Proceedings of Machine Learning Research vol 284:1–6, 2025 19th Conference on Neurosymbolic Learning and Reasoning

Validating Free Text Against N-Ary Knowledge Graphs

Jayavanth Shenoy, Jin-Ching Lim, Guha Jayachandran
{JAYAVANTH, JIN, GUHA}@ONAI.COM
Onai Inc.

Editors: Leilani H. Gilpin, Eleonora Giunchiglia, Pascal Hitzler, and Emile van Krieken

Abstract

The proliferation of generative AI models increases the need to validate free text assertions against repositories of information of known provenance, such as with knowledge graphs. This validation allows determination of which assertions are attested by whom with what confidence. We present a neurosymbolic approach that leverages information retrieval, embeddings, and an LLM to identify and corroborate assertions in text. Given a passage of text, our method identifies knowledge graph statements that support specific claims in the text, or identifies the absence of such support for specific claims. Unlike prior work, we are able to demonstrate support for n-ary relations (relations with qualifiers). We demonstrate and test the method using the Wikidata graph. We present the method's performance on various metrics, including identification accuracy of assertions and specificity and sensitivity of mappings.

1. Introduction

With the growth of text generation by both humans and machines, being able to validate the accuracy of the assertions has become essential. This is critical regardless of whether misinformation is intentional or unintentional. One of our company's efforts centers on better enabling the use of knowledge graphs based on data from various government agencies and other organizations. This notably includes the Proto-OKN graph(s) (National Science Foundation, 2023). The tool we describe in this paper utilizes neurosymbolic AI to process text – whether produced by an LLM or a human – and corroborate it against such knowledge graph(s). Our contributions include: a) Neurosymbolic approach to end-to-end free text validation against a knowledge graph (KG) with n-ary relations. b) Human-annotated dataset and analysis framework.

2. Methodology

We perform end-to-end validation of free text in three steps: **1. Triple Variations Extraction**: In this first step of the pipeline (Fig. 1(a)), we prompt an LLM to generate triples that represent all of the assertions in the inputted text passage. **2. Subject Identification**: For each distinct subject across the generated triples, we utilize a combination method for subject disambiguation, identifying the knowledge graph entity that corresponds to the given subject with the use of a text search, an embedding-based search, and an LLM query. **3. KG Matching**: Given a purported triple extracted from the text, we now determine whether the "fact" is present in the KG (Fig. 1(b)). We first obtain all the properties (relations) of subject, identified in the previous step. Then we utilize an LLM to determine which relation, if any, in the KG relations list of the subject is equivalent to the fact being validated.

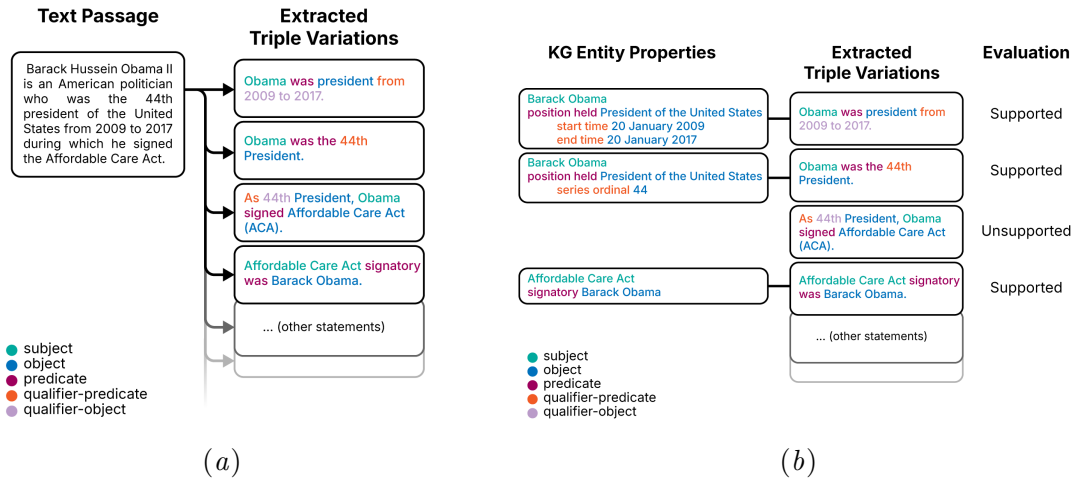

Figure 1: (a) Multiple variations of triples are generated to describe all the information in the text passage. (b) For a given extracted triple, a match is sought in the KG properties of the identified subject. Not all triples have a match; some are unsupported.

3. Experiments and Results

Setup: The evaluation data was constructed from the official Wikipedia dump (Wikimedia Foundation, 2023) by randomly selecting two or more introductory paragraphs of 138 text passages. Dataset metrics are listed in Appendix B. Trained humans evaluated the accuracy of results of each step of the method against the dataset, and their annotations about assertions, identified subjects, and triple matches were saved (total of 10,912 annotations). We used Nemotron-Ultra-253B (Bercovich et al., 2025), with bf16 quantization, as the LLM. We used Wikidata API for text-based entity search and Wikidata Embedding Project API for embedding-based entity search (Wikimedia Deutschland, 2025).

In triples extraction, we observed full coverage: extracted triples cumulatively represented all the assertions in the text. 99.22% of the extracted triples were found to exist in the text. The overall accuracy of subject disambiguation was 89.40%. KG matching had an overall accuracy of 94.48%. More detailed metrics for each step are listed in Appendix C.

4. Conclusion and Future Work

Our method to corroborate claims in free text against n-ary knowledge graph relations shows good performance in statement extraction, subject identification, and matching of claims. We evaluated our technique using the Wikidata graph. We believe the annotated dataset will be of use for efforts to develop or improve approaches to validation of free text against Wikidata. Ongoing work involves evaluating our technique with smaller LLMs, replacing triple variation generation with more sophisticated LLM-based semantic matching, resolving challenges such as implicit assertions and aliases, and handling multihop relations. We will evaluate the performance of the methods on additional KGs, including domain-specific KGs such as the graphs in Proto-OKN.

References

Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, et al. Llama-nemotron: Efficient reasoning models, 2025. URL https://arxiv.org/abs/2505.00949.

National Science Foundation. Proto-OKN collaboration: Facilitating the development and deployment of the first-ever prototype open knowledge network, 2023. URL https://www.proto-okn.net/.

Wikimedia Deutschland. Wikidata embedding project. https://www.wikidata.org/wiki/Wikidata:Embedding_Project, 2025. Accessed: 2025-06-06.

Wikimedia Foundation. Wikimedia downloads, 2023. URL https://dumps.wikimedia.org.

Appendix A. LLM Prompts

A.1. Statement Variation Generation Prompt

```
You are a knowledge graph expert. I would like you to extract wikidata
    -style n-ary triples that correspond to the text block I provide.
The triples describe the set of subjects, objects, predicates, and
    qualifiers wherever it makes sense to add qualifiers. Give me many
    different triples (plus qualifiers when it makes sense) that
    represent assertions in the text, and give me several variations of
     each triple. The triples you produce should comprehensively
    encompass all the information in the text. You should produce so
    many triples that there is overlapping information between them, as
     you are producing triples that represent different ways of
    structuring the information. The variations could be in different
    subjects/objects/predicates that are chosen or the way qualifiers
    are arranged that all might represent the same assertion.

Your output is a json array of triples. Each triple is a json object
    with the following structure:

{
  "subject": "Subject of the statement.",
  "predicate": "Predicate that describes the relationship.",
  "object": "Object or target of the predicate.",
  "qualifiers": [
    {
      "predicate": "Qualifier predicate (if any).",
      "object": "Qualifier object (if any)."
    }
  ]
}


Make sure that subject and object that you pick has a good chance of
    being an entity name. When the subject/object is a reference to an
```

```
    entity mentioned in a prior sentence, then use the context to
    determine the referred to entity and specify that as the subject/
    object. This means, for example, that instead of a pronoun like "he
    " or "she", you would put the person's name. Or instead of "a game"
     if that is a reference to a specific game mentioned in the
    preceding context, you would specify the particular game referenced
    .

Use surface forms only - no QIDs. The output should be only valid json
    . Do not include any code formatting (no triple backticks),
    explanations, or extra text. Do not output explanations or extra
    text, only the json.

Provide triples for ALL OF THE ASSERTIONS in this text, and remember
    to provide multiple variations.
```

A.2. Subject Disambiguation Prompt

```
{qid_list_str}
Which of the above list items is referenced by the below paragraph?
Just write the full QID starting with Q of the best match. No need to
    write your reasoning. If there is no good match in the list, just
    output NO MATCH.
Paragraph:
{paragraph}
```

A.3. KG Matching Prompt

```
You are an ontology expert. Please follow the instructions below.

Later I'm going to give you a json array that I'll call JSON_A. It has
     n-ary triple data about the subject {main_subject_QID}. Each
    element in the array specifies a triple object, predicate, and
    qualifiers (with the subject being {main_subject_QID}).

Here are the properties of the subject {main_subject_QID} from a
    knowledge graph. I'll call this JSON_B:

{json_2}

For each element in the JSON_A array, determine if there is a relation
     in JSON_B that has an equivalent meaning. Output an array of equal
     length as the JSON_A array. Element i of the outputted array
    should be as follows:

If a triple from JSON_B has the same meaning as element i of JSON_A
    array, then output that the JSON_B triple verbatim (including any
    qualifiers) as element i. (Never make up a triple that doesn't
```

```
    appear in JSON_B or blindly copy the element from JSON_A if it's
    not present in JSON_B.)
If no triples in JSON_B has an equivalent meaning to element i of
    JSON_A, then output the string "None".

To emphasize, each element of the output array should either be a
    triple that appears exactly in JSON_B or should be the string "None
    ". Never output an element from JSON_A if it's not present in
    JSON_B.

It is ok to output a given triple from JSON_B more than once (for it
    to be equivalent to more than one triple from JSON_A).

Note that hypernyms don't count as having equivalent meaning: A triple
     from JSON_B doesn't count as a match to the array element in
    JSON_A if it is making a less specific statement. (For example, a
    relation in JSON_B saying that a person was born in the United
    States doesn't have equivalence to a triple in JSON_A saying that
    the person was born in California.)

Return only valid json as the output. Do not include any code
    formatting (no triple backticks), explanations, or extra text.
Use only the data that I have provided you. Do not attempt to
    translate QIDs. Do not use QIDs. Output only the json array that I
    requested, do not include any explanations or other text.

Here is JSON_A:
{json_1}
```

Appendix B. Dataset

Table 1: Dataset statistics and annotation summary

Dataset Metric	**Count**
Number of paragraphs	138
Number of extracted statements	4,015
Number of valid statements	3,933
Annotation Statistics	
Statement validity annotations	4,015
Subject disambiguation annotations	3,983
Statement matching annotations	2,914
Total annotations	10,912

Appendix C. Evaluation Results

The overall accuracy of subject disambiguation was 89.40%. We identified that there was no match for the queried subject in the KG with an 89.47% precision. When there was an actual match for the subject, we identified it correctly with an accuracy of 88.64%.

KG matching had an overall accuracy of 94.48%. The method had a precision of 97.89% in determining that there was no corroboration for a statement in the KG. Hallucinations occur when a statement predicted as a match against a KG does not exist in the KG. The hallucinations rate—the ratio of the number of hallucinations to the total number of matches (correctly or incorrectly) generated—was 2.63%. A mismatch is defined as a triple from the KG identified as a match for an extracted triple which does not have an equivalent meaning to the extracted triple. The mismatch rate was 15.79%. Mismatches could be attributed to challenges such as outdated knowledge in the KG or text (for example, population figures); the correct match being a qualifier in the KG triple instead of being the primary triple; and the need to utilize multiple relations as described in the next section.

Table 2: Evaluation results

Evaluation Step	**Metric**	**Performance**
Triple Extraction	Precision	99.22% (3933/3964)
	Missing Assertions	0 per paragraph
Subject Disambiguation	Overall Accuracy	89.40% (3543/3963)
	No match Precision	89.47% (561/627)
	Accuracy when Match Present	88.64% (2982/3364)
KG Matching	Overall Accuracy	94.48% (2739/2899)
	No match Precision	97.89% (2227/2275)
	Hallucination rate	2.63% (16/608)
	Mismatch Rate	15.79% (96/608)