Evaluating LLMs in Ontology Transformation: Study on Property Chain Shortcutting

Kateřina Haniková[1,*], Vojtěch Svátek[1,*] and Ondřej Zamazal[1,*]

[1]*Prague University of Economics and Business, Czechia*

Abstract

Automated ontology transformation, which further informs the transformation of knowledge graphs referring to the ontology, focuses on finding occurrences of patterns that can be converted into alternative representations while preserving their semantic meaning. We have determined that this process can be divided into up to four phases in which LLMs can be employed: ontology match eligibility evaluation, naming of new ontology entities, knowledge graph match eligibility evaluation, and naming of new knowledge graph instances. In this paper, we investigate the performance of LLMs in the first two phases, on a small sample of transformation tasks in property chain shortcutting. We tested several LLMs and prompts for ontology transformation on a sample, assessed their results, and observed what kind of errors the models make. The results suggest that the quality of the naming in the transformed structure may be a reasonable indicator of the eligibility for transformation.

Keywords

Ontology transformation, entity naming, evaluation, eligibility, large language models

1. Introduction

While knowledge graphs (KG) are, in theory, much more flexible than traditional data representations such as relational tables, this phenomenon is not yet being sufficiently exploited. Applications normally consume the KG triples "as they are", or, when some restructuring (such as property chain shortcutting, addition of inverse links, or instantiation of resources to classes derived from their property values) is needed, it is accomplished in an application-dependent manner.

In this paper, we consider the transformation of the KG as a set of transparent operations to be carried out as a knowledge engineering activity and under the guidance of documented patterns. Unlike KG evolution, which mostly reflects changes in the world (or additional information acquired about it), in KG transformation we assume that the world has not changed, only the formal structures by which we model it have. The motivations why we are interested in such a 'meaning-preserving' KG transformation may vary, for example:

- We may want to directly connect entities that are too distant in the original representation; this comes into play, e.g., in visualizing large KGs or in automatically searching for association patterns under pattern size constraints (in KG mining tasks).
- We may need to comply with the preferences of a KG-processing tool for certain kinds of constructs – e.g., meta-model classes through property values to disentangle multi-hierarchies, or, conversely, construct new classes based on properties and their values.
- We may prefer to smoothly eliminate graph nodes whose identity became useless after they were encrypted for privacy reasons.

Transformations are first defined at the ontology level, using *ontology transformation patterns* [1]. Let us consider, for example, a situation in an ontology O in which a property p has domain X and range Y, and a property r has domain Y and range Z. This structurally matches the 'property chain shortcut'

ELMKE: Evaluation of Language Models in Knowledge Engineering 3rd Workshop co-located with ESWC 2026, Dubrovnik, Croatia

*Corresponding author.

katerina.hanikova@vse.cz (K. Haniková); svatek@vse.cz (V. Svátek); ondrej.zamazal@vse.cz (O. Zamazal)

https://nb.vse.cz/~svatek/ (V. Svátek); https://nb.vse.cz/~svabo/ (O. Zamazal)

0009-0009-7162-5925 (K. Haniková); 0000-0002-2256-2982 (V. Svátek); 0000-0002-7442-9016 (O. Zamazal)

ontology transformation pattern, which could then yield a new property s with domain X and range Z. The instantiated ontology transformation pattern for shortcutting the p/r path to s can then be propagated to a concrete KG G through an appropriate SPARQL UPDATE operation. O and G would thus conform to the 'dual pattern' of linked data modeling by Krisnadhi et al. [2].

The process leading to a change of a KG shape can, in fact, be decomposed into up to four phases, and LLMs can potentially be employed in all of them.

1. *Ontology* match *eligibility* evaluation: determining whether an ontology transformation pattern structurally matching (through its left-hand side) a certain ontology subgraph and if should indeed be applied to this subgraph or not.
2. New *ontology* entity *naming*: the ontology transformation always yields at least one new entity not yet having a name (textual label and/or human-readable IRI) and a description. The name should reflect the meaning of existing entities matched on the left-hand side of the transformation pattern; e.g., in the property chain shortcut example, the name of the new property s should be derived (whether using the same lexemes or through some conceptual abstraction and synonymy exploitation) from the chained properties p and r.
3. *KG* match *eligibility* evaluation: specifically for certain ontology entities, transformation of particular instance structures might not be desirable. For example, a shortcut of p and r by s should possibly not be made for triples apb, brc if a and c are already linked by a property (possibly from another ontology) whose meaning is equivalent to s.
4. New KG instance naming: the new KG instances may, in some contexts, also need to be equipped with human-readable labels derived from pre-existing entities (both ontology and instance KG ones).

In this paper, we focus on the ontology transformation phases, i.e., 1 and 2. Using LLMs in phase 2 has already been addressed by our previous research [3], where we provided an interactive *evaluation testbed* for LLMs (and their parameterization and zero- vs. few-shot scenarios) with respect to their fulfillment of the new ontology entity namer's role. However, no systematic experiment was performed and the eligibility problem was not taken into account, i.e., the (rather inadequate) assumption was that all syntactically matching cases would be eligible.

Compared to directly using LLMs to design an ontology from unstructured resources [4], the task is much more constrained. The transformation patterns correspond to a firm symbolic 'scaffold' along which the transformation can proceed. The eligibility assessment as a task for an LLM then may actually melt down to (binary or ordinal) *classification* task. From the perspective of evaluating the LLMs involved, this is much easier than evaluating generated content. However, an important aspect of the classification could be its *explanation*, at least for debugging purposes. We thus assume that the evaluation of LLMs for the transformation eligibility task should subsume both the (straightforward, automated) evaluation of classification accuracy and the (sophisticated, often manual) evaluation of the adequacy of the classification explanation generated by the LLM.

This paper presents the initial account of the transformation eligibility task with a rudimentary experiment whose evaluation so far consists only of qualitative remarks.

2. Methodology

We only focus on one of the previously defined transformation patterns[1]. The object property chain shortcut (CHASH) is shown in Figure 1. The pattern is about detecting chains of properties that can be shortened without losing any important information. This could be useful, for example, when we have some information in the chain that cannot be shared with other parties, such as trade secrets.

[1]Other patterns are described in more detail in previous work [1].

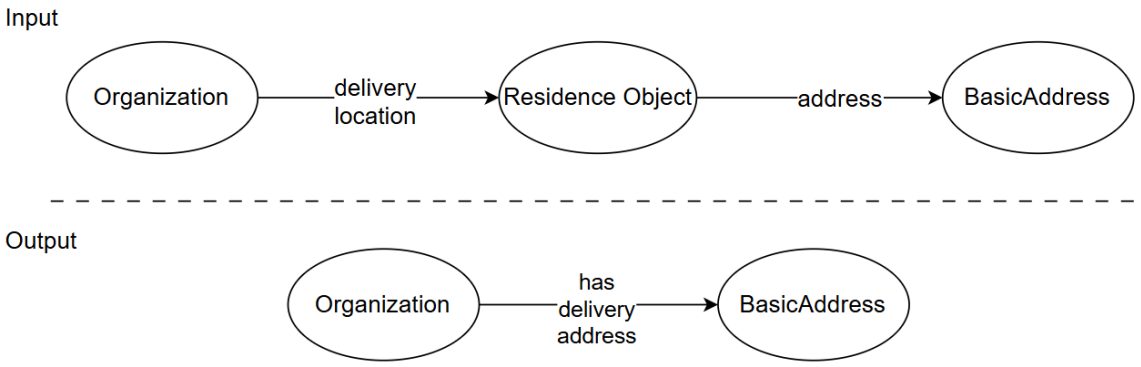

Figure 1: Application of the object property chain shortcut (CHASH) pattern

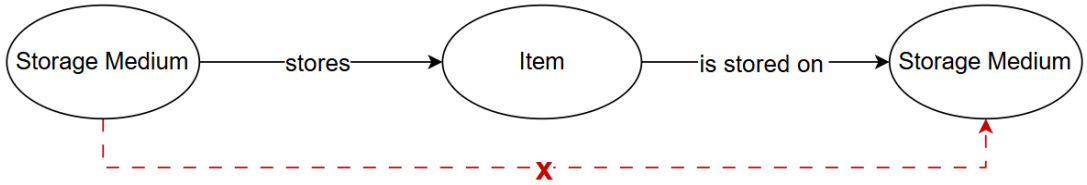

Figure 2: Illustration of the example not eligible for the CHASH pattern

2.1. Data collection and ad hoc exploration for CHASH eligibility

Our experiments were performed on ontologies from DBpedia Archivo.[2] We used the latest available collection of parsed ontologies as of October 2025. In total, there have been 1811 ontologies, of which we randomly selected 20%, i.e., 362. On this sample, we executed a SPARQL query[3] to identify the pattern. We adapted this query to retrieve only English labels for classes and properties, and then used it to randomly select at most five instances of the pattern from each ontology. Within the 362 ontologies, the CHASH pattern was identified in 71 of them. The SPARQL query retrieves everything that matches its pattern; however, the resulting instances are not always suitable candidates for the actual transformation. For example, we obtain matches like the one shown in Figure 2, which are clearly not suitable for the transformation for the following reasons:

1. the properties to be chained are inverse to one another,
2. the first one ('stores') is likely inverse-functional, while the second one ('is stored on') is likely functional.

Hence, the chained property would just trivially link an object to itself ('media stores what is stored on it'). Next, if we flipped the relations in the chain (putting the functional one first and the inverse-functional one second), the shortcut would become more meaningful (although possibly too abundant for being materialized in the KG), yielding the symmetric relationship between objects stored on the same media.

On the other hand, the same pattern of chaining a functional and inverse-functional property does not work well in the example in Figure 3. Here, the shortcuts would connect a position in a company to one of its employees, which is unlikely to be of any use.

In contrast, shortcutting two properties of which one is both functional and inverse functional seems promising, e.g., considering the chain of domain/range classes and properties *Meeting – meeting notes – Meeting Notes – contains action item – Action Item*, where a meeting could be directly connected to an action item that resulted from it, skipping the 'meeting notes' indirection. While in the so-far mentioned

[2]https://www.dbpedia.org/resources/archivo/

[3]The SPARQL query for the pattern detection is available at: https://github.com/Onto-DESIDE-VSE/TransformationPatterns/blob/main/experiments/CHASH.md.

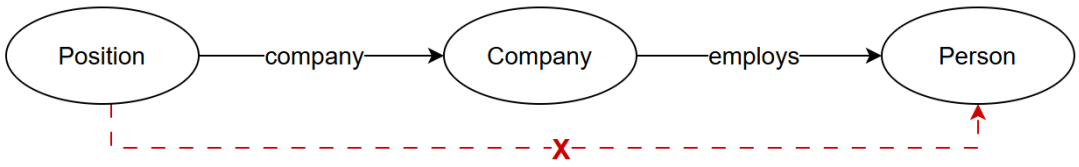

Figure 3: Illustration of a more complex example that is not eligible for the CHASH pattern

cases the axiomatization (functionality and inversion of properties) may provide a reasonable clue, there will also be cases when even two M:N properties do not represent a meaningful chain deserving a shortcut despite their shared range/domain class – especially, if the scope of the ontology is broad and the two properties belong to different 'spheres' of a highly general class such as *Person* or *Organization*.

Therefore, when making the transformation eligibility decision, particularly at the ontology level, applying formal domain-agnostic *logical* rules (which can be linked to the transformation patterns) is not sufficient. As humans, we need to apply common sense to decide whether inserting a property chain shortcut (or applying any other transformation) brings added value to the ontology and KG. Consequently, we may want to test the hypothesis that *large language models* can leverage the lexical labels of existing interconnected entities and reach a plausible conclusion about whether to apply a transformation.

So, we began manually annotating the dataset for CHASH pattern eligibility and held two group discussions with ontology experts and linguists to address some of the more complex examples (such as those labeled as "borderline" in Appendix B). Ultimately, we found that 28 ontologies in our dataset contain at least one meaningful occurrence of the object property shortcutting pattern.

2.2. Protocol of the experiment

Although a SPARQL query serves as a pattern detector, it cannot decide whether the transformation should actually be carried out. To overcome this limitation, we explored whether the decision process could be solved automatically with the help of LLMs. Earlier experiments had demonstrated that LLMs can be used as name generators for new instances that are created during transformation (for example, the new object property "has delivery address" in Figure 1) [1].

We started by having simple conversations with LLMs through their user interfaces (like ChatGPT or Gemini). It quickly became clear that these models could not straightforwardly judge whether a pattern was reasonable to use. Their responses varied depending on how the question was phrased. For instance, if we asked something with a positive framing, such as *"Can we shorten the following chain of properties?"*, they would usually answer yes. And when we challenged their response, they tended to adjust their answer accordingly. So, rather than directly asking whether the pattern should be applied, we first have the model generate the label, even if it does not make sense, and then let the LLM assess whether its own output is suitable for the transformation. This led us to the main question: *How can we evaluate whether the generated instances are eligible for transformation or not?*

Based on our manual analysis of the data, we know that the correct answer to the eligibility question is not always 0 or 1 (use or not to use). There are cases that depend on deep knowledge of the given ontology and on the context for which the pattern would be used. So, we decided to use the five-point Likert scale to evaluate the degree of eligibility.

The next step was to create a prompt. In previous experiments we used both textual prompts and a prompt containing a SPARQL query. The textual prompt was iteratively improved until we achieved satisfactory results. However, for our application, we have decided to use the SPARQL prompt (Appendix A). The main advantage of this prompt is that it can automatically adapt to different patterns, eliminating the need for a separate prompt for each pattern.

For these first experiments, we used the following models: ChatGPT and Copilot models through

their user interface, and a few Llama models (llama-3.3:70b-q4_k_m[4], llama-3.3-70b-instruct:q4[5] and deepseek-r1:70b-llama-distill-q4_K_M[6]) on our own server. The LLMs' task was to suggest a new property label and a corresponding Likert score (5 for the most eligible, 1 for the least), and we selected 12 examples from our annotated dataset; see Appendix B (2 eligible, 2 non-eligible, and 8 borderline – more complex for decision).

2.3. Post-evaluation of LLM results

The manual evaluation of the LLM performance focused on the following questions:

- Does a new shortcut property provide any added value to knowledge graphs based upon this ontology? Is it ontologically sound? If yes, how likely is it that it would indeed be used?
- Provided the property is sound as such: does the model suggest a name for it that truly expresses the meaning?
- Provided the name of the property expresses its meaning well: is it easy to parse for a human? Is it parsimonious and elegant?

The first question does not evaluate the LLM as such, but rather determines the context in which it must operate. It can be hypothesized that if the meaning of the shortcut is obscured or nonsensical then the generated name will not make sense either. The two remaining questions then constitute two possible levels of LLM name-generation performance quality.

By closely examining the results of LLM-based naming[7], some general observations can be made regarding the points of (significant) failure:

- Mismatch of the verbally suggested *domain/range* of the property. For example, an organization is linked to its abstract delivery address (bypassing the physical building under this address) by a property that the LLMs calls "delivers to", as if the organization in concern were supposed to deliver to its own address.
- *Loss of one segment* of the chain. For example, when generating a direct link from a person to a recommendation provided to that person's connection (i.e., another person) in a social network, the LLM proposes the name "recommended by", completely ignoring the "connection" relationship from the original chain.
- *Inverting* a relationship. For example, when shortcutting the properties "sub task of" and "has requirement", the LLM generates a name such as "has sub task requirement" (thus flipping the relationship between a subtask and a supertask).

Less significant flaws include:

- Generalizing the names of specific relationships to vague terms such as "related".
- Names that contain all tokens needed to preserve the information, but in an *ungrammatical* and/or *hard to parse* sequence, e.g. "market segment seller's market segment focus".
- Correct but *non-thrifty* structures resulting from literal copying of the names of original entities to the new name, e.g., "meeting notes contain action item" (even if the action items can be directly linked to the meeting from which they result, omitting the notes).
- Trivial violation of conventions, such as use of plural endings, such as "recommendations from connections".

[4] https://ollama.com/library/llama3.3:70b-instruct-q4_K_M
[5] https://ollama.com/library/llama3.3:70b-instruct-q4_0
[6] https://ollama.com/library/deepseek-r1:70b-llama-distill-q4_K_M
[7] All results at: https://github.com/Onto-DESIDE-VSE/TransformationPatterns/blob/main/experiments/LLMs%20experiments/Eligibility/analysis-of-llms-results.xlsx.

Table 1
LLMs aggregate results of eligibility

Model/Metric	Avg. Yes	Avg. No	Avg. Borderline
deepseek-r1:70b-llama-distill-q4_K_M	4,5	2,5	3
llama-3.3-70b-instruct:q4	1,5	2	3,125
llama-3.3:70b-q4_k_m	1,5	3	3,375
ChatGPT	5	3,5	3,5
Copilot	4,5	3,5	2,5

Besides label generation, the LLMs should also provide an output score on Likert's scale (1-5) indicating whether it makes sense to create the shortcut. The average numbers for each category (yes/ no/ borderline) are shown in Table 1. The results vary by category and are presented solely for the sake of demonstrating the evaluation methodology, given the low number of examples. The two eligible examples were evaluated as 5 by the Copilot model, while the two non-eligible examples were best detected by the model llama-3.3-70b-instruct:q4 (both examples received a score of 2). The borderline examples are the most complicated, and their proper eligibility evaluation would really depend on the needs of a particular application, but all models had an average close to 3 in this category. If we consider price and performance, we could say that the *deepseek-r1-distill-llama-70b:q4_k_m* model is best for deciding whether to create the shortcut. However, if we are looking for a model that would suggest the best labels, ChatGPT would be the best choice.

3. Conclusion

This paper introduced the task of ontology transformation, with particular emphasis on determining when a transformation is applicable and on evaluating the quality of the label proposed for the new entities. We provide the results of tiny experiments for a single transformation pattern: property chain shortcutting.

For practical purposes, we foresee an approach in which LLMs act as assistants, ranking results so that pattern instances that appear more suitable for transformation receive higher scores than those that are less appropriate. The final decision still rests with users and ontology experts; however, this method can support them by pre-processing and filtering large volumes of data.

Although the methodology of both the LLM prompting for the eligibility classification and for evaluating its performance (especially wrt explaining its classification decision) is now only tentative and with lots of blind spots, there already exists an implementation in an updated version of the PatOMat2[8] ontology transformation tool [5]. Hence, any gradually improved technique can be integrated into this tool and tested with users.

Declaration on Generative AI

During the preparation of this work, the author(s) used Writefull and Grammarly in order to: Grammar and spelling check. After using these tool(s)/service(s), the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] V. Svátek, O. Zamazal, K. Haniková, D. Chudán, M. J. Saeedizade, E. Blomqvist, Welcome, newborn entity! On handling newly generated entities in ontology transformation, in: Joint Proceedings of Posters, Demos, Workshops, and Tutorials of the 24th International Conference on Knowledge

[8] https://owl.vse.cz/patomat2/

Engineering and Knowledge Management (EKAW-PDWT 2024), CEUR-WS, 2025. URL: https://ceur-ws.org/Vol-3967/PD_paper_187.pdf.

[2] A. Krisnadhi, N. Karima, P. Hitzler, R. Amini, V. Rodríguez-Doncel, K. Janowicz, Ontology design patterns for linked data publishing, in: P. Hitzler, A. Gangemi, K. Janowicz, A. Krisnadhi, V. Presutti (Eds.), Ontology Engineering with Ontology Design Patterns - Foundations and Applications, volume 25 of *Studies on the Semantic Web*, IOS Press, 2016, pp. 201–232. URL: https://doi.org/10.3233/978-1-61499-676-7-201. doi:10.3233/978-1-61499-676-7-201.

[3] P. Vajdecka, V. Svátek, Testbed for evaluating llms on concept naming within ontology transformation, in: E. Curry, V. Presutti, J. P. McCrae, M. Alam, P. Colpaert, J. X. Parreira, D. Collarana, M. Sabou, A. Harth, P. Lisena (Eds.), The Semantic Web: ESWC 2025 Satellite Events - Portoroz, Slovenia, June 1-5, 2025, Proceedings, volume 15832 of *Lecture Notes in Computer Science*, Springer, 2025, pp. 162–166. URL: https://doi.org/10.1007/978-3-031-99554-5_30. doi:10.1007/978-3-031-99554-5_30.

[4] A. S. Lippolis, M. J. Saeedizade, R. Keskisärkkä, S. Zuppiroli, M. Ceriani, A. Gangemi, E. Blomqvist, A. G. Nuzzolese, Ontology generation using large language models, in: E. Curry, M. Acosta, M. Poveda-Villalón, M. van Erp, A. Ojo, K. Hose, C. Shimizu, P. Lisena (Eds.), The Semantic Web, 22nd European Semantic Web Conference, ESWC 2025, Portoroz, Slovenia, June 1–5, 2025, Proceedings, Part I, Springer Nature Switzerland, ????, pp. 321–341. doi:10.1007/978-3-031-94575-5_18.

[5] O. Zamazal, M. Ledvinka, V. Svátek, PatOMat2: A Tool for Pattern-Based Ontology Transformation using SPARQL, in: Joint Proceedings of Posters, Demos, Workshops, and Tutorials of the 24th International Conference on Knowledge Engineering and Knowledge Management (EKAW-PDWT 2024), CEUR-WS, 2025. URL: https://ceur-ws.org/Vol-3967/PD_paper_182.pdf.

A. SPARQL prompt for eligibility task

You have the following SPARQL update operation:

```
INSERT {
    ?newProp a owl:ObjectProperty;
    rdfs:domain ?a;
    rdfs:range ?c.
}
WHERE {
    ?p rdfs:domain ?a .
    ?p rdfs:range ?b .
    ?r rdfs:domain ?b .
    ?r rdfs:range ?c .
}
```

The examples of the names of the properties and the classes behind the variables are as follows:

```
?a = Person
?b = City
?c = State
?p = was born in city
?r = is located in
?newProp = was born in state.
```

Your task is, when I give you another set of properties and classes, to suggest a new label for the property ?newProp, and to give a number on Likert's scale (1-5), whether it makes sense to create this shortcut, or not. Consider the ontological structure and whether the new property would make sense in the real world.

These are properties and classes for your task:

```
?a =
?p =
?b =
?r =
?c =
```

In output, return only the comma-separated line per input: the number of input, its Likert score, and the new property label. No further output is allowed. The Likert score should reflect meaningfulness and allow ordering the examples in detail.

B. Selected examples for experiments

Table 2
Selected examples for experiments

ID	Example – the whole chain	Eligible
Ex1	Organization -> delivery location -> Residence Object -> address -> BasicAddress	yes
Ex2	Meeting -> Meeting Notes -> Meeting Notes -> Contains Action Item -> Action Item	yes
Ex3	Feature -> parent feature -> Feature -> feature -> Class	borderline
Ex4	Feature -> neighbour -> Feature -> map -> Map	borderline
Ex5	Task -> Sub Task Of -> Task -> Has Requirement -> Data Requirement	borderline
Ex6	Event -> eventPerson -> Person -> placeOfBirth -> Place	borderline
Ex7	Instrument -> instrument used in Science -> Science -> requires knowledge from -> Science	borderline
Ex8	Storage Medium -> stores -> Item -> is stored on -> Storage Medium	no
Ex9	Market Segment -> seller -> Market Seller Entity -> has market segment focus -> Market Segment	borderline
Ex10	Market Seller Entity -> hasMarketSegmentFocus -> Market Segment -> seller -> Market Seller Entity	borderline
Ex11	Access Condition Set -> has variable -> Variable -> has value -> Value	no
Ex12	Person -> connection -> Person -> recommendationsReceived -> Recommendation	borderline