Can LLMs be Knowledge Graph Curators for Validating Triple Insertions?

André Gomes Regino
Institute of Computing – University of Campinas – Brazil
Center for Information Technology Renato Archer – Brazil
andre.regino@students.ic.unicamp.br

Julio Cesar dos Reis
Institute of Computing – University of Campinas – Brazil
jreis@ic.unicamp.br

Abstract

As Knowledge Graphs (KGs) become central to modern applications, automated methods for validating RDF triples before insertion into these graphs are essential. The complexity and scalability challenges in manual validation processes have led researchers to explore Large Language Models (LLMs) as potential automated validators. This study investigates the feasibility of using LLMs to validate RDF triples by focusing on four distinct and complementary validation tasks: class and property alignment, URI standardization, semantic consistency, and syntactic correctness. We propose a systematic validation method that uses prompts to guide LLMs through each stage of the triple evaluation of the RDF. In our experiments, four models are evaluated across these tasks. Our results reveal that more advanced models like Llama-3-70B-Instruct offer superior accuracy and consistency. Our findings emphasize the practical open challenges of deploying LLMs in real-world RDF validation scenarios, including domain generalization, semantic drift, and the need for human-in-the-loop interventions. This investigation advances the research on the refinement and integration of LLM-based RDF validation techniques into KG management workflows.

1 Introduction

Knowledge Graphs (KGs) have emerged as essential artifacts to represent structured knowledge in various digital applications, such as search engines, recommendation systems, and question-answering platforms. Having underlying logical and semantically consistent KGs is relevant for applications because they rely on them to improve user experience and help decision making.

At the core of KGs are Resource Description Framework (RDF) triples, which consist of subject-predicate-object expressions that form the basic building blocks of KGs. An RDF triple links a subject to an object through a predicate, encapsulating a single piece of knowledge (Bizer et al., 2023). In this context, ontologies play a significant role in KGs by defining structured schema by ensuring consistency and semantic integrity within KGs. Ontologies specify the classes, properties, and relationships that form the backbone of KGs by guiding the data management and querying processes.

Maintaining the integrity and consistency of KGs as new RDF triples are added is a complex and ongoing challenge. Traditional methods for validating and inserting RDF triples often involve manual efforts by ontology experts, which can be time consuming and prone to human errors. These methods struggle to keep pace with modern data environments' dynamic and large-scale nature. The limitations of current approaches highlight the urgent need for advanced automated tools that can support ontology experts in the management of KG. Automating the identification and elimination of erroneous information improves efficiency and accuracy, reducing the dependency on extensive human intervention.

The insertion of new RDF triples into an existing KG presents issues that can undermine the graph's reliability and usability. First, there is the problem of violating the predefined classes and properties in place, where new triples might not conform to the established ontology schema. Second, URI standardization and duplication pose significant challenges; ensuring that new triples do not introduce redundant or conflicting URIs is essential for maintaining a coherent KG. Third, semantic inconsistency is an issue, as newly added triples might contradict existing knowledge, leading to logical inconsistencies within the graph. Lastly, the syntactic correctness of the triples, respecting a pre-defined language (*e.g.,* n-triples (Beckett et al., 2014) and turtle (Beckett et al., 2014) syntax), avoids malformed triple errors in RDF parsers. These issues collectively impact the overall effectiveness of KGs, compromising their ability to deliver accurate

Proceedings of the Generative AI and Knowledge Graph Workshop (GenAIK), pages 87–99
January 19, 2025.

and reliable knowledge representation.

Recent advancements in Language Models (LLMs) have opened new avenues for addressing the challenges associated with KG management. LLMs, which excel in various natural language processing (NLP) tasks, have demonstrated capabilities in understanding and generating human language with contextual and semantic accuracy (Tang et al., 2023). The intersection of KGs and LLMs presents a promising opportunity to leverage these models to enhance KG management processes (Pan et al., 2024). We originally hypothesized that the advanced semantic understanding of LLMs could assist in identifying violations of classes and properties, standardizing URIs, and ensuring syntactic and semantic consistency of triples. This integration has the potential to significantly improve the efficiency and reliability of KG management, providing ontology experts with powerful tools to maintain high-quality KGs.

This study investigates and evaluates the use of LLMs in validating and inserting RDF triples into existing KGs without negatively impacting their integrity. Specifically, we develop a methodology that ensures new triples are consistent with the existing KG and conform to underlying ontologies.

The broader implications of this research include potential benefits for both academia and industry, such as more reliable KGs and improved data management processes over time. Integrating LLMs into KG curation tasks can lead to more intelligent and automated knowledge management systems, offering enhanced capabilities for handling complex and dynamic data environments.

This article is organized as follows: Section 2 reviews related work, discussing previous approaches for RDF triple validation. Section 3 defines the addressed issues in validating RDF triples. Section 4 outlines our designed method. Section 5 presents the evaluation procedures and obtained results. Section 6 discusses our findings and open research challenges. Finally, Section 7 draws conclusion remarks.

2 Related Work

This section summarizes existing investigations and approaches to validating RDF triples. A recently published survey describing the intersection between LLMs and KGs (Khorashadizadeh et al., 2024) identified KG validation as an essential research venue. KG validation is categorized into two main approaches: fact-checking and inconsistency detection. Our present solution concentrates on inconsistency detection, a relatively underexplored area within the broader context of KG validation. The survey highlights only one significant study in this domain: ChatRule (Luo et al., 2023). ChatRule is a framework that leverages KGs to build LLM prompts, generating rules to detect inconsistencies within the KG. Our work further extends this field by systematically evaluating the capability of LLMs to validate RDF triples in KG insertion operations, focusing on various types of inconsistencies.

Huaman and Fensel presents a methodical approach to improving KG quality without using LLMs (Huaman and Fensel, 2021). The framework integrates existing tools and workflows to ensure correctness, completeness, and usability of KGs. It employs rule-based methods for quality assessment, using metrics like accuracy and completeness. Cleaning tasks involve schema verification through constraint languages (e.g., SHACL, ShEx) and fact validation using internal consistency checks or external sources like Wikipedia. For enrichment, it detects duplicates and resolves conflicts using tools such as SILK and LIMES.

Frey *et al.* (Frey et al., 2023) demonstrated the evaluation of various LLMs, including GPT-4[1] and Claude 2[2], revealing their proficiency in working with Turtle, an RDF triple serialization format. Their study introduced some tasks to assess the models' ability to parse, understand, and create KGs in Turtle syntax. While newer versions of GPT and Claude demonstrate promising capabilities, they frequently struggle with strict output formatting, often including unnecessary explanations, complicating their integration with RDF tools. We face similar problems with our method and despite these challenges, the models show huge potential for assisting in KG engineering.

The Triples Accuracy Assessment (TAA) (Liu et al., 2017) approach offers an automated method for validating RDF triples in a KG using other KGs. Unlike traditional methods that rely on internal information, TAA identifies equivalent resources across different KGs and matches predicates to assess the correctness of triples. A confidence score is generated to indicate the accuracy of each triple, showing promising results with high F-measure

[1] https://openai.com/index/gpt-4/
[2] https://www.anthropic.com/research

scores in evaluations using the FactBench dataset.

Our originality lies in the innovative use of LLMs to validate RDF triples for KG insertion operations, which traditionally solely rely on rule-based methods or external KG interlinks. Unlike prior approaches, such as the Triples Accuracy Assessment (Liu et al., 2017), which leverages other KGs for validation, our study explores the potential of LLMs to bring deeper semantic understanding and context to the validation process. To the best of our knowledge, this study is the first to systematically assess the potential effectiveness of LLMs in this specific application and across various RDF validation tasks. Our study offers new, original insights into LLMs' potential to enhance the accuracy and efficiency of RDF triple validation.

3 Problem Formulation

This section outlines four critical problems our approach addresses when validating and inserting new triples into a KG using LLMs. The rationale behind choosing the following problems is that they align with existing standards and best practices in RDF and ontology management. They are common underlying problems encountered during the construction and maintenance of KGs.

Problem 1: Violation of Predefined Classes and Properties

One fundamental issue in maintaining KG's integrity is ensuring that new triples adhere to the predefined classes and properties outlined in the ontology. During the generation of triples, it is essential to specify which classes and properties the KG structure requires. The critical task is to verify if any generated triple contains classes or properties not part of the predefined list provided by the ontology maintainer.

Let $\mathcal{C}$ be the set of essential classes and Pr be the important properties the ontology defines. For each triple $t = (s, p, o)$ in the set of $\mathcal{T}$, the predicate p, and the object o (if it is a class) must be elements of Pr and $\mathcal{C}$, respectively.

For $\mathcal{C} = \{Person, Organization, Product\}$, $Pr = \{hasName, isPartOf, produces\}$, $t_1 = (Organization/X, produces, Product/X\_AI)$ and $t_2 = (Person/SteveJobs, born, State/California)$. All the elements from t_1 can be found in $\mathcal{C}$ and Pr. If the object "State" is not in $\mathcal{C}$, then t_2 should be flagged.

Problem 2: URI Standardization

The addition of new triples requires no duplicated URIs within the KG. Duplicates and redundancies increase the size of KGs without adding relevant knowledge. This problem arises when different URIs refer to the same real-world entity. Guaranteeing the uniqueness of URIs is vital to maintaining a coherent representation of entities.

For any new triple $t = (s, p, o)$ in $\mathcal{T}$, the subject s and the object o – if it is a URI – must be checked against existing URIs in the KG. Let $\mathcal{U}$ be the set of all URIs in the existing KG. The new URIs s and o must not introduce duplicates.

If the resource $Car/Tesla\_S\_2023$ is present in the KG, then the addition of $t = (Car/Tesla\_S\_23, hasFeature, Electric\_Drive)$ should be flagged since $Car/Tesla\_S\_2023$ and $Car/Tesla\_S\_23$ refer to the same entity.

Problem 3: Semantic Inconsistency

Semantic inconsistency occurs when new triples contradict the existing triples in the KG. A resource cannot simultaneously possess mutually exclusive properties. Ensuring semantic consistency requires checking the logical compatibility of new triples with the existing KG data.

Let $\mathcal{R}$ be a set of semantic statements and constraints the ontology defines. For each new triple $t = (s, p, o)$ in $\mathcal{T}$, we must verify that t does not violate any rule $r \in \mathcal{R}$ based on the existing triples in the KG.

An example of a rule using Semantic Web Rule Language (SWRL) states that a person can not be sibling and married to the same person: $Sibling(?x, ?y) \wedge MarriedTo(?x, ?y) \rightarrow false$.

If $t_1 = (Phone/iPhone\_X, isCompatibleWith, Gadget/USB\_C)$ in $\mathcal{T}$, and there is an existing t_2 in the KG which states that the iPhone X is incompatible with USB C, adding t_1 would create a contradiction with t_2, based on a criterion $r \in \mathcal{R}$ that states that two resources cannot be compatible and incompatible with each other simultaneously.

Problem 4: Syntactic Inconsistency

In addition to semantic checking, ensuring the syntactic correctness of RDF triples is essential to maintaining the structural integrity of a KG. For instance, a syntactically valid RDF triple using the n-triples syntax must have three components: a subject, a predicate, and an object. Any deviation from this, such as triples with fewer or more than three components, constitutes a syntactic error and

can disrupt the proper functioning of the KG.

Each triple $t = (s, p, o)$ in $\mathcal{T}$ must adhere to the required syntactic structure. This involves checking that each triple has precisely one subject, predicate, and object.

For instance, an existing triple $t_1 = (Island/Santorini, hasPopulation)$, which lacks an object, would be flagged as a syntactic error. A triple like $t_2 = (Island/Crete, hasArea, 8336, km)$ with an extra component would also be erroneous.

4 Validating Generated RDF Triples based on LLMs

Our proposed method consists of four main steps, each involving a specific prompt and requiring the intervention of an ontology maintainer to ensure correctness. These steps systematically validate and prepare RDF triples for precise insertion into an existing KG. The steps address the critical issues of class and property compliance, URI uniqueness, semantic and syntactic consistency, and challenges explained with more details in Section 3. Figure 1 presents our method to validate RDF triples.

The input is a set of RDF triples formatted as $\mathcal{T} = \{(s_1, p_1, o_1), (s_2, p_2, o_2), ..., (s_n, p_n, o_n)\}$. The output is a set of final validated RDF triples as $T_{final} = \{(s_1, p_1, o_1), (s_2, p_2, o_2), ..., (s_n, p_n, o_n)\}$. Algorithm 1 shows the procedure of our method.

The first step (#1 in Figure 1) aims to verify that $\mathcal{T}$ contains classes and properties listed as necessary by the ontology maintainer. The process begins with the maintainer creating a List of Important Classes and Properties $\mathcal{L}_{c,p}$ (line 2 in Algorithm 1). This list outlines the crucial classes and properties that must be present in the RDF triples.

The list $\mathcal{L}_{c,p}$ is manually curated by the ontology maintainer, who possesses a deep knowledge of the KG's structure and the relevant domain. This list is derived directly from the ontology (cf. Figure 1).

Although the important classes and properties human-curated lists may limit generalizability, they are important for ensuring semantic coherence and alignment with the KGs domain. These curated inputs are minimal compared to the automated processing enabled by LLMs in the pipeline.

The LLM evaluates each triple from all triples $\mathcal{T}$ (line 3 of Algorithm 1) to check for compliance with the provided list. The compliance check is defined as: $\forall (s, p, o) \in \mathcal{T}, (class(s) \in \mathcal{L}_{c,p}) \wedge (property(p) \in \mathcal{L}_{c,p})$.

To materialize this step (line 4 of Algorithm 1), we use the prompt[3] $pr_1 = (i_1, \mathcal{T}, \mathcal{L}_{c,p})$ composed of the following components: an initial instruction i_1 on evaluating the presence of properties and classes, the set of RDF triples to be analyzed $\mathcal{T}$ and the List of Important Classes and Properties $\mathcal{L}_{c,p}$. Line 4 of Algorithm 1 shows a summarized version of i_1.

Any triples containing classes or properties not included in the List of Important Classes and Properties are flagged (lines 5 and 6 of Algorithm 1). The ontology maintainer reviews these non-compliant triples and determines whether they should be removed (line 9 of Algorithm 1). This step ensures that all generated triples adhere to the predefined schema.

The second step (#2 in Figure 1) ensures that new triples do not introduce duplicate resources into the KG. After removing the triples flagged in Step 1, the remaining triples $\mathcal{T}$ are checked for resource duplication.

The LLM performs SPARQL queries on the existing KG to identify similar resources, generating a List of Duplicate Resources $\mathcal{L}_{dr}$ (line 12 of Algorithm 1). Different from $\mathcal{L}_{c,p}$, the generation of $\mathcal{L}_{dr}$ does not require human intervention.

SPARQL queries serve as an interface with the KG. The second step uses SPARQL queries to retrieve resources in the KG similar to those in the triples under analysis. We identify these similar resources by querying the KG with SPARQL and filling $\mathcal{L}_{dr}$ with the results.

The prompt (line 13 of Algorithm 1) for this step $pr_2 = (i_2, \mathcal{T}, \mathcal{L}_{dr})$ includes an initial instruction i_2 on identifying duplicate resources, the set of RDF triples to be checked $\mathcal{T}$ and the List of Duplicate Resources $\mathcal{L}_{dr}$.

The ontology maintainer reviews the flagged duplicates and updates the triples as necessary. If a resource is confirmed as duplicate, the maintainer updates the triples to use the correct, existing resource values (line 18 of Algorithm 1). If a resource is erroneously marked as a duplicate, it is ignored. This step guarantees the uniqueness of URIs in the KG, preventing conflicts and ensuring a coherent representation of entities.

The third step (#3 in Figure 1) ensures that the new triples do not violate predefined semantic restrictions. The ontology maintainer provides a List of Semantic Restrictions $\mathcal{L}_{sr}$ (line 21 of Algorithm

[3]All the prompts listed in this section can be found in https://zenodo.org/records/13712876

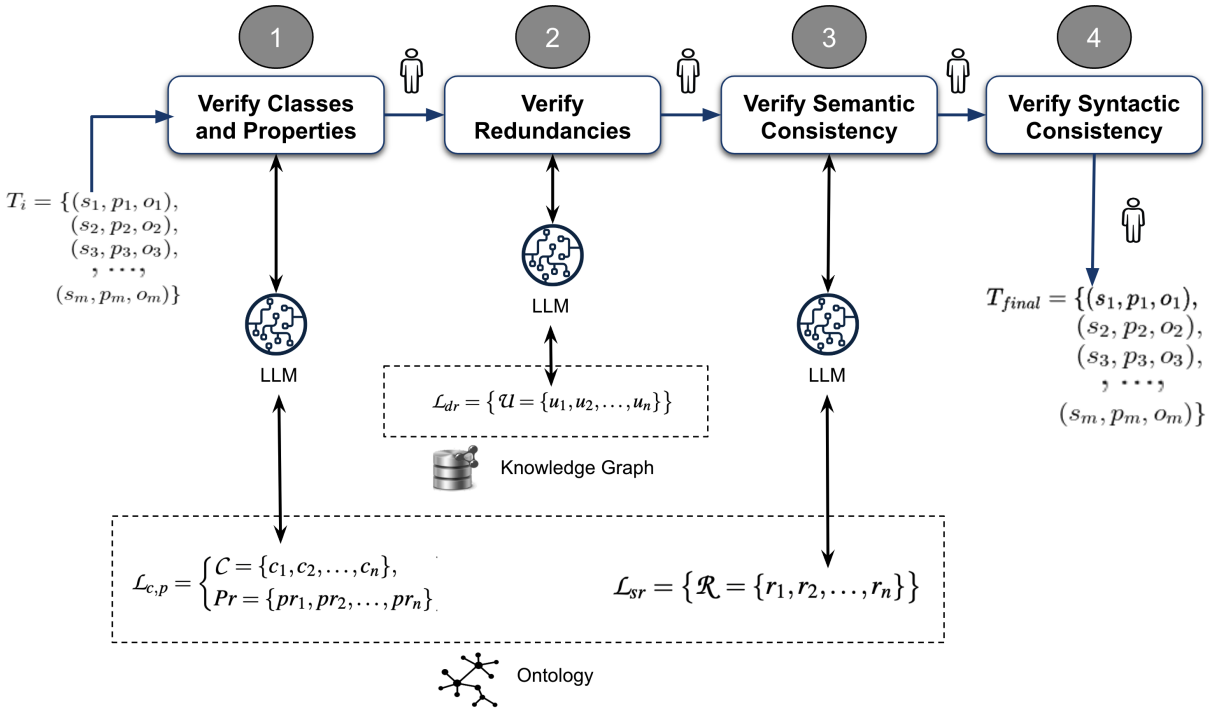

Figure 1: Method to validate RDF triples. The boxes with grey circles represent the steps to transform the initial triples $\mathcal{T}$ in the validated triples T_{final}. Among the steps, the method requires human intervention, represented by the human icon. Steps 1, 2, and 3 use three lists as input: $\mathcal{L}_{c,p}$ and $\mathcal{L}_{sr}$ – part of the ontology – and $\mathcal{L}_{dr}$ – part of the KG.

1), primarily consisting of rules specified in SWRL.

The set of triples modified by the previous steps $\mathcal{T}$ is then compared against these restrictions by the LLM (line 22 of Algorithm 1). The language model identifies any triples that potentially violate the semantic rules.

The prompt for this step $pr_3 = (i_3, \mathcal{T}, \mathcal{L}_{sr})$ includes an initial instruction i_3 on identifying triples with semantic restrictions, the set of RDF triples to be analyzed $\mathcal{T}$ and the List of Semantic Restrictions $\mathcal{L}_{sr}$.

The ontology maintainer reviews the flagged triples and decides whether to remove them (line 27 of Algorithm 1). This step prevents the introduction of logical contradictions, such as an object being simultaneously marked as compatible and incompatible with another object, thus maintaining the semantic integrity of the KG.

The final step (#4 in Figure 1) ensures the syntactic correctness of the RDF triples before they are inserted into the KG. The set of triples modified by the previous steps, $\mathcal{T}$, is provided as input, and the language model checks for any syntactic errors (line 30 of Algorithm 1).

This step does not require additional lists as the previous steps. The language model identifies and flags triples that do not conform to the required RDF structure (line 34 of Algorithm 1), ensuring that only syntactically correct triples are considered for insertion into the KG.

The prompt for this step $pr_4 = (i_4, \mathcal{T})$ includes an initial instruction i_4 about the syntactic validation and the set of RDF triples to be analyzed $\mathcal{T}$.

The ontology maintainer proceeds with a final validation on the flagged triples and T_{final}, ensuring they are ready to be inserted in the KG.

5 Evaluation

This evaluation assesses if the developed method and the designed prompts instructing the LLMs can effectively identify and correct specific issues within the RDF triples. The evaluation focuses on the four distinct problems identified in Section 3 and addressed by our solution (Section 4).

Section 5.1 describes the models, datasets, and procedures used in this evaluation. Section 5.2 demonstrates the obtained results.

5.1 Setup and Procedures

The experimental evaluation used four distinct Language Models: Bloom-176B (Scao et al., 2022), Mixtral-7B-Instruct (Jiang et al., 2024), Gemma2-

9B-Instruct (Team, 2024; Team et al., 2024), and Llama-3-70B-Instruct (AI@Meta, 2024). We chose Bloom because it was one of the first large-scale language models launched, setting a precedent in the open-source community. Among the four models, Bloom is the largest, with 176 billion parameters, which enables it to capture a wide range of linguistic nuances and knowledge. Bloom is free, although it limits the number of tokens generated per minute[4]. These factors were key reasons for including Bloom in our evaluation.

Mixtral was selected for its unique architecture as a mixture of experts (Jiang et al., 2024), differentiating it from the other LLMs. This model combines multiple specialized sub-models, or "experts", to process different input parts, allowing for more efficient computations. Despite being a smaller language model with 7 billion parameters, Mixtral is cost-effective and has demonstrated impressive results, even outperforming some closed-source LLMs like GPT-3.5 (Jiang et al., 2024).

Gemma 2 was included because it originated as an open-source model developed by Google (Team et al., 2024), known for competitive results on public LLM leaderboards. With 9 billion parameters, Gemma 2 balances size and computational cost. Its performance relative to its size, cost, and open-source nature justified its selection for our study.

Finally, Llama-3-70B was chosen because it is one of the top-performing models on the LLM leaderboard[5], especially considering its size of 70 billion parameters. Produced by Meta, Llama-3 inclusion in our evaluation was driven by its leading performance, size, and alignment with the other open-source models in our study. Together, these models represent the current state of open-source LLMs across various scales and architectures.

The dataset consists of 500 records of questions and answers related to product compatibility from ten different e-commerce stores. This dataset was generated in 2023 using random samples of actual customer interactions. These e-commerce stores are customers of GoBots[6], a Brazilian AI startup specializing in e-commerce solutions. The GoBots maintains an existing KG focused on product compatibility, which has been successfully deployed in a production environment. The triples used in our evaluation are sourced directly from this KG. They reflect real-world scenarios and have proven their utility in supporting e-commerce operations.

Each of the 500 records includes (1) A question posed by a customer about the compatibility of a car with a product; (2) An answer provided by a seller indicating compatibility or incompatibility; (3) A set of RDF triples associated with the question-answer pair, representing the car, the product, and their compatibility status[7]. The RDF triples were automatically generated by a system developed by the Brazilian AI startup. This system generates and integrates RDF triples into an existing KG (Sant'Anna et al., 2020).

To evaluate specific aspects of this investigation, noise was randomly introduced into the dataset, targeting particular defined problems. It is important to note that these noises were added automatically, ensuring the randomness of the process and eliminating any possibility of bias that could be attributed to manual interference. This approach was deliberately chosen to ensure a fair and unbiased evaluation of the model's ability to handle data inconsistencies, regardless of how the noise was introduced.

- *Noise type 1*: For 100 randomly chosen records, triples with classes and properties not allowed are added to the existing triples (problem 1);
- *Noise type 2*: For another 100 randomly chosen records, resources similar to existing resources (with minor modifications like year changes) are added (problem 2);
- *Noise type 3*: For another 100 randomly chosen records, RDF triples indicating false compatibility (contradicting existing SWRL rules) are introduced (problem 3);
- *Noise type 4*: For another 100 randomly chosen records, RDF triples with four components are added at the end of the triple list (subject, predicate, object, and a random fourth component), disrupting the syntactic consistency (problem 4);
- *Control*: No noise is added for the remaining 100 records of the dataset, serving as a control group.

[4] https://huggingface.co/bigscience/bloom

[5] https://huggingface.co/spaces/lmsys/chatbot-arena-leaderboard

[6] https://gobots.ai/

[7] An example of a dataset record can be found in https://zenodo.org/records/13722627

Each of the 500 records contains either one type of noise or no noise (in the case of the 100 records from the control group). No record contains more than one type of noise. The evaluation measured each model's accuracy, precision, recall, and F1 score in identifying the introduced noise types.

The evaluation followed the following steps:

1. **Noise Introduction:** Introduced specific types of noise into the dataset to simulate the four problems (as described).

2. **Method Execution:** Apply the corresponding prompts to the dataset:

    - Prompt 1: Identifies triples with classes and properties not allowed by the ontology. We added a noisy RDF triple, indicating the car speed. There is no class or property in the ontology (and consequently in the list of allowed classes and properties) related to car speed;
    - Prompt 2: Detects duplicate resources. We added noisy RDF triples related to the model year of a car. For instance, we added the triple related to the car "HRV 21", expecting that the LLM could detect the duplication with an already existing resource in the KG, "HRV 2021";
    - Prompt 3: Checks semantic consistency by searching for contradictions in compatibility among products and cars. We added the example from Section 4, adding SWRL related to compatible and incompatible products and cars. We added noisy compatibility triples, expecting that the LLM could identify them;
    - Prompt 4: Verifies the correct syntax of triple insertion. We added noisy RDF triples with four components.

3. **Metrics Computation:** Calculate accuracy, precision, recall, and F1 for each prompt by each model, evaluating the number of correctly identified records versus false positives and false negatives. For example, in the case of 100 records with noise from problem 1, a true positive is when the model correctly identifies the problem in a record. A true negative occurs when the model correctly identifies that one of the remaining 400 records has no issues. A false negative would be when the model fails to identify problem 1 in one of the 100 problematic records, while a false positive would occur if the model incorrectly identifies one of the 400 noise-free records as problematic. These values are used to compute the evaluation metrics for each model.

4. **Analysis of Results:** Quantitative analysis to determine the effectiveness of each prompt in addressing the specific problems.

5.2 Results

In evaluating the models across the four RDF validation problems, a clear trade-off emerges between precision, recall, and accuracy. For instance, in the Class and Properties Violation Problem, the Llama-3 70B Instruct model got an accuracy of 0.84, coupled with a balanced precision and recall of 0.78 and 0.89, respectively. This indicates that the model was good at identifying valid triples and minimizing false positives and negatives. On the other hand, Bloom-176B showed a more balance between precision and recall (0.54 vs. 0.56) but at a much lower accuracy (0.55), reflecting difficult-to-maintain consistent results across the scenarios.

We observed that models like Mixtral-7B and Gemma2-9B exhibit higher recall than precision in some instances, such as the "Class and Properties Violation" and "Syntactic Inconsistency Problem". This comes at the cost of higher false positives, reflected in lower precision. The balance between these metrics suggests that selecting a model for RDF validation requires prioritizing the metrics most relevant to the specific validation scenario, whether catching more errors (recall) or ensuring fewer false positives (precision).

The models showed varying degrees of sensitivity to different types of RDF validation issues, revealing insights into their strengths and weaknesses. In the "Syntactic Inconsistency Problem", where adherence to RDF structure is required, Llama-3 70B Instruct outperformed all other models with almost perfect accuracy (0.99) and F1 score (0.98). This indicates that this model is well-suited for tasks requiring precise syntactic validation. However, Bloom-176B struggled with syntactic errors, achieving a low accuracy of 0.36, suggesting it is less adept at handling structural rules.

In the "Semantic Inconsistency Problem" and "URI Standardization", which involves relationships and contextual knowledge, Gemma2-9B showed higher metric values than in syntactic tasks. This could be attributed to their ability to recog-

nize complex ontological relationships, although their recall and F1 scores are behind Llama-3. The results suggest that while some models specialize in specific RDF issues, they face challenges when encountering unfamiliar error types.

6 Discussion and Open Research Challenges

This research inquired how LLMs can be suited to contribute as KG Curators in the operations of triple insertion. This research demonstrated that LLMs for the distinct problems addressed can be applicable as an approach to help ontology engineers address RDF validation. In the following, we underline key findings and challenges regarding several aspects of our experimental results and the consequences of applying our solution to operational settings.

The most performing LLM. Overall, we found that the Llama-3 70B Instruct model consistently outperformed the others across all validation problems, excelling in tasks that demand high precision and recall. Its effectiveness in the "Syntactic Inconsistency Problem" (0.99 accuracy) and "URI Standardization Problem" (0.96 accuracy) underscored its robustness in handling structural data such as RDF triples. In our understanding, this model's success is due to its large parameter size and fine-tuning, which are geared explicitly towards instruction-based tasks, enabling it to generalize across diverse RDF validation scenarios.

Underperformance consistently. Conversely, Bloom-176B consistently underperformed, particularly in the "Semantic Inconsistency Problem" (0.29 accuracy) and "Syntactic Inconsistency Problem" (0.36 accuracy). Its lower accuracy and inconsistent precision-recall balance show its limitations in handling the rule-based nature of RDF validation. The gap in results between Llama-3 and Bloom can be explained by differences in model size, training datasets, domain-specific tuning, and more than two years between the release of both models.

The most challenging problems. Discussing the four validation problems, the "URI Standardization Problem" obtained the best overall results, with a mean accuracy of 0.71. This can be attributed to the nature of 'standardizing URIs', which primarily involves pattern recognition that LLMs are well-equipped to handle. The best model, Llama-3, achieved a near-perfect accuracy of 0.96 for this problem, showing its ability to manage standardized data consistently. On the contrary, the "Syntactic Inconsistency Problem" proved to be the most challenging overall, with an average accuracy of 0.53 and a mean F1 score of 0.47. This difficulty arose from Gemma 2 reaching a precision of 0.08 and accuracy of 0.26, which was the worst precision and accuracy of the evaluation. Mixtral-7B got better results in this task. Comparing the two models with similar sizes, Mixtral outperformed Gemma 2 in semantic-related tasks, and Gemma 2 outperformed Mixtral in syntactic-related tasks.

Cost vs. Accuracy Trade-off. The cost-effectiveness of deploying different LLMs for RDF triple validation is critical for real-world applications. For instance, while the Llama model achieved superior accuracy and overall metrics, it comes with a significant computational cost of $0.88 per million tokens. In contrast, the Gemma-2 9B Instruct model, which costs $0.30 per million tokens, provides a balanced trade-off between cost and accuracy but falls short of achieving the precision needed for more complex scenarios. Mixtral 7B Instruct offers a middle ground in cost and performance at $0.60 per million tokens. At the same time, Bloom is a freely available model that, despite being cost-free, exhibits significantly lower accuracy and reliability. These costs were gathered in two companies that provide LLMs APIs: TogetherAI[8] and Hugging Face[9]. The costs reflect the price found when this manuscript was written - September 2024.

Semantic Drift in Long Triple Chains. One issue encountered in RDF triple validation using LLMs is the potential for semantic drift when evaluating long chains of interconnected triples. In this context, semantic drift refers to the model losing coherence as it processes extended sequences. This drift is increased by the models' limited memory retention and inability to consistently track relationships across multiple triples. Triples involving big and deep ontological hierarchies or chains that span various levels may introduce errors as the models struggle to maintain context. As a future work, addressing this challenge may require fine-tuning LLMs with specific datasets designed to enhance memory retention over long sequences or integrating mechanisms that allow for continuous context tracking in KG context. Without such interventions, long triple chains remain a source of inaccuracy.

[8] https://api.together.ai/models
[9] https://huggingface.co/models

Ontology Complexity and Coverage. The complexity of ontologies, characterized by rich hierarchies, specialized vocabularies, and relationships, introduces significant challenges for LLM-based RDF validation. The method revealed that as ontologies grow more complex, models like Bloom and Mixtral struggle to navigate the intricate set of classes and properties accurately. A notable issue is incomplete ontology coverage, where the models lack sufficient information about specialized vocabularies, leading to false positives or negatives during validation. For example, triples involving lesser-known properties or deep subclass hierarchies often went unrecognized, highlighting gaps in the models' ontological understanding. Addressing this issue may require expanding the training datasets to include more comprehensive ontology samples for future work.

Ontology Size. The experiments conducted in this study did not suffer from token limitation, as the ontology used is relatively small and well within the context size limits of the employed LLMs. However, we acknowledge that scaling the approach to large ontologies remains an open issue. Future work will explore strategies to handle extensive schemas, such as breaking them into subsets or leveraging hierarchical representations to fit within the token constraints of LLMs.

7 Conclusion

Ensuring the quality and consistency of KGs is critical for real-world applications that rely on semantic accuracy. As KGs become more integral to artificial intelligence systems, advancing methods for their automated validation might play a key role in driving accurate, reliable, and scalable semantic solutions. Our study explored using LLMs to validate RDF triples by addressing critical challenges in automating a traditionally manual process. We showcased the strengths and limitations of current LLMs in KG curation by examining the effectiveness of models like Llama-3-70B-Instruct and Bloom-176B across four RDF validation tasks. The Llama-3 model demonstrated competitive results, particularly in maintaining syntactic and semantic consistency, showing the potential for real-world deployment. Our results highlighted the complexity and cost implications, especially in handling errors requiring more context. The findings suggested future research directions, including more sophisticated approaches to reducing semantic drift in longer triple chains and enhancing model generalization across domains. Also, incorporating humans into the loop and refining prompt engineering techniques could enhance LLM results.

Limitations

One limitation found during the development of this investigation is the low accuracy of some models when handling intricate RDF syntax and semantics. For instance, models like Bloom-176B demonstrated considerable inconsistency in detecting syntactic errors, due to their less targeted training. This variability among models indicates that not all LLMs can address complex validation tasks, suggesting a need for further fine-tuning and model selection based on specific KG characteristics.

Another limitation was handling with long triple chains, where models experienced semantic drift. Certain LLMs struggled to retain the necessary context across interconnected triples as the chain increased, leading to validation inaccuracies. Addressing this drift might require models specifically trained to manage extended sequences or incorporate a human-in-the-loop strategy. Additionally, the significant computational cost of more accurate models, like Llama-3-70B, limits scalability in practical applications, where cost-effective but reliable validation solutions are desirable.

We acknowledge that the proposed approach involves some degree of manual effort, mainly through the involvement of the ontology maintainer in providing inputs such as lists of essential classes and properties. However, this involvement is necessary to ensure the RDF's semantic alignment and domain specificity triples with the existing KG and ontology. Although LLM automation significantly reduces the overall workload, human oversight remains essential to maintain the quality and reliability of the KG.

Acknowledgments

This study was financed by the National Council for Scientific and Technological Development - Brazil (CNPq) process number 140213/2021-0. In addition, this research was partially funded by the São Paulo Research Foundation (FAPESP) (grants #2022/13694-0, #2022/15816-5 and #2024/07716-6). The opinions expressed in this work do not necessarily reflect those of the funding agencies. We thank GoBots for providing the infrastructure used in this research.

References

AI@Meta. 2024. Llama 3 model card.

David Beckett, Tim Berners-Lee, Eric Prud'hommeaux, and Gavin Carothers. 2014. Rdf 1.1 turtle. *World Wide Web Consortium*, pages 18–31.

Christian Bizer, Tom Heath, and Tim Berners-Lee. 2023. Linked data-the story so far. In *Linking the World's Information: Essays on Tim Berners-Lee's Invention of the World Wide Web*, pages 115–143.

Johannes Frey, Lars-Peter Meyer, Natanael Arndt, Felix Brei, and Kirill Bulert. 2023. Benchmarking the abilities of large language models for rdf knowledge graph creation and comprehension: How well do llms speak turtle? *arXiv preprint arXiv:2309.17122*.

Elwin Huaman and Dieter Fensel. 2021. Knowledge graph curation: a practical framework. In *Proceedings of the 10th International Joint Conference on Knowledge Graphs*, pages 166–171.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Hanieh Khorashadizadeh, Fatima Zahra Amara, Morteza Ezzabady, Frédéric Ieng, Sanju Tiwari, Nandana Mihindukulasooriya, Jinghua Groppe, Soror Sahri, Farah Benamara, and Sven Groppe. 2024. Research trends for the interplay between large language models and knowledge graphs. *arXiv preprint arXiv:2406.08223*.

Shuangyan Liu, Mathieu d'Aquin, and Enrico Motta. 2017. Measuring accuracy of triples in knowledge graphs. In *Language, Data, and Knowledge: First International Conference, LDK 2017, Galway, Ireland, June 19-20, 2017, Proceedings 1*, pages 343–357. Springer.

Linhao Luo, Jiaxin Ju, Bo Xiong, Yuan-Fang Li, Gholamreza Haffari, and Shirui Pan. 2023. Chatrule: Mining logical rules with large language models for knowledge graph reasoning. *arXiv preprint arXiv:2309.01538*.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. 2024. Unifying large language models and knowledge graphs: A roadmap. *IEEE Transactions on Knowledge and Data Engineering*.

Diogo Teles Sant'Anna, Rodrigo Oliveira Caus, Lucas dos Santos Ramos, Victor Hochgreb, and Julio Cesar dos Reis. 2020. Generating knowledge graphs from unstructured texts: Experiences in the e-commerce field for question answering. In *ASLD@ ISWC*, pages 56–71.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Xiaojuan Tang, Zilong Zheng, Jiaqi Li, Fanxu Meng, Song-Chun Zhu, Yitao Liang, and Muhan Zhang. 2023. Large language models are in-context semantic reasoners rather than symbolic reasoners. *arXiv preprint arXiv:2305.14825*.

Gemma Team. 2024. Gemma.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

A Appendix - Algorithm

B Appendix - Summary of Results

Table 1 presents the results. It demonstrates varied effectiveness across the four evaluated problems, with differences in accuracy, precision, recall, and F1-score among the four language models.

For **Problem 1**, which focused on detecting violations of predefined classes and properties, Llama-3-70B-Instruct achieved the highest accuracy (0.84) and F1-score (0.80), followed by Mixtral-7B-Instruct with an accuracy of 0.64 and an F1-score of 0.61. The overall mean accuracy for this problem across all models was 0.61, with a mean precision of 0.65, recall of 0.71, and F1-score of 0.58.

Algorithm 1 Our Method for Validating RDF Triples for Knowledge Graph Insertion

Require: Set of RDF triples $\mathcal{T}$, List of Important Classes and Properties $\mathcal{L}_{c,p}$, Knowledge Graph $\mathcal{KG}$, List of Semantic Restrictions $\mathcal{L}_{sr}$

Step 1: Verify Classes and Properties using LLM and Prompt 1
$\mathcal{L}_{c,p} \leftarrow$ createListOfImportantClassesAndProperties() ▷ Created by ontology maintainer
for each $(s,p,o) \in \mathcal{T}$ **do**
    response $\leftarrow$ LLM(Prompt 1: "Check if the triple (s,p,o) violates any predefined classes or properties in $\mathcal{L}_{c,p}$")
    if *response* = violation **then**
        $flaggedTriples_1 \leftarrow flaggedTriples_1 \cup \{(s,p,o)\}$
    end if
end for
$\mathcal{T} \leftarrow \mathcal{T} \setminus flaggedTriples_1$ ▷ Reviewed by ontology maintainer
Step 2: Verify Redundancies using LLM and Prompt 2
for each $(s,p,o) \in \mathcal{T}$ **do**
    $\mathcal{L}_{dr} \leftarrow$ queryForDuplicateResources(s,o, $\mathcal{KG}$)
    response $\leftarrow$ LLM(Prompt 2: "Check if the triple (s,p,o) contains duplicate or similar resources in $\mathcal{L}_{dr}$")
    if *response* = duplicate **then**
        $flaggedTriples_2 \leftarrow flaggedTriples_2 \cup \{(s,p,o)\}$
    end if
end for
$\mathcal{T} \leftarrow$ updateResourcesInTriples($\mathcal{T}$, $flaggedTriples_2$, $\mathcal{KG}$) ▷ Reviewed by ontology maintainer
Step 3: Verify Semantic Consistency using LLM and Prompt 3
for each $(s,p,o) \in \mathcal{T}$ **do**
    $\mathcal{L}_{sr} \leftarrow$ createListOfRules() ▷ Created by ontology maintainer
    response $\leftarrow$ LLM(Prompt 3: "Check if the triple (s,p,o) violates any semantic restrictions defined in $\mathcal{L}_{sr}$")
    if *response* = violation **then**
        $flaggedTriples_3 \leftarrow flaggedTriples_3 \cup \{(s,p,o)\}$
    end if
end for
$\mathcal{T} \leftarrow \mathcal{T} \setminus flaggedTriples_3$ ▷ Reviewed by ontology maintainer
Step 4: Verify Syntactic Consistency using LLM and Prompt 4
for each $(s,p,o) \in \mathcal{T}$ **do**
    response $\leftarrow$ LLM(Prompt 4: "Check if the triple (s,p,o) is syntactically correct")
    if *response* = correct **then**
        $\mathcal{T}_{final} \leftarrow \mathcal{T}_{final} \cup \{(s,p,o)\}$
    else
        $flaggedTriples_4 \leftarrow flaggedTriples_4 \cup \{(s,p,o)\}$
    end if
end for
$\mathcal{T} \leftarrow \mathcal{T} \setminus flaggedTriples_4$ ▷ Reviewed by ontology maintainer
$flaggedTriples \leftarrow flaggedTriples_1 \cup flaggedTriples_2 \cup flaggedTriples_3 \cup flaggedTriples_4$
return $\mathcal{T}_{final}, flaggedTriples$ ▷ Final set of triples ready for insertion into the Knowledge Graph

Concerning **Problem 2**, which involved identifying and standardizing duplicate resources, Llama-3-70B-Instruct achieved the highest results with an accuracy of 0.96, precision of 0.92, recall of 0.97, and F1-score of 0.94. The mean accuracy of each model for this problem was 0.71, with a mean precision of 0.75, recall of 0.73, and F1-score of 0.67.

Table 1: Results of the experimental evaluation. The first column lists the four problems described in Section 3; the second column lists the four LLMs used in the evaluation; the remaining columns show the values of each metric in each model and problem. Bold values represent the best score for each metric and each problem.

Problem	Model	Accuracy	Precision	Recall	F1
1 - Class and Properties Violation	Bloom-176B	0.55	0.54	0.56	0.50
	Mixtral-7B-Instruct	0.64	0.67	0.77	0.61
	Gemma2-9B-Instruct	0.42	0.59	0.61	0.42
	Llama-3-70B-Instruct	**0.84**	**0.78**	**0.89**	**0.80**
	Mean	0.61	0.65	0.71	0.58
2 - URI Standardization	Bloom-176B	0.44	0.50	0.49	0.41
	Mixtral-7B-Instruct	0.52	0.64	0.69	0.51
	Gemma2-9B-Instruct	0.90	**0.93**	0.75	0.80
	Llama-3-70B-Instruct	**0.96**	0.92	**0.97**	**0.94**
	Mean	0.71	0.75	0.73	0.67
3 - Semantic Inconsistency	Bloom-176B	0.29	0.34	0.26	0.26
	Mixtral-7B-Instruct	0.56	0.39	0.36	0.37
	Gemma2-9B-Instruct	0.81	0.83	0.53	0.50
	Llama-3-70B-Instruct	**0.92**	**0.86**	**0.95**	**0.89**
	Mean	0.65	0.61	0.53	0.51
4 - Syntactic Inconsistency	Bloom-176B	0.36	0.32	0.23	0.27
	Mixtral-7B-Instruct	0.50	0.59	0.63	0.49
	Gemma2-9B-Instruct	0.26	0.08	0.37	0.13
	Llama-3-70B-Instruct	**0.99**	**0.99**	**0.97**	**0.98**
	Mean	0.53	0.50	0.55	0.47

Problem 3, focused on detecting semantic inconsistencies, yielded similar trends, with Llama-3-70B-Instruct showing the highest accuracy (0.92) and F1-score (0.89). The mean accuracy across all models for this problem was 0.65, with a precision of 0.61, recall of 0.53, and F1-score of 0.51.

For **Problem 4**, which addressed syntactic inconsistencies in RDF triples, Llama-3-70B-Instruct delivered the best results with an accuracy of 0.99, precision of 0.99, recall of 0.97, and F1-score of 0.98. The mean accuracy for this problem across models was 0.53, with a mean precision of 0.50, recall of 0.55, and F1-score of 0.47.

Figure 2 presents the mean metric values across all four problems, highlighting Llama-3-70B-Instruct as the top-performing model, with a mean accuracy of 0.93, precision of 0.89, recall of 0.95, and F1-score of 0.90. Mixtral-7B-Instruct and Gemma2-9B-Instruct had moderate overall results, with mean accuracies of 0.55 and 0.59, respectively. Mixtral-7B-Instruct exhibited a mean precision of 0.57, recall of 0.61, and F1-score of 0.50, while Gemma2-9B-Instruct achieved a mean precision of 0.60, recall of 0.56, and F1-score of 0.46. Bloom-176B had the lowest mean with an accuracy of 0.41, precision of 0.42, recall of 0.38, and F1-score of 0.36 across all problems.

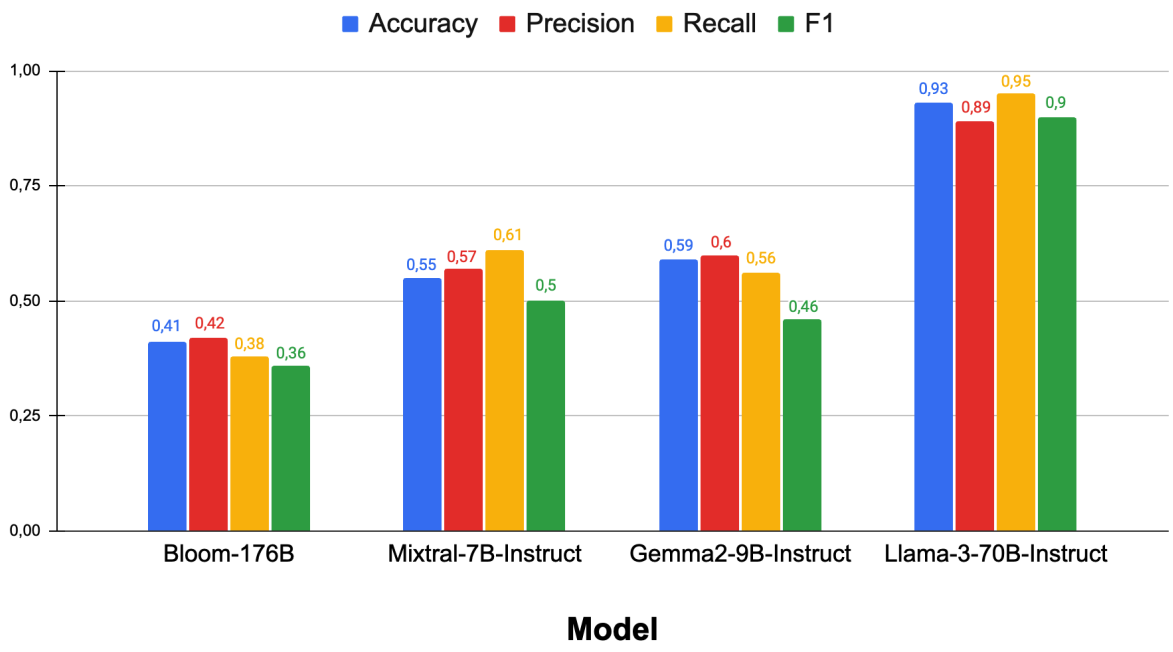

Figure 2: Summarization of the results achieved in the experimental evaluation. The x-axis represents the models. The y-axis represents values in the range [0,1] of each metric. The values shown are the mean values for each metric (accuracy, precision, recall, and F1) across all four problems. For example, the accuracy for Bloom-176B (0.41) was calculated by averaging the accuracy results obtained across the four problems, and similarly for the other metrics and models.