mapping agency

attributes. After enriching the 3D city model, we are able to see the added attributes in the database. These attributes are highlighted in Figure 3.

To demonstrate the capabilities of the Enrichment Agent for transportation features, we selected a street in Numazu City, Shizuoka Prefecture, Japan, as a test case to evaluate the framework's applicability beyond European contexts. The street, known as 沼津港線 (Numazu-kō-sen), is modeled in CityGML 3.0 using the components *TrafficArea*, *TrafficSpace*, *Road*, *AuxiliaryTrafficSpace*, and *AuxiliaryTrafficArea*. An example of a *TrafficArea* and its associated attributes is shown in Figure 5. As illustrated, important attributes such as the maximum speed, road type, road material, and road name are not available in the original CityGML dataset. In contrast, this information can be retrieved from external data sources such as OSM using the Overpass API, which provides more detailed road data than Nominatim. OSM includes a wide range of road types, such as footway, highway, and cycleway, as well as intersections data. These intersections can also be queried via Overpass API and subsequently mapped to the corresponding intersections in the CityGML model. Figure 6 presents examples of both road and intersection data retrieved using Overpass QL. However, fetching precise Overpass API queries requires advanced knowledge of the Overpass QL language, which is not easy to learn. Additionally, the returned data is often unstructured and can include metadata in multiple languages, such as Japanese in this case. Fortunately, LLMs such as OpenAI's GPT-4o have demonstrated strong capabilities in understanding Overpass QL as well as processing multilingual data, which makes the GPT-4o model well-suited for this type of semantic enrichment task.
Given the type of data available in OpenStreetMap (OSM), we decided to semantically enrich both *TrafficArea* and *Intersection* elements in the CityGML model. To achieve this, the agent connects to the 3DCityDB and retrieves all *TrafficArea* features by filtering based on their *objectclass_id*, which corresponds to 613 in the 3DCityDB schema. Using this *feature_id*, the agent then queries the corresponding geometry from the *geometry_data* table and converts it into WGS84 coordinates (latitude and longitude) using the following SQL query:

```
SELECT ST_AsText(ST_Transform(ST_Envelope(geometry), 4326)) AS bbox_wgs84
FROM geometry_data
WHERE id = 734;
```

The agent extracts the geometry for each feature and formulates an Overpass API query to retrieve attributes related to the corresponding road segment. However, a limitation of the Overpass API is that it returns not only the attributes of the targeted road segment but also those of nearby roads. This is due to the lack of support for spatial predicates such as within, contains, or fully inside in Overpass QL. To overcome this limitation, a secondary filtering function was implemented using the function calling tool. The added function first filters Overpass results by element type (e.g., way for roads, node for intersections) and then uses the Python Shapely library to spatially match only those features that fall within the geometry retrieved from the 3DCityDB. If multiple matching candidates are found, the agent selects the first match for data enrichment. Finally, the agent constructs and executes the corresponding SQL INSERT queries to populate the new attributes after translating them to English into the database. An example of enriched attributes added to a *TrafficArea* is illustrated in Figure 7.

An example of the interaction with the Enrichment Agent is shown on the left of the Figure 8. Moreover, it is also possible to directly interact with this data using the Query Agent as shown on the right of the Figure 8. To make the interface user friendly, we used the Gradio Library with Python.

5. Evaluation

To evaluate the effectiveness of our proposed framework for semantic enrichment of CityGML 3.0 models using Large Language Models (LLMs), we conducted an evaluation focused

id	feature_id	datatype_	namespace_ic	name	val_int	val_double	val_string	val_timestamp	val_uri
56	16	15	1	externalReference	[NULL]	[NULL]	[NULL]	[NULL]	DEBYvAAAAABS5duV
57	16	5	3	GMLID	[NULL]	[NULL]	DEBY_LOD2_4967557	[NULL]	[NULL]
59	16	4	3	building_gross_floor_area	[NULL]	602	[NULL]	[NULL]	[NULL]
61	16	5	3	building_usage	[NULL]	[NULL]	Daycare	[NULL]	[NULL]
63	16	5	3	district_heating	[NULL]	[NULL]	not connected	[NULL]	[NULL]
66	16	5	3	ownership_type	[NULL]	[NULL]	Public	[NULL]	[NULL]
67	16	5	3	refurbishment_state	[NULL]	[NULL]	not refurbished	[NULL]	[NULL]
68	16	5	3	refurbishment_type	[NULL]	[NULL]	n/a - RBS	[NULL]	[NULL]
69	16	4	3	usable_and_living_area	[NULL]	476	[NULL]	[NULL]	[NULL]
71	16	4	3	yearly_ENEV	[NULL]	640.67	[NULL]	[NULL]	[NULL]
72	16	7	8	dateOfConstruction	[NULL]	[NULL]	[NULL]	1972-01-01 01:00:00.000 +0100	[NULL]
73	16	702	8	height	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
74	16	17	8	value	[NULL]	4.446	[NULL]	[NULL]	[NULL]
79	16	3	10	storeysAboveGround	1	[NULL]	[NULL]	[NULL]	[NULL]
138	16	8	10	address	[NULL]	[NULL]	[NULL]	[NULL]	[NULL]
78	16	14	10	function	[NULL]	[NULL]	31001_9998	[NULL]	[NULL]
19,912	16	5	3	contact_phone	[NULL]	[NULL]	+49 89 318884650	[NULL]	[NULL]
19,913	16	7	3	contact_website	[NULL]	[NULL]	[NULL]	[NULL]	https://stadt.muenchen.de/service/in
19,914	16	5	3	contact_email	[NULL]	[NULL]	kita.hugowolfstr.66@muen	[NULL]	[NULL]

Figure 4. The newly added attributes in the 3DCityDB V5 to the kindergarten building using the Enrichment Agent

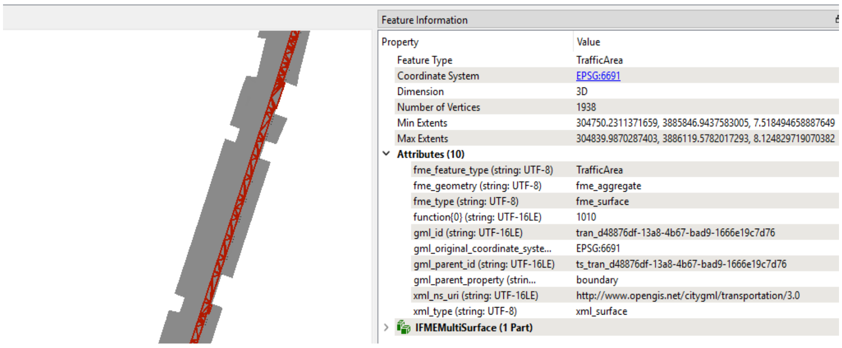


Figure 5. A TrafficArea and its attributes in the Numazu-kō-sen street modelled in CityGML 3.0

on two main aspects: (1) the correctness of the attribute extraction and naming, and (2) the semantic and structural accuracy of the generated SQL queries.

We selected a subset of energy certificates and OpenStreetMap (OSM) data as input for the data enrichment agent. Then, we manually created attribute names and SQL insert statements in order to compare the LLM-generated results with the manually added attributes for 20 buildings.

To assess the quality of the generated attribute names and SQL statements, we applied two widely recognized metrics from the natural language processing domain: BLEU (Bilingual Evaluation Understudy) and ROUGE-L (Recall-Oriented Understudy for Gisting Evaluation - Longest Common Subsequence variant). These metrics are commonly used to evaluate machine-generated text and are increasingly adopted in scientific literature on LLM-based tasks. In our use case, BLEU was used to measure the n-gram overlap (which is the number of matching n-tokens) between the generated and reference strings. We used BLEU-4 with smoothing to penalize overly short outputs and improve stability across examples. So, for example, if the expected SQL query for inserting attributes is:

Correct query: *INSERT INTO property (feature_id, name, namespace_id, val_timestamp) VALUES ...*

Agent-generated query: *INSERT INTO property (feature_id, name, namespace_id, val_date) VALUES ...*

Then the BLEU-4 metric will catch the *val_date* which is incorrect and should be *val_timestamp*, as there is no *val_date* column in the PROPERTY table, which will result in a lower score. BLEU-4 is particularly effective in penalizing small but critical mistakes. Because database queries are highly sensitive to syntax and schema-defined names, BLEU is an appropriate and strict metric for this purpose.

ROUGE-L, on the other hand, measures the semantic similarity by computing the Longest Common Subsequence (LCS) between the reference and the generated output. Unlike BLEU, it is recall-oriented and allows for non-contiguous matches, as long as word order is preserved. For example:

Correct query: *INSERT INTO property (feature_id, name, namespace_id, val_timestamp) VALUES (123, 'dateOfConstruction', 8, '1985-01-01');*

Agent-generated query: *INSERT INTO property (feature_id, name, namespace_id, val_timestamp) VALUES (123, 'constructionDate', 8, '1985-01-01');*

The LCS in this case contains 13 out of 16 tokens, resulting in ROUGE-L F1 Score of 0.81 for this specific example. This reflects that the generated query retains the correct structure and logic, while highlighting the semantic error in attribute naming. However, flexibility in CityGML attribute names such as *dateOfConstruction* and *storeysAboveGround* is not acceptable and would result in database errors or loss of information. Therefore, the ROUGE-L was only used with generic attributes that are not defined in the CityGML standard. This makes it well suited for evaluating attribute naming when predefined names are not strictly required, for example, when mapping *energy_demand_annual* to *annualEnergyDemand*. Table 2

Figure 6. Available attributes in OSM for a road segment (left) and an intersection (right) in the Numazu-ko-sen Street.

id	feature_id	parent_id	datatype_id	namespace_id	name	val_int	val_double	val_string
879	354	[NULL]	14	7	function	[NULL]	[NULL]	1010
880	354	[NULL]	11	1	lod3MultiSurface	[NULL]	[NULL]	[NULL]
2,005	354	[NULL]	5	3	highwayClass	[NULL]	[NULL]	secondary
2,006	354	[NULL]	5	3	maxSpeed	[NULL]	[NULL]	50
2,007	354	[NULL]	5	3	name	[NULL]	[NULL]	Baraki–Numazu Line
2,008	354	[NULL]	5	3	oneWay	[NULL]	[NULL]	yes
2,009	354	[NULL]	5	3	ref	[NULL]	[NULL]	139
2,010	354	[NULL]	5	3	sourceName	[NULL]	[NULL]	YahooJapan/ALPSM.

Figure 7. The attributes integrated into the 3DCityDB for buildings on Numazu-ko-sen Street based on OSM data.

shows the results of the evaluation using BLEU-4 and ROUGE-L F1 scores for 20 buildings with energy certificates, while Table 3 shows the evaluation of the results with OSM data.

In addition to these quantitative results, a qualitative manual inspection confirmed that the LLM could reliably distinguish between standard CityGML 3.0 attributes and those that should be stored in the Generics module. Furthermore, the agent correctly identified the corresponding *namespace_id* and *datatype* for insertion, which showed its good ‘understanding’ of the CityGML 3.0 data model and 3DCityDB schema.

Metric	Predefined Attributes	Generic Attributes	SQL Queries
BLEU-4	0.82	0.89	0.81
ROUGE-L F1	-	0.90	0.92

Table 2. Results of the evaluation of the framework for the 20 buildings with energy certificates

Metric	Predefined Attributes	Generic Attributes	SQL Queries
BLEU-4	0.78	0.81	0.83
ROUGE-L F1	-	0.85	0.87

Table 3.Results of the evaluation of the framework for the 20 buildings with OSM data

6. Discussion

The evaluation results demonstrate that our proposed framework for semantic enrichment of CityGML 3.0 models using LLMs is both technically feasible and promising in terms of accuracy and usability. The high BLEU and ROUGE-L scores reflect the agent's ability to correctly identify relevant attributes from unstructured data sources and generate well-structured SQL statements aligned with the 3DCityDB schema to insert or update the database. BLEU-4 proved particularly effective in detecting syntax-level and naming inconsistencies in the generated SQL queries, especially for predefined attributes like *dateOfConstruction* and *storeysAboveGround*, where any naming deviation would lead to invalid queries. ROUGE-L, on the other hand, was useful for evaluating structural similarity in cases involving more flexible attribute names, such as generic properties (e.g., "*annualHeatingDemand*" vs. "*heating_energy*"), which do not follow strict schema constraints. However, our evaluation also showed some limitations. First, the ROUGE-L metric may overestimate similarity in cases where attribute names are semantically incorrect but structurally similar. For example, if the agent proposes *constructionDate* instead of the required *dateOfConstruction*, the LCS-based evaluation may still yield a high score despite the syntactic error. While this may not be a problem for natural language applications, in our case, it is critical. This highlights the need for stricter evaluation criteria for predefined attributes and maybe consider adding a more detailed CityGML data model description using RAG, as our description can be considered short. These errors could also be reduced using the communication with the agent where users approve or rename attributes before insertion.
Overall, the architecture which combines retrieval-augmented generation (RAG) for document understanding with function-calling tools for API access enables a flexible and extensible approach that can adapt to various differently structured and unstructured data sources. The approach shows that LLMs can significantly simplify and accelerate the semantic enrichment of 3D city models when they are equipped with a good comprehension of the data model and database schema. However, this comprehension should also be evaluated and tested with different scenarios. What would happen if we add a layer of additional information to the prompt? How good does the LLM behave when using few-shot in context learning versus zero-shot in context learning? These are essential questions that should be investigated in the future and which we believe are of a great importance.

7. Conclusion and Outlook

In this paper, we proposed, implemented and evaluated a framework for the semantic enrichment of CityGML 3.0 models using LLMs. The framework relies on LLM agents equipped

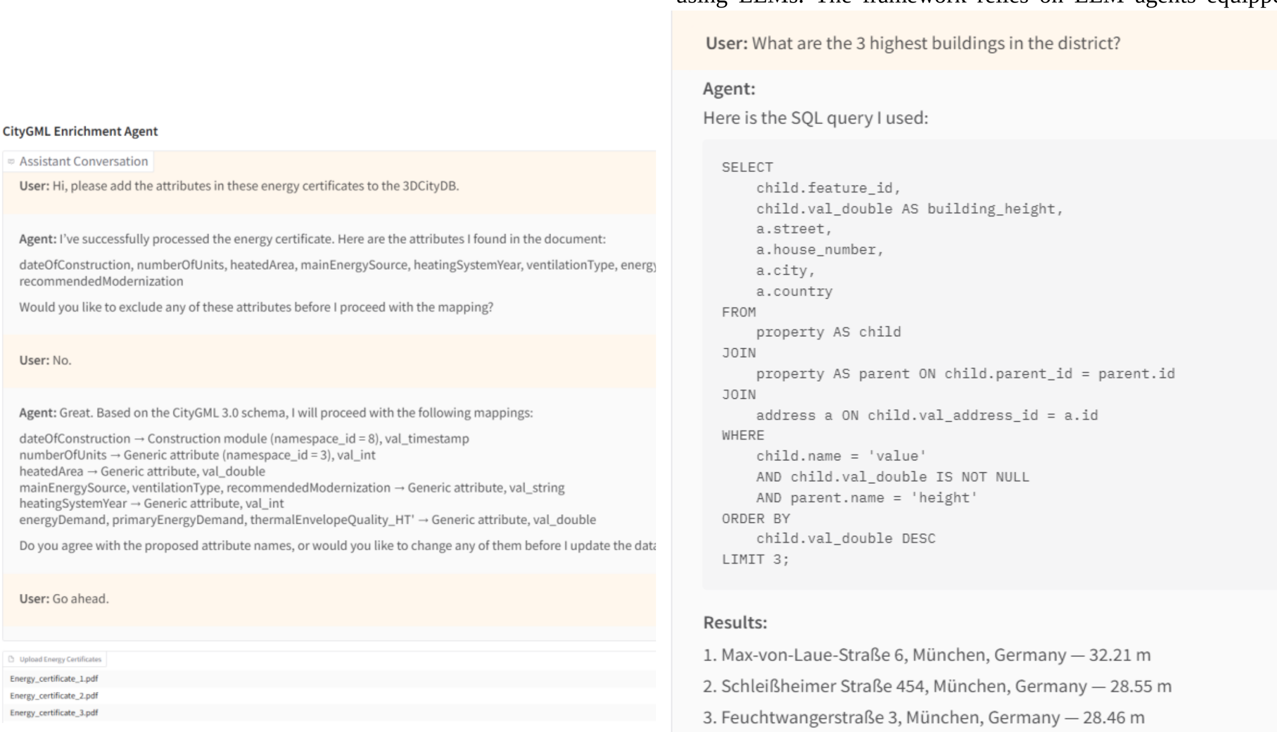


Figure 8. On the left: Interaction between the Enrichment Agent and the user / On the right: Interaction between user and Query agent

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-4-W6-2025-105-2025 | © Author(s) 2025. CC BY 4.0 License.

with RAG and function calling to automatically extract and integrate relevant attributes from various heterogenous data sources, including PDF documents and OpenStreetMap which includes more up-to-date data. The enriched data is inserted into a 3DCityDB v5 database following the CityGML 3.0 data model for both buildings and street models.

Through our use case we demonstrated that the LLM-based agent can better comprehend CityGML concepts, extract the relevant information from the heterogenous data sources, and generate correct SQL queries to insert them into the database, while interacting with users to control its behaviour. The evaluation using BLEU-4 and ROUGE-L metrics confirmed that the generated attribute names and SQL statements were syntactically and semantically aligned with human-written references.

Despite the promising results, several challenges remain. The framework assumes clean, machine-readable and trustful input data such as the energy certificates. Future work should address the integration of noisy or scanned documents through OCR and error-tolerant extraction techniques, as well as the use cases where the input data is wrong or contains errors. Moreover, while this study focused on buildings and transportation, the approach can be extended to other CityGML modules such as Vegetation, and Bridges. Incorporating dynamic data sources through the Dynamizer module, for instance using real-time sensor values from SensorThings API, could also offer exciting potential for real-time digital twin applications. Finally, feedback from city planners and GIS experts could help assess the accuracy of the results and enhance the framework. This would help identify not only technical gaps but also interface user-friendliness, privacy, and trust-related issues that must be addressed.

References

3DCityDB Steering Group. (2025). *3D City Database v5 documentation*. https://3dcitydb.github.io/3dcitydb-mkdocs/3dcitydb/

Ahmadian, S., & Pahlavani, P. (2022). Semantic integration of OpenStreetMap and CityGML with formal concept analysis. *Transactions in GIS*, *26*(8), 3349–3373.

Ding, L., Xiao, G., Pano, A., Fumagalli, M., Chen, D., Feng, Y., Calvanese, D., Fan, H., & Meng, L. (2024). Integrating 3D city data through knowledge graphs. *Geo-Spatial Information Science*, 1–20.

Falquet, G., Metral, C., & Cutting-Decelle, A.-F. (2009). Towards semantically enriched 3d city models: An ontology-based approach. *Proceeding GeoWeb*, 40–45.

Gröger, G., Kolbe, T. H., Nagel, C., & Haefele, K.-H. (2012). *OGC City Geography Markup Language (CityGML) Encoding Standard* (pp. 12–019). Open Geospatial Consortium.

Hong, Z., Yuan, Z., Zhang, Q., Chen, H., Dong, J., Huang, F., & Huang, X. (2025). *Next-Generation Database Interfaces: A Survey of LLM-based Text-to-SQL*. https://doi.org/10.48550/arXiv.2406.08426

Jiang, Y., & Yang, C. (2024). Is ChatGPT a Good Geospatial Data Analyst? Exploring the Integration of Natural Language into Structured Query Language within a Spatial Database. *ISPRS International Journal of Geo-Information*, *13*(1), 26.

Juhász, L., Mooney, P., Hochmair, H. H., & Guan, B. (2023). ChatGPT as a mapping assistant: A novel method to enrich maps with generative AI and content derived from street-level photographs. *Spatial Data Science Symposium 2023*.

Kasneci, G., & Kasneci, E. (2024). *Enriching Tabular Data with Contextual LLM Embeddings: A Comprehensive Ablation Study for Ensemble Classifiers*. https://doi.org/10.48550/arXiv.2411.01645

Kumke, H., Hoegner, L., Meng, L., & Stilla, U. (2007). Visualization of building models and factual data integrated by CityGML. *International Conference on Communications*.

Kutzner, T., Chaturvedi, K., & Kolbe, T. H. (2020). CityGML 3.0: New Functions Open Up New Applications. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, *88*(1), 43–61.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., Kiela, D. (2020). RetrievalAugmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems, 33

Li, Z., Ning, H. (2023). Autonomous GIS: The next-generation AI-powered GIS. International Journal of Digital Earth, 16(2).

Mooney, P., Cui, W., Guan, B., Juhász, L. (2023). Towards Understanding the Geospatial Skills of ChatGPT: Taking a Geographic Information Systems (GIS) Exam. Proceedings of the 6th ACM SIGSPATIAL International Workshop on AI for Geographic Knowledge Discovery.

Mountantonakis, M., Koumakis, M., Tzitzikas, Y. (2024). Combining LLMs and Hundreds of Knowledge Graphs for Data Enrichment, Validation and Integration Case Study: Cultural Heritage Domain. Proceedings of the International Conference on Museum Big Data (MBD 2024), November 18–19, Greece.

Ogawa, Y., Sato, G., Sekimoto, Y. (2024). Geometric-based approach for linking various building measurement data to a 3D city model. PLOS ONE, 19(1).

Hurst, A., Lerer, A., Goucher, A. P., Perelman, A., Ramesh, A., Clark, A., ..., Kivlichan, I. (2024). Gpt-4o system card. https://arxiv.org/abs/2410.21276

Powałka, L., Poon, C., Xia, Y., Meines, S., Yan, L., Cai, Y., Stavropoulou, G., Dukai, B., & Ledoux, H. (2024). cjdb: A Simple, Fast, and Lean Database Solution for the CityGML Data Model. *Recent Advances in 3D Geoinformation Science* (pp. 781–796).

Smart, P. D., Quinn, J. A., Jones, C. B. (2011). City model enrichment. ISPRS Journal of Photogrammetry and Remote Sensing, 66(2).

Yao, Z., Nagel, C., Kunde, F., Hudra, G., Willkomm, P., Donaubauer, A., Adolphi, T., Kolbe, T. H. (2018). 3DCityDB - a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML. Open Geospatial Data, Software and Standards, 3(1).