# Knowledge Graph-Guided Entity Linking with LLMs for the Identification of Construction Cost Items

**Renato Ramos**[1], **Renato Fileto**[1], **Diego Calvetti**[2]

[1]Depart. of Informatics and Statistics
Federal University of Santa Catarina (UFSC) – Florianópolis, Brazil

[2]School of Civil Construction, Faculty of Engineering
Pontificia Universidad Católica de Chile – Santiago, Chile

r.ramos.l.s@posgrad.ufsc.br, r.fileto@ufsc.br, diego.calvetti@uc.cl

***Abstract.*** *Linking construction service descriptions to cost catalog items requires technical compatibility, not only textual similarity. This paper frames the task as catalog-constrained entity linking, using a Knowledge Graph (KG) derived from SINAPI, the Brazilian National System for Construction Costs and Indexes. The proposed pipeline applies this KG to constrain decisions and retrieve compatible items that are then disambiguated by a Large Language Model (LLM). On 354 annotated descriptions from three public-building projects, the pipeline improves final Hit@3 by 10.73–32.77 percentage points over a hybrid retrieval-and-disambiguation baseline across model and candidate-list configurations, reaching 96.61% in the best configuration.*

## 1. Introduction

Construction cost estimation, procurement, and planning workflows often summarize services, materials, quantities, and technical requirements in spreadsheets and service lists [Akanbi and Zhang 2021, Sousa et al. 2025]. When standardized cost systems are used, these descriptions must be linked to catalog items so that estimates remain traceable, comparable, and compatible with the adopted cost reference. Incorrect links may propagate to cost estimates, reduce auditability, and compromise compliance with the adopted reference system. This task is therefore not only a text classification problem: the selected item must also satisfy technical scope and catalog constraints [Deza et al. 2022].

In Brazil, this issue is especially relevant because public-work estimates commonly rely on SINAPI, a national cost-reference system jointly produced by the Brazilian Institute of Geography and Statistics (IBGE) and CAIXA [Brasil 2021, IBGE 2026, CAIXA 2026]. SINAPI items encode technical composition variants, not only prices or textual labels. In the service families analyzed here, these variants differ by component type, material, dimension, execution method, application location, and included inputs. Thus, linking a project description to a SINAPI item requires checking technical constraints, not only lexical or semantic similarity.

Prior natural language processing (NLP) and LLM-based work in construction has addressed information extraction, classification, and cost-description processing [Ding et al. 2022, Shamshiri et al. 2024, Gatto et al. 2025]. However, these capabilities alone do not guarantee that a selected item complies with the technical constraints

of a standardized cost catalog. This limitation motivates treating cost item identification as an entity linking problem, in which construction service descriptions must be associated with entities in a reference catalog [Shen et al. 2015, Sevgili et al. 2022]. Ontologies and knowledge graphs (KGs) have been used in architecture, engineering, and construction (AEC) to integrate heterogeneous information and support semantic queries [Rasmussen et al. 2019, Farghaly et al. 2024]. However, existing approaches do not explicitly model catalog-specific technical constraints as graph-retrievable restrictions for LLM-based linking to standardized construction cost items.

To address this gap, this paper presents a staged LLM+KG pipeline that combines LLM-based interpretation with a knowledge graph derived from SINAPI technical documentation. The experimental scope covers cold-water supply, sanitary sewage, electrical installation, electrification, and external lighting services. The LLM-based stages perform language-dependent interpretation, while the KG-based stages constrain the decision space through classes, functional groups, technical documents, components, attributes, valid values, and graph-retrieved candidates.

This design leads to the following research question: *To what extent does a staged LLM+KG pipeline improve the linking of project service descriptions to standardized SINAPI cost items, compared with a dense–sparse retrieval baseline followed by LLM disambiguation?* The evaluation considers both final linking performance and the auditability of intermediate decisions.

In an experimental evaluation over 354 annotated construction service descriptions and 3,174 SINAPI cost items, the proposed pipeline improves final Hit@3 by 10.73 to 32.77 percentage points over the baseline across the evaluated model–(k) configurations. Stage-wise diagnostics show that most remaining errors are associated with missing or mismatched attributes, candidate-generation failures after earlier-stage mistakes, and residual ambiguity among technically similar SINAPI services.

The main contributions are: (i) a SINAPI knowledge graph representing hydrosanitary and electrical installation classes, technical documents, components, attributes, valid values, and services, constructed using an ontology engineering methodology [Suárez-Figueroa et al. 2015]; (ii) an LLM+KG pipeline for linking construction service descriptions to SINAPI cost items through staged semantic interpretation, graph-constrained candidate retrieval, and LLM disambiguation; and (iii) an experimental evaluation and stage-wise error analysis over annotated service descriptions and SINAPI cost items, comparing the proposed pipeline with a dense–sparse retrieval-and-disambiguation baseline.

## 2. Related Work

NLP and text mining have been applied to construction documents for document management, compliance checking, information extraction, and digital construction workflows [Ding et al. 2022, Shamshiri et al. 2024]. In cost-related workflows, prior studies have extracted design information from specifications [Akanbi and Zhang 2021], classified bill-of-quantities or cost descriptions [Deza et al. 2022, Gatto et al. 2025], and built Portuguese BoQ datasets with LLM-based augmentation [Sousa et al. 2025]. These works show the potential of NLP and LLMs for construction cost text. However, in these settings, the output is typically modeled as a category, an extracted field, or a re-

trieval/matching result, rather than as a catalog-constrained entity linking decision.

The task addressed in this paper is related to entity linking, in which textual mentions are associated with entities in a reference catalog or knowledge base [Shen et al. 2015, Sevgili et al. 2022]. A common formulation separates candidate generation from ranking or disambiguation; in this paper, the baseline follows this pattern through dense–sparse retrieval followed by LLM-based disambiguation. Here, the target entities are standardized SINAPI cost items whose validity depends on technical constraints such as component, material, dimension, execution method, application location, and included input items [CAIXA 2026]. Therefore, linking a project description to a SINAPI item requires candidates that are both semantically plausible and technically compatible.

Ontologies and knowledge graphs have also been investigated in AEC as mechanisms for representing concepts, relations, and constraints in machine-interpretable form [Rasmussen et al. 2019, Farghaly et al. 2024]. Recent studies have combined KGs, retrieval mechanisms, and LLMs for asset information, IFC querying, and product matching [Boscariol et al. 2024, Iranmanesh et al. 2025, Stellmacher et al. 2025]. These studies show the usefulness of semantic representations for querying, integration, extraction, and matching tasks. However, they do not explicitly use a catalog-derived KG to restrict intermediate decisions in a staged entity-linking pipeline for standardized construction cost items.

This paper differs from these lines of work by using a SINAPI-derived KG not only as a semantic representation, but as a constraint layer for staged entity linking. The graph restricts intermediate decisions and candidate retrieval using catalog-specific classes, technical documents, components, attributes, and valid values, while preserving intermediate outputs for stage-wise error diagnosis.

## 3. Proposed Approach

### 3.1. Overview and Problem Formulation

The proposed approach links construction service descriptions to standardized SINAPI service items through a staged LLM+KG pipeline. Given a description $d_i$, the objective is to predict a ranked final set $\hat{Y}_i \subseteq Y$, where $Y$ is the set of services represented in the SINAPI knowledge graph. The LLM cannot generate arbitrary SINAPI codes; instead, it makes intermediate decisions that progressively restrict the catalog search space before final disambiguation. Figure 1 summarizes the two connected layers: a SINAPI knowledge graph and a staged linking pipeline.

### 3.2. SINAPI Knowledge Graph Construction

The method relies on a domain knowledge graph derived from SINAPI technical documentation. Rather than representing SINAPI as a flat catalog of codes, the graph encodes the decision structure used by the pipeline and constrains the admissible paths that lead to each service item. The evaluated graph covers hydrosanitary installations, including cold-water supply and sanitary sewage systems, and electrical installations, including building electrification and external lighting. This scope comprises 3,174 SINAPI services, 35 technical documentation units, 307 normalized component groups, 133 attribute types, and 1,227 attribute values.

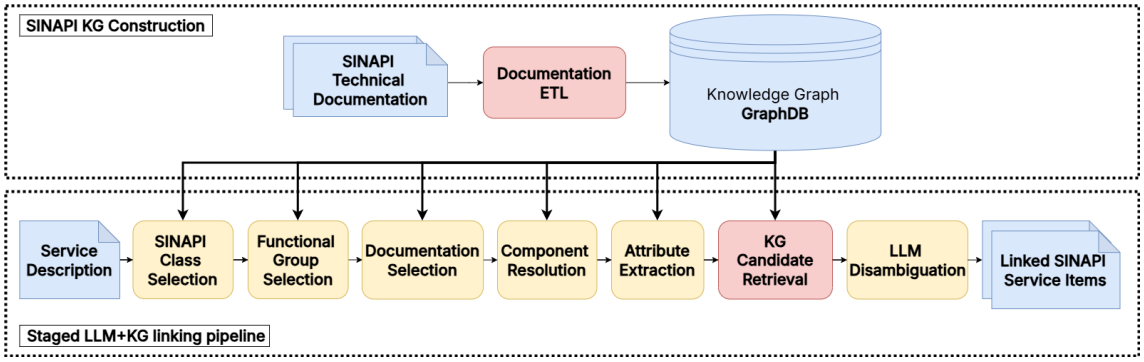



**Figure 1. Overview of the proposed LLM+KG pipeline.**

Graph construction was treated as a re-engineering process that transforms a non-ontological technical resource into an ontological representation, following NeOn ontology-engineering principles [Suárez-Figueroa et al. 2015]. The ontology defines the decision hierarchy, while the KG instantiates SINAPI entities and relations as RDF resources. This design allows each intermediate LLM prediction to be checked against explicit graph constraints before a final service item is selected.

During inference, the graph acts as a symbolic constraint layer. After the LLM predicts the intermediate decisions for a service-item description, the graph expands normalized component groups to their corresponding original SINAPI components and retrieves only service items available in compatible documentation branches and attribute-value configurations. KG Candidate Retrieval is implemented over the materialized RDF graph in GraphDB using SPARQL.

### 3.3. Staged LLM+KG Linking Pipeline

The linking pipeline separates language-dependent decisions, produced by the LLM as JSON outputs, from deterministic operations executed over the KG. At each LLM-based stage, the prompt contains the service description and admissible labels retrieved from the graph; the LLM must return only labels copied from that set. KG-based stages validate selections, expand normalized components to original SINAPI components, and retrieve service items using SPARQL. As a consequence, SINAPI service codes can be selected only during LLM Disambiguation and only from the retrieved candidate list.

Each input instance corresponds to one row in a project service list. Descriptive columns, such as section, subsection, item, and description, are concatenated to form $d_i$; quantity, unit, and source metadata are stored for provenance but not used as linking evidence. As a running example, consider `PVC ACESSÓRIOS – PVC ESGOTO – TUBO PVC PONTA-BOLSA C/ VIROLA – 100 MM – 4"`. The pipeline processes $d_i$ through SINAPI Class Selection, Functional Group Selection, Documentation Selection, Component Resolution, Attribute Extraction, KG Candidate Retrieval, and LLM Disambiguation.

First, Class, Functional Group, and Documentation Selection restrict downstream retrieval branches. In the example, `PVC ESGOTO` and `TUBO` support selecting `INHI – INSTALAÇÕES HIDROSSANITÁRIAS`, the functional group `INHI – Esgoto sanitário, águas pluviais,`

caixas enterradas e tratamento local, and the documentation unit SINAPI-CT-INSTALACOES-PREDIAIS-DE-ESGOTO-TUBOS-E-CONEXOES. Class Selection is single-label because the evaluated SINAPI classes are broad and mutually exclusive, while multiple compatible functional groups or documentation units may be kept to avoid discarding the reference path too early.

Second, Component Resolution selects normalized component groups associated with the selected documentation units. In the example, TUBO PVC PONTA-BOLSA C/ VIROLA is resolved to TUBO. Each valid pair of documentation unit and normalized component defines a component branch; if multiple branches remain compatible with $d_i$, they are processed independently.

Third, Attribute Extraction selects attribute–value pairs from branch-specific constraints retrieved from the KG. In the example, 100 MM is mapped to DIMENSAO_NOMINAL=100 MM; empty, null, non-applicable, or non-listed values are discarded, so candidate retrieval receives only graph-represented constraints.

Fourth, KG Candidate Retrieval queries the RDF graph using the selected documentation unit, expanded original SINAPI components, and extracted attributes. In the example, TUBO expands to original SINAPI components for normal-series PVC tubes installed in ramal, prumada, or subcoletor aéreo; retrieval then returns services whose RDF attributes include nominal dimension 100 MM. Candidates are scored by exact attribute matches, merged across branches, deduplicated by SINAPI code, ranked, and truncated to the candidate-list size $k$.

Finally, LLM Disambiguation receives the retrieved candidates and returns at most $K$ SINAPI service items. Multiple outputs are allowed because some descriptions omit catalog distinctions needed for a unique code. In the example, the description does not specify whether the tube is installed in a ramal, prumada, or subcoletor aéreo; therefore, the reference set contains technically compatible alternatives, such as 89714 (TUBO PVC, SERIE NORMAL, ESGOTO PREDIAL, DN 100 MM, FORNECIDO E INSTALADO EM RAMAL DE DESCARGA OU RAMAL DE ESGOTO SANITÁRIO. AF_08/2022), plus 89800 for prumada de esgoto sanitário or ventilação and 89848 for subcoletor aéreo de esgoto sanitário. The LLM must copy each selected code and description from the candidate list. The stored trace contains intermediate hypotheses, extracted attributes, retrieved candidates, and final selected services, allowing each error to be assigned to the first stage where all reference-compatible options are discarded.

## 4. Experiments

### 4.1. Dataset, Gold Standard, and Evaluation Design

The experiments evaluate whether the proposed staged LLM+KG pipeline improves spreadsheet-to-SINAPI linking compared with a hybrid retrieval-and-disambiguation baseline. Both approaches use the same input descriptions, SINAPI service set, installation scope, candidate-list size $k$, and LLM Disambiguation protocol. The main difference is candidate generation: the proposed pipeline applies graph-constrained intermediate decisions before retrieval, whereas the baseline retrieves candidates directly from embedded SINAPI service descriptions. The gold standard is used only after prediction generation, for evaluation.

The dataset was built from publicly available reference projects for Brazilian public facilities: the Novo PAC Saúde *Unidade Básica de Saúde (UBS) Porte I*, the FNDE *Creche Pré-Escola Tipo 1*, and the Ministry of Culture *CEU da Cultura – Núcleo Básico*. The selected spreadsheets cover hydrosanitary material lists, including cold-water supply and sanitary sewage systems, and electrical installation material lists. The extractor generated 813 service descriptions from seven spreadsheet sources; after exact-duplicate consolidation, while preserving project provenance, the dataset contained 683 unique descriptions.

Rows outside the evaluated SINAPI scope, lacking sufficient technical evidence, representing administrative or generic rows, or without compatible SINAPI items in the evaluated graph subset were excluded from the benchmark. The reported metrics are therefore restricted to 354 manually annotated service descriptions and 332 distinct valid target SINAPI codes.

The gold standard was manually curated from the deduplicated descriptions. Each entry records one SINAPI class; acceptable functional groups, technical documentation units, and normalized component groups when multiple compatible paths are supported; relevant attribute–value pairs; and one or more valid target SINAPI codes. Automated retrieval outputs were used only as lookup aids during annotation, not as ground-truth labels, and each accepted code was verified against the SINAPI catalog information represented in the KG.

The annotation policy allows multiple acceptable functional groups, documentation units, normalized component groups, or SINAPI codes when the source description does not provide enough evidence to distinguish between technically compatible alternatives. A prediction is considered correct if the selected SINAPI code belongs to the valid target set defined for that description.

### 4.2. Compared Methods

The experiments compare two linking strategies. The proposed method follows the staged LLM+KG pipeline described in Section 3.3. LLM-based stages produce structured JSON outputs, while graph-based stages provide closed option sets, validate selections, and retrieve service candidates that satisfy the structured constraints. LLM Disambiguation may select at most $K$ final SINAPI service items by copying codes and descriptions from the retrieved candidate list.

The hybrid retrieval baseline uses the same inputs, SINAPI service set, installation scope, candidate-list sizes, and LLM Disambiguation protocol, but replaces staged symbolic interpretation with direct retrieval. It retrieves semantically and lexically similar SINAPI services and exposes service metadata during LLM Disambiguation, without enforcing graph-based compatibility constraints before retrieval.

In the offline indexing phase, the SINAPI KG is used only as a source of catalog records. Each indexed document represents one service item: the official SINAPI description is indexed as text, while class, functional group, documentation unit, component, attributes, and composition items are stored as metadata for disambiguation prompts.

At inference time, each spreadsheet-derived description is submitted directly to Qdrant using dense embeddings and BM25-based sparse vectors

[Robertson and Zaragoza 2009, Qdrant 2026]. Dense embeddings capture semantic similarity, while sparse vectors preserve lexical evidence such as material, diameter, voltage, installation type, and component name. Dense vectors are generated with `qwen3-embedding:0.6b`, while sparse vectors use `Qdrant/bm25` through FastEmbed.

The top-$k$ retrieved SINAPI services, with $k \in \{5, 10, 15\}$, are passed to the same LLM Disambiguation stage used by the proposed method. The LLM receives the input description, retrieved candidates, and candidate metadata. It may copy at most $K$ codes and descriptions from the list or return an empty selection. Thus, unlike the proposed pipeline, the baseline generates candidates from dense–sparse (hybrid) similarity over service descriptions rather than from staged graph constraints over SINAPI classes, documentation units, components, and attributes.

### 4.3. Implementation and Model Configuration

The reported protocol evaluates two linking strategies, two LLMs, and three candidate-list sizes $k \in \{5, 10, 15\}$. Each run stores input metadata, project provenance, model and retrieval configurations, pipeline outputs, token usage, execution time, and gold-standard annotations for evaluation. The candidate-list size $k$ is global: after merging and ranking, both approaches pass at most $k$ candidates to LLM Disambiguation. In the proposed pipeline, Functional Group Selection, Documentation Selection, and Component Resolution are also limited to at most three intermediate labels, preserving multiple compatible routes when the description is underspecified. The output limit $K$, defined as the maximum number of final service items that LLM Disambiguation may return, is also applied equally to both approaches.

For the proposed pipeline, stages before KG Candidate Retrieval are executed once per model and input description, because they do not depend on $k$. The implementation then reruns only KG Candidate Retrieval and LLM Disambiguation for each $k$. The proposed pipeline uses GraphDB and SPARQL, while the baseline uses Qdrant over embedded SINAPI service descriptions.

All LLM calls use temperature 0. Structured outputs are used when supported by the selected model and provider. Invalid structured outputs, unavailable responses, invalid intermediate labels, or final codes absent from the candidate list are marked as unresolved. Unresolved final predictions count as incorrect in Hit@$K$.

The evaluated model IDs are `openai/gpt-5.4-nano` and `google/gemini-3.1-flash-lite`, selected as cost-conscious LLMs from different providers. The implementation, generated artifacts, and curated benchmark data are available at `https://github.com/rrdls/sinapi-kg-entity-linking`.

### 4.4. Evaluation Metrics

The evaluation reports final linking quality, candidate-generation quality, stage-wise diagnostics, and operational diagnostics. The evaluation unit is a spreadsheet-derived service description with at least one valid target SINAPI code. Before comparison, textual labels are normalized by converting them to uppercase, removing diacritics, and collapsing whitespace; attribute values are additionally normalized by removing internal spaces.

Final linking quality is measured with Hit@$K$, where $K$ is the maximum number of final service items returned by LLM Disambiguation. A prediction is correct when at least one selected SINAPI code belongs to the valid target set for the description. Empty outputs, unresolved disambiguation results, and selected codes outside the valid target set count as errors. Hit@$K$ is the main comparison metric because both approaches use the same output limit $K$. In the experiments, $K = 3$, so final-linking results are reported as Hit@3.

Candidate generation is evaluated with Candidate Recall@$k$, which equals 1 when at least one valid target SINAPI code appears in the candidate list passed to LLM Disambiguation, and 0 otherwise. This metric separates retrieval failures from LLM Disambiguation failures and is reported for both approaches.

The proposed pipeline is also evaluated with stage-wise diagnostics. Class Selection uses precision, recall, and F1. Functional Group Selection, Documentation Selection, and Component Resolution use Hit over their stage output sets, limited to at most three labels in this experiment; a stage is correct when at least one gold-compatible label remains available for downstream processing. This reflects their role as path-preserving filters rather than final single-label decisions: their objective is to keep the reference-compatible catalog path available for Attribute Extraction, KG Candidate Retrieval, and LLM Disambiguation. Attribute Extraction uses precision, recall, and F1 over aggregated gold and predicted attribute–value pairs.

Operational diagnostics include candidate-set size, component-resolution rate, retrieval rate, LLM call count, token usage, execution time, unresolved-output counts, and no-answer counts. Because the baseline does not produce graph-validated functional groups, documentation units, components, or controlled attributes before retrieval, stage-wise metrics are reported only for the proposed pipeline.

## 5. Results and Discussion

This section compares the proposed LLM+KG pipeline with the hybrid retrieval baseline. Here, $k$ denotes the maximum number of retrieved candidates passed to LLM Disambiguation, whereas $K = 3$ denotes the maximum number of final SINAPI service items returned by the LLM. Final linking quality is reported as Hit@3, candidate generation with Candidate Recall@$k$, and stage-wise outputs diagnose where the valid target path is lost.

### 5.1. Final Linking and Candidate Retrieval

Table 1 reports Hit@3 and Candidate Recall@$k$ for both approaches across the evaluated models and candidate-list sizes. Since the benchmark, SINAPI scope, candidate-list sizes, and disambiguation protocol are fixed, the comparison isolates the linking strategy: staged graph-constrained linking versus direct dense–sparse (hybrid) retrieval followed by LLM Disambiguation. In the table, KG denotes the proposed LLM+KG pipeline, Base. denotes the hybrid retrieval baseline, Rec. denotes Candidate Recall, and $\Delta$ pp denotes the absolute difference in percentage points.

The proposed LLM+KG pipeline achieved higher Hit@3 than the baseline in all six model–$k$ configurations, with gains from 10.73 to 32.77 percentage points. The gap is largest at $k = 5$, where direct retrieval often fails to place a valid SINAPI code among the

**Table 1. Final linking and candidate-generation performance. Higher is better.**

| Model | $k$ | KG Hit@3 | Base. Hit@3 | $\Delta$ pp | KG Rec.@$k$ | Base Rec.@$k$ |
|---|---|---|---|---|---|---|
| `gpt-5.4-nano` | 5 | 86.72 | 61.86 | 24.86 | 87.01 | 62.71 |
| `gpt-5.4-nano` | 10 | 87.85 | 71.47 | 16.38 | 88.42 | 74.01 |
| `gpt-5.4-nano` | 15 | 87.85 | 77.12 | 10.73 | 88.42 | 82.77 |
| `gemini-3.1-flash-lite` | 5 | 95.20 | 62.43 | 32.77 | 96.05 | 62.99 |
| `gemini-3.1-flash-lite` | 10 | 96.33 | 72.60 | 23.73 | 96.89 | 73.73 |
| `gemini-3.1-flash-lite` | 15 | 96.61 | 82.49 | 14.12 | 97.46 | 83.05 |

top candidates. As $k$ increases, the baseline improves because more valid targets enter the candidate list; however, even at $k = 15$, the LLM+KG pipeline remains ahead by 10.73 percentage points for `gpt-5.4-nano` and 14.12 for `gemini-3.1-flash-lite`. In contrast to the baseline, the proposed pipeline already has high Candidate Recall@$k$ at $k = 5$, suggesting that graph-constrained stages make valid items available with smaller candidate lists. The small gap between Candidate Recall@$k$ and Hit@3 further indicates that final disambiguation adds relatively few errors once a valid target code is present.

### 5.2. Stage-wise Diagnostic Analysis

Table 2 reports intermediate metrics for the best-performing configuration, `gemini-3.1-flash-lite` with $k = 15$. These diagnostics indicate whether each stage preserves at least one gold-compatible route to a valid SINAPI code. F1 is used for stages that output sets of labels or attributes; Hit is used for routing stages and counts success when at least one compatible label remains available. Candidate Recall@$k$ checks whether a valid code is retrieved among the $k$ candidates, while Hit@3 checks whether the final LLM output contains a valid code. In the table, $n$ denotes the number of benchmark instances considered by each metric; smaller values indicate stage-specific subsets, such as Attribute Extraction instances with applicable branches ($n = 348$) and after correct Component Resolution ($n = 325$).

**Table 2. Stage-wise diagnostic performance of the proposed LLM+KG pipeline.**

| Stage | Metric | $n$ | Value (%) |
|---|---|---|---|
| SINAPI Class Selection | F1 | 354 | 99.72 |
| Functional Group Selection | Hit | 354 | 98.87 |
| Documentation Selection | Hit | 354 | 97.74 |
| Component Resolution | Hit | 354 | 98.31 |
| Attribute Extraction | F1 | 348 | 91.32 |
| Attribute Extraction after correct component | F1 | 325 | 93.22 |
| KG Candidate Retrieval | Recall@$k$ | 354 | 97.46 |
| LLM Disambiguation | Hit@3 | 354 | 96.61 |

The stage-wise profile shows reliable initial routing: Class Selection reaches 99.72% F1, Functional Group Selection 98.87% Hit, Documentation Selection 97.74%, and Component Resolution 98.31%. Attribute Extraction is the weakest measured intermediate stage, with 91.32% F1 overall and 93.22% after correct component resolution, indicating that many remaining errors come from missing or mismatched attributes such as dimension, material, installation condition, or execution details. Candidate Recall@15 reaches 97.46%, only 0.85 percentage points above final Hit@3, so most errors occur before or during candidate retrieval rather than in final LLM disambiguation.

### 5.3. Operational Cost and Accuracy–Cost Trade-off

The higher linking quality of the proposed pipeline comes with higher LLM-call and token usage. The baseline uses one LLM call per description, while the LLM+KG pipeline uses multiple structured stages before disambiguation: 6.31 calls per description for `gpt-5.4-nano` and 6.35 for `gemini-3.1-flash-lite`. Table 3 relates average token usage to the Hit@3 gain; Token ratio is computed as KG tokens divided by baseline tokens.

**Table 3. LLM-call and token usage with final Hit@3 gain. Token values are averages per description.**

| Model | $k$ | Base. tokens | KG tokens | Token ratio (KG/Base) | $\Delta$ Hit@3 pp |
|---|---|---|---|---|---|
| `gpt-5.4-nano` | 5 | 2.9k | 14.6k | 4.99 | 24.86 |
| `gpt-5.4-nano` | 10 | 5.0k | 16.2k | 3.25 | 16.38 |
| `gpt-5.4-nano` | 15 | 7.0k | 17.4k | 2.49 | 10.73 |
| `gemini-3.1-flash-lite` | 5 | 2.8k | 16.1k | 5.66 | 32.77 |
| `gemini-3.1-flash-lite` | 10 | 5.1k | 17.8k | 3.50 | 23.73 |
| `gemini-3.1-flash-lite` | 15 | 7.4k | 19.4k | 2.64 | 14.12 |

The trade-off is clearest at $k = 5$: the LLM+KG pipeline uses about five to six times more tokens than the baseline, but also produces the largest Hit@3 gains. As $k$ increases, the baseline becomes more competitive because Candidate Recall@$k$ improves, and the token ratio falls to 2.49 for `gpt-5.4-nano` and 2.64 for `gemini-3.1-flash-lite`. These results characterize the proposed pipeline as an accuracy–cost trade-off: the additional LLM calls buy higher linking quality and auditability, while the baseline remains a lower-token alternative when retrieval alone is sufficient.

### 5.4. Error Patterns and Limitations

No-answer counts are lower for the staged approach. In the baseline, instances without an LLM Disambiguation answer decrease as $k$ grows, from 53 to 40 for `gpt-5.4-nano` and from 67 to 25 for `gemini-3.1-flash-lite`. In the LLM+KG pipeline, the corresponding counts are lower: 27, 23, and 24 for `gpt-5.4-nano`, and 7, 5, and 6 for `gemini-3.1-flash-lite` across $k = 5, 10, 15$. This suggests that graph-constrained retrieval provides more coherent candidate sets for LLM Disambiguation.

Remaining errors are associated with underspecified spreadsheet descriptions, missing or mismatched attributes, occasional documentation-unit or component confusions, and final ambiguities among technically close SINAPI services. This pattern is consistent with SINAPI linking: small details such as diameter, voltage, material, installation condition, or included input items can determine the final code.

The benchmark is based on spreadsheet-derived descriptions, which are often short and concentrate relevant lexical evidence in a single item. This setting can favor direct hybrid retrieval when the correct SINAPI service shares explicit terms with the source description and a larger candidate list is allowed. A potential advantage of the staged LLM+KG formulation should therefore be investigated in longer narrative project documents, where evidence about component, material, dimension, execution condition, and scope may be distributed across sentences or sections.

Several threats to validity should be considered. The benchmark is description-oriented, not a full cost-estimate or construction-activity model. The gold standard allows multiple valid SINAPI codes when descriptions are underspecified, so Hit@$K$ measures compatibility with a valid target set rather than recovery of a single unique item. The evaluated scope is limited to hydrosanitary and electrical installation services represented in the KG, and both approaches depend on graph completeness, candidate metadata, and model-specific structured-output behavior.

## 6. Conclusion

This paper addressed the problem of linking construction service descriptions to standardized SINAPI cost items. The proposed staged LLM+KG pipeline uses LLMs for language-dependent interpretation, while a SINAPI knowledge graph constrains catalog decisions through documentation units, component groups, attributes, valid values, and graph-retrieved candidates.

The pipeline improves final Hit@3 by 10.73 to 32.77 percentage points over the dense–sparse (hybrid) retrieval-and-disambiguation baseline, reaching 96.61% Hit@3 and 97.46% Candidate Recall@15 in the best configuration. Stage-wise analysis shows that the trace can diagnose whether errors arise in class selection, documentation selection, component resolution, attribute extraction, candidate retrieval, or final disambiguation. The main contribution is to formulate SINAPI cost item identification as auditable entity linking rather than only text retrieval or classification. Future work includes extending the graph to additional SINAPI classes, improving normalization, evaluating longer narrative documents, and investigating symbolic post-validation of selected services.

## References


Akanbi, T. and Zhang, J. (2021). Design information extraction from construction specifications to support cost estimation. *Automation in Construction*, 131:103835.

Boscariol, M., Meschini, S., and Tagliabue, L. C. (2024). A methodological approach to asset information management via knowledge graphs and large language models. In *Proceedings of the 2024 European Conference on Computing in Construction*.

Brasil (2021). Lei nº 14.133, de 1º de abril de 2021. Lei de Licitações e Contratos Administrativos.

CAIXA (2026). SINAPI – sistema nacional de pesquisa de custos e Índices da construção civil. Página institucional. Acesso em: 12 jun. 2026.

Deza, J. I., Ihshaish, H., and Mahdjoubi, L. (2022). A machine learning approach to classifying construction cost documents into the international construction measurement standard. *arXiv preprint arXiv:2211.07705*.

Ding, Y., Ma, J., and Luo, X. (2022). Applications of natural language processing in construction. *Automation in Construction*, 136:104169.

Farghaly, K., Soman, R., and Whyte, J. (2024). cSite ontology for production control of construction sites. *Automation in Construction*, 158:105224.

Gatto, C., Mirarchi, C., and Pavan, A. (2025). Towards automation in cost estimation: LLM-based methodology for classifying and extracting cost data from tender docu-

ments. In *Proceedings of the 2025 European Conference on Computing in Construction*.

IBGE (2026). SINAPI – sistema nacional de pesquisa de custos e Índices da construção civil. Página institucional. Acesso em: 19 jun. 2026.

Iranmanesh, S., Saadany, H., and Vakaj, E. (2025). LLM-assisted Graph-RAG information extraction from IFC data. In *Proceedings of the 2025 European Conference on Computing in Construction*.

Qdrant (2026). Hybrid and multi-stage queries. Documentação técnica. Acesso em: 19 jun. 2026.

Rasmussen, M. H., Lefrançois, M., Pauwels, P., Hviid, C. A., and Karlshøj, J. (2019). Managing interrelated project information in AEC knowledge graphs. *Automation in Construction*, 108:102956.

Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

Sevgili, Ö., Shelmanov, A., Arkhipov, M., Panchenko, A., and Biemann, C. (2022). Neural entity linking: A survey of models based on deep learning. *Semantic Web*, 13(3):527–570.

Shamshiri, A., Ryu, K. R., and Park, J. Y. (2024). Text mining and natural language processing in construction. *Automation in Construction*, 158:105200.

Shen, W., Wang, J., and Han, J. (2015). Entity linking with a knowledge base: Issues, techniques, and solutions. *IEEE Transactions on Knowledge and Data Engineering*, 27(2):443–460.

Sousa, L. J. d., Martins, J. P., Sanhudo, L., and Silva, J. M. (2025). Portuguese construction dataset for AI: BOQ text extraction and synthetic data augmentation using LLMs. In *Proceedings of the 2025 European Conference on Computing in Construction*.

Stellmacher, D., Große Lögten, K., Theißen, S., Kelm, A., and Meins-Becker, A. (2025). Digital decision support in construction: Analysis of information requirements and data provision for AI-based selection of sustainable building products. In *Proceedings of the 2025 European Conference on Computing in Construction*.

Suárez-Figueroa, M. C., Gómez-Pérez, A., and Fernández-López, M. (2015). The NeOn methodology framework: A scenario-based methodology for ontology development. *Applied Ontology*, 10(2):107–145.