Enriching RDF Data with LLM based Named Entity Recognition and Linking on Embedded Natural Language Annotations

Michael Freund[1][0000−0003−1601−9331], Rene Dorsch[1][0000−0001−6857−7314], Sebastian Schmid[2][0000−0002−5836−3029], Thomas Wehr[2][0000−0002−0678−5019], and Andreas Harth[1,2][0000−0002−0702−510X]

[1] Fraunhofer Institute for Integrated Circuits IIS, Nürnberg, Germany
firstname.lastname@iis.fraunhofer.de
[2] Friedrich-Alexander-Universität Erlangen-Nürnberg, Nürnberg, Germany

Abstract. In this paper, we present a processing pipeline for transforming natural language annotations in RDF graphs into machine-readable and interoperable semantic annotations. The pipeline uses Named Entity Recognition (NER) and Entity Linking (EL) techniques based on a foundational Large Language Model (LLM), combined with a Knowledge Graph (KG) based knowledge injection approach for entity disambiguation and self-verification. Through a running example in the paper, we demonstrate that the pipeline can increase the number of semantic annotations in an RDF graph derived from information contained in natural language annotations. The evaluation of the proposed pipeline shows that the LLM-based NER approach produces results comparable to those of fine-tuned NER models. Furthermore, we show that the pipeline using a chain-of-thought prompting style with factual information retrieved via link traversal from an external KG achieves better entity disambiguation and linking than both a pipeline without chain-of-thought prompting and an approach relying only on information within the LLM.

Keywords: Natural Language Processing · KG enhanced LLM · RDF.

1 Introduction

Resource Description Framework (RDF) data is widely used in various domains where semantic information and knowledge needs to be stored in an interoperable and machine-readable format. For instance, in the Internet of Things (IoT) domain, through the World Wide Web Consortium's (W3C) Web of Things (WoT) Architecture [4], where RDF is used to describe the capabilities and APIs of sensors and actuators to create semantic interoperability [3]. Similarly, in archaeology [17], where RDF is used to describe discovered historical objects and their properties, such as where the objects were found and what materials the objects are made of, to create a comprehensive and interoperable knowledge base. RDF data typically consists of two parts: the main part, which is machine-readable information based on formal ontologies, and a smaller part,

which includes natural language annotations such as free-form text found in labels and comments.

When RDF data is created by non-experts who are unfamiliar with relevant ontologies and established RDF data modeling techniques, or by experts who assume that the data will only be accessed by humans, they often annotate some relevant information using solely natural language comments or labels for simplicity. However, only ontology-based information is machine-readable and interoperable. The information in natural language annotations embedded within RDF data can still be highly relevant and valuable. But, the embedded information often remains largely inaccessible to machines, due to the challenges of processing natural language, and to humans when the annotations are in a language they do not understand.

Recent advances in natural language processing (NLP) techniques, particularly with foundational large language models (LLMs), have significantly progressed the field [10]. Unlike traditional Named Entity Recognition (NER) models, which rely on extensive labeled training data adapted to specific application domains, large foundational LLMs can perform effectively without the need for domain-specific labeled data or fine-tuning [16]. Even without fine-tuning, foundational LLMs have been shown to outperform fine-tuned domain-specific models, for instance in the medical field [14].

Therefore, a pipeline that uses a foundational LLM for NER to extract relevant entities from natural language annotations in RDF data, in combination with Entity Linking (EL) to map the recognized entities to concepts in existing Knowledge Graphs (KGs) or ontologies, could be flexible and capable of making natural language information machine-accessible and interoperable. With that approach RDF data could be made more descriptive without the need for domain-specific model fine-tuning.

Extracting relevant information from text in RDF data is challenging due to the typically short and variable nature of embedded natural language annotations, which often consist of single-word labels or short sentences in comments. The inherent ambiguity of natural language, where a concept can be described in multiple ways and the meaning of a term can change based on context, further complicates this task. Unlike ontology-based information, which follows a formal schema, natural language annotations are unstructured, making systematic parsing and interpretation difficult.

In order to extract information from natural language annotations in RDF graphs, we propose such an extraction pipeline that uses LLM-based NER techniques to identify relevant entities in embedded natural language annotations. After entity recognition, the pipeline links the identified entities to concepts in pre-selected KGs and ontologies using a retrieval augmented generation (RAG) [5] approach with semantic similarity search based on vector embeddings to retrieve the URIs of identified concepts. By dereferencing these URIs, RDF documents with additional factual information are retrieved, which are then used for entity disambiguation and self-verification through knowledge injection techniques [7], mitigating the risk of hallucination. Finally, the pipeline generates based on the

identified entities RDF triples and integrates them into the original graph, converting the natural language information into ontology-based, machine-readable, and interoperable information.

The key contributions of our work are as follows:

- The introduction of a pipeline for transforming natural language annotations embedded in RDF graphs into additional semantic annotations using LLMs, NER, and EL.
- The introduction of a KG and knowledge injection-based method for entity disambiguation and self-verification.
- An evaluation of our LLM-based NER system compared to a traditional NER method, along with an assessment of the performance of the introduced pipeline compared to an LLM-only approach.

2 Running Example

To better illustrate the contributions of this work, we use the semantic description of an IoT Bluetooth Low Energy sensor following the Web of Things recommendation as a running example throughout this paper.

The RDF graph in turtle serialization describing the API of the sensor, is illustrated in Listing 1.1[3] and mainly uses the Thing Description ontology [1].

Listing 1.1. RDF graph describing the sensor introduced in the running example in Turtle serialization with relevant information contained in natural language text. Note that the contents of td:hasForm statements are omitted for simplicity.

```
[] a td:Thing ;
  td:title "Flower"@en ;
  td:description "Xiaomi Flower Care sensor in room 40."@en ;
  td:hasPropertyAffordance [
    td:name "temperature" ;
    jsonschema:readOnly "True"^^xsd:boolean ;
    td:description "In degrees Celsius."@en ;
    td:hasForm [ ... ] ] ;
  td:hasPropertyAffordance [
    td:name "humidity" ;
    jsonschema:readOnly "True"^^xsd:boolean ;
    td:description "The humidity value in %."@en ;
    td:hasForm [ ... ] ] .
```

From the semantic annotations, we can see that the sensor provides two property affordances: one named `temperature` and the other named `humidity`, and

[3] Well-known prefixes are omitted in all listings, but can be looked up on `http://prefix.cc/`

that both affordances are read-only. Additional information is contained in natural language via the three `td:description` (lines 3, 7, 12) and the two `td:name` annotations (lines 5, 10). Based on those annotations, we as humans can infer that the device is a Flower Care sensor manufactured by Xiaomi, deployed in room 40, and capable of measuring relative humidity in percent and temperature in degrees Celsius. But this information is currently only contained in natural language annotations and is therefore inaccessible to machines without the capability to process natural language and is therefore not interoperable.

In the following sections, we demonstrate how we extract the information using NER, associate and link the detected entities with selected vocabularies and KGs, disambiguate and verify the result, and generate triples to make the information machine-readable.

3 Related Work

Natural language processing is a difficult task that has challenged researchers for decades [12]. A subtask of the NLP domain is Named Entity Recognition, where the objective is to identify and classify relevant entities in text, where the meaning of entities may be context-dependent, requiring sequence labeling techniques [13]. Entity Linking complements NER by matching identified entities with existing concepts in ontologies or KGs. EL systems generate a list of potential candidates and perform entity disambiguation by considering the context in which the entity is used in the text. After successfully linking an entity to an associated concept, additional factual information can be used in further processing steps [20] and the entity can be integrated in the existing RDF graphs in the form of additional RDF triples.

With the recent increase in popularity of LLMs, such as the GPT series of models [6], and their ability to process natural language text [26], researchers have investigated the usability of LLMs for NER tasks. Wang et al. [23] explored the application of LLMs to NER tasks by introducing an approach called GPT-NER. The GPT-NER approach transforms the sequence labeling task into a generation task to exploit the strengths of LLMs. To address issues such as hallucination and overprediction when no entities are detected, a simple self-verification step is included, where the LLM is prompted again and asked to verify the given label. Monajatipoor et al. [11] used LLMs for a specialized NER task in the biomedical domain and found that providing relevant in-context examples via the input prompt is key for good performance.

Our work builds on the insights of the two LLM-based NER approaches and refines them by additionally using entity linking to identify the entities corresponding concepts in selected ontologies and KGs in order to leverage the factual knowledge through knowledge injection [7], where relevant information contained in a KG is provided as additional input to the LLM in order to improve the quality of the generated output. The introduced pipeline uses knowledge injection in an entity disambiguation and self-verification step.

To interact with LLMs, both approaches use so-called prompt engineering techniques, which involve natural language inputs that define what the LLM should do. A commonly used basic prompt format consists of an *instruction*, a *context*, and an *input text*. The instruction provides a general task description, the context offers few-shot examples, and the input text is the text that needs to be processed [15]. The quality of the prompt can in general impact the performance of the LLM in the desired task [22]. To solve more complex tasks, a more advanced chain-of-thought prompt [24] can be used, in which several intermediate steps of reasoning are used to reach the final goal.

The two previously introduced NER approaches use a simple prompt consisting of instruction, context, and input text. In contrast, we use chain-of-thought prompting in our pipeline.

To counter the tendency of LLMs to generate irrelevant or false outputs, known as hallucinations, a widely used approach is retrieval augmented generation [5]. The RAG method involves retrieving related information through dense vector similarity search and incorporating it as additional input during inference, i.e., injecting additional knowledge. Matsumoto et al. [8] implemented RAG in combination with KGs in a framework called KRAGEN. The authors embedded the entire KG, including all entities and relations, into a vector database to retrieve the factual information during inference.

In contrast, our approach focuses on embedding only the corresponding URIs of entities in the KG and using link traversal to retrieve the additional information, allowing for greater flexibility and precise linking to identified entities.

4 Enriching RDF Data based on Natural Language Annotations

Our processing pipeline to enrich RDF data based on embedded natural language annotations is illustrated in Fig. 1. Enriching RDF data with additional semantic information extracted from natural language annotations helps to improve the understandability and interoperability of the data, making it more useful for various applications in research [19] and industry [2]. The proposed annotation pipeline consists of four sequential steps: `Named Entity Recognition`, `Entity Linking`, `Disambiguation and Self-Verification`, and `Triple Generation`. The steps are discussed in detail below.

Named Entity Recognition The first step in the proposed processing pipeline is the recognition of relevant named entities. We start by retrieving the initial RDF data and extracting natural language annotations, such as labels, comments, or descriptions, using SPARQL queries. The extracted natural language annotations form our input set, which means that no duplicate entries are processed. Next, we use a foundational large language model to perform the NER task. The use of a foundational LLM-based NER approach allows us to bypass the need to fine-tune traditional NER models for entity recognition across various domains, which typically requires the manual generation of a significant number of labeled examples used as training data [18]. To interact with and

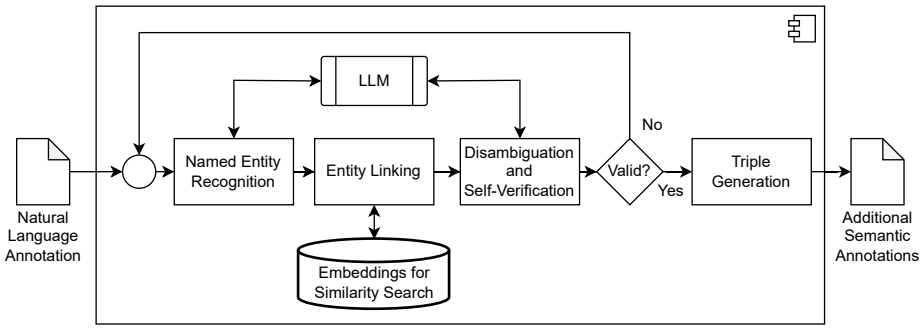

Fig. 1. The processing pipeline for transforming natural language annotations embedded in RDF data into semantic annotations. Key steps and data flows are illustrated.

instruct the LLM, we create a template that implements a prompting strategy based on chain-of-thought prompting, where information and definitions about the NER task and the target domain are introduced, followed by a classical prompt containing a task description, a small set of few-shot examples, and the natural language annotation to be classified. In addition, we provide a clear specification of the expected response format, which improves the consistency of the LLM output and eases the overall parsing of the generated output, allowing the integration of the output information into further steps of the processing pipeline. For each group of named entities that the processing pipeline needs to recognize, we manually create a template that implements the chain-of-thought prompting for the specific domain. Groups of named entities include units of measurement in text (e.g., degrees Celsius) or symbolic form (°C), observable properties (e.g., temperature), or location-specific entities (e.g., room 40). Finally, after recognizing the named entities in a natural language annotation, we task the LLM to verify that the discovered entities have been assigned the correct label in the context of the natural language annotation, similar to the validation approach introduced by GPT-NER [23].

Example 1. In the context of the running example, the input set I containing all natural language annotations is defined as $I = \{$"Xiaomi Flower Care sensor in room 40.", "temperature", "In degrees Celsius.", "rel . humidity", "The rel . humidity value in %." $\}$. The NER step processes the input set and generates the output set R containing corresponding recognized entities where $R = \{$ "Xiaomi", "room 40", "temperature", "degrees Celsius", "humidity", "%"$\}$

Entity Linking The second step of the processing pipeline is entity linking. During the EL process, we use the named entities identified in the NER step to construct entity embeddings using a pre-trained deep learning model. The embeddings are low-dimensional, dense vector representations that encapsulate the semantic and syntactic properties of the named entities [21]. By calculating

semantic similarity scores between the embeddings of the recognized named entities and those of pre-selected domain-specific concepts from KGs or ontologies, we are able to generate a list of potential candidates to which a named entity might be related. The candidate list may have zero entries if no similar concept exists in the pre-selected KGs or ontologies, or only one entry if only one similar concept is found within a given distance. The next task is to disambiguate and select the most appropriate candidate from the list, taking into account the context provided by the original natural language annotation. This disambiguation is done in the next step of the pipeline.

Example 2. The entity linking step uses the set R as input and transforms R into a set E, which contains, for each detected named entity, a candidate list of potential corresponding ontology or KG entries. In the running example, we recognize entries from Wikidata[4], QUDT[5], and a local floor plan KG[6].

Only the named entity for humidity produced a candidate list containing more than one entry, since there exist entries for *relative humidity* and the absolute humidity simply called *humidity*. Note, that if the candidate list contains only one entry, that entry is directly added to the set E instead of the list. Therefore, the set E is given by $E = \{$"wd:Q1636958", "ex:room40", "wd:Q11466", "qudt:DEG_C", ["wd:Q2499617", "wd:Q180600"], "qudt:PERCENT"$\}$.

Disambiguation and Self-Verification After performing the linking step, the next stage of the processing pipeline disambiguates candidates when more than one potential match is found. If only one candidate is present in the list, the pipeline verifies the result to ensure the linked entity is meaningful.

If more than one possible candidate is discovered during the entity linking step, the pipeline first retrieves relevant RDF documents containing definitions and information about the potential candidates using link traversal starting from the URIs identified in the EL step. The factual information contained in the retrieved RDF documents can then be provided as additional input to the LLM. Specifically, we inject the additional factual knowledge from the RDF data into the prompt, along with the recognized entity and the original natural language sentence. The task of the LLM is to verify that the identified concept, based on the corresponding definition in the RDF document, is correct for the recognized entity in the context of the sentence and to explain the reasoning steps that lead to the decision. In a second step, the original task and the generated reasoning output are fed back to the LLM, which is tasked with generating a final decision if the concept is correct and outputting either `yes` or `no`. This approach allows the LLM to disambiguate entries in the candidate list using external information that may change over time and was not available during training using a chain-of-thought prompting style.

The disambiguation approach can also be seen as self-verification, where additional external knowledge is added as context for the LLM to determine if the

[4] prefix wd: `https://www.wikidata.org/wiki/`
[5] prefix qudt: `https://qudt.org/vocab/unit/`
[6] prefix ex: `https://example.com/`

entity is being used correctly. Therefore, when the candidate list contains only one entry, we apply the same steps as in the disambiguation approach to verify that the candidate is used in the correct context.

If the results are not as expected, or if the candidate list cannot be disambiguated or verified, we restart the whole pipeline, repeating the previous entity recognition and linking steps, and providing the error information as additional input in the prompt.

Example 3. The Disambiguation and Self-Verification step uses the set E as input and produces the disambiguated and verified set E'. In the running example, only one ambiguity occurred, in the case of humidity. The original natural language annotation is `The humidity value in %.` and the recognized entity is `humidity`. The candidate list contains `wd:Q180600` for `humidity` and `wd:Q2499617` for `relative humidity` with the following natural language definitions extracted from the RDF data:

1. `humidity`: amount of water vapor in the air
2. `relative humidity`: ratio of the partial pressure of water vapor in humid air to the equilibrium vapor pressure of water at a given temperature

For each candidate, the original natural language annotation, the recognized entity, and the RDF data containing, among other things, the natural language definition presented above are then fed into the LLM, which is tasked with deciding if the definition is the correct one in the context of the natural language annotation.

The LLM determines that in this context, `relative humidity` is the correct definition because relative humidity is a ratio, and ratios are measured in percent. The self-verification process did not produce any errors for all other entities.

Therefore, the resulting set E' is given as $E' = \{$"wd:Q1636958", "ex:room40", "wd:Q11466", "qudt:DEG_C", "wd:Q2499617", "qudt:PERCENT"$\}$.

Triple Generation The final step in the processing pipeline is to generate RDF triples consisting of a *subject*, *predicate*, and *object* for insertion into the existing RDF graph, thereby making the information extracted from natural language annotations machine-readable and interoperable. The object position of the new triple contains the result of the NER and EL processes described in the previous steps, where named entities were identified and linked to corresponding entities in existing ontologies and KGs.

To find an appropriate predicate for the identified object, we use a pre-defined hashmap that maps different types of identified entities to their corresponding predicates. The use of a hashmap ensures that the relationships between entities are correct and stored in a scalable way. Additionally, the mapping can be modified and adjusted to include other ontological terms without the need to retrain a statistical model. For the subject position of the new triple, we reuse the same subject as in the original natural language annotation, preserving the context of the information.

Listing 1.2. RDF graph describing the sensor, with additional triples generated based on the embedded natural language annotations.

```
1  @prefix ex: <https://example.com/> .
2
3  [] a td:Thing ;
4    td:title "Flower"@en ;
5    td:description "Xiaomi Flower Care sensor in room 40."@en ;
6    schema:manufacturer wd:Q1636958 ;
7    schema:location ex:room40 ;
8    td:hasPropertyAffordance [
9      td:name "temperature" ;
10     jsonschema:readOnly "True"^^xsd:boolean ;
11     sosa:observes wd:Q11466 ;
12     td:description "In degrees Celsius."@en ;
13     qudt:unit qudt:DEG_C ;
14     td:hasForm [ ... ] ] ;
15   td:hasPropertyAffordance [
16     td:name "humidity" ;
17     jsonschema:readOnly "True"^^xsd:boolean ;
18     sosa:observes wd:Q2499617 ;
19     td:description "The humidity value in %."@en ;
20     qudt:unit qudt:PERCENT ;
21     td:hasForm [ ... ] ] .
```

Example 4. The Triple Generation step uses E' as input and generates the set of all output triples T. For instance, consider the first entry in E' where the NER step detected `Xiaomi` with the label `manufacturer`. The linking step linked the entity to `wd:Q1636958`, and the disambiguation and self-verification step confirmed the correctness. Therefore, we know that a manufacturer has been recognized, and we look up the predicate associated with the label `manufacturer` in the hashmap, which returns the property `schema:manufacturer`. The original natural language annotation was associated with a blank node of type `td:Thing`, so the new triple will also be associated with the same blank node. The generated triple $t_1 \in T$ is therefore given as $t_1 =$ (`[]`, `schema:manufacturer`, `wd:Q1636958`).

After all additional triples have been generated, all elements of the set T are added to the RDF graph, as shown in Listing 1.2. The information previously contained only in natural language annotations is now available as machine-readable and interoperable semantic annotations (lines 6, 7, 11, 13, 18, 20), resulting in a much more expressive RDF graph overall.

5 Empirical Evaluation

We evaluated the introduced pipeline using the Gemma2 27B language model which is part of Googles Gemma family of language models [9] and we link found entities against the Wikidata KG, and classes in the QUDT ontology.

In the empirical evaluation, we aim to answer the following two research questions:

- **R1:** How accurate is the foundational LLM-based NER approach for extracting entities from RDF graphs with natural language annotations compared to the classical approach of fine-tuning a domain-specific NER model?
- **R2:** How accurate is the pipeline using chain-of-though prompting in identifying and linking correct entities compared to the pipeline without chain-of-though prompting and to an approach using only the LLM?

All results and scripts to reproduce the evaluation can be found on GitHub[7].

5.1 R1: Comparison of LLM-based NER with Existing Approaches

In our comparative analysis, we focus on the recognition of units of measure in textual form (e.g., degrees Celsius) and symbolic form (e.g., °C). In addition, we recognize observable properties (e.g., temperature). We evaluated our proposed pipeline against a fine-tuned English language NER model[8]. The model was fine-tuned using the spaCy[9] framework on a synthetic data corpus specific to the IoT and WoT domains, generated using the llama3 70B language model based on a small hand-crafted dataset [25]. The corpus includes three SI base units with corresponding observed properties (time, length, and mass) and three derived SI units with corresponding observed properties (temperature, frequency, and acceleration), as well as percent as a pseudo-unit for dimensionless ratios such as the observed property relative humidity. Our corpus contains a total of 5,769 entries. A training script and the data corpus are available on GitHub[10]

For the comparison with the fine-tuned NER model and the pipeline, we use a dataset from the IoT/WoT domain. The dataset consists of a total of 69 RDF graphs describing APIs of IoT devices. Of these, 68 RDF graphs come from the W3C WoT TD implementation report[11] and are based on APIs of working systems created by contributing organizations such as Siemens AG, Intel, and Oracle. In addition, one RDF graph was manually created describing the interface of a Xiaomi Flower Care sensor used in the running example. The 69 RDF graphs contain a total of 620 natural language annotations in the form of `td:name` or `td:description` of which 44 contain references to units or observed properties.

[7] `https://github.com/FreuMi/ner_pipeline`
[8] `https://spacy.io/models/en#en_core_web_lg`
[9] `https://github.com/explosion/spaCy`
[10] `https://github.com/FreuMi/NER_Training`
[11] `https://w3c.github.io/wot-thing-description/testing/report.html`; available as a single RDF file at `https://www.vcharpenay.link/talks/td-sem-interop.html`.

Table 1. Comparison of classical fine-tuning and LLM-based NER approaches. The average runtime per annotation is reported in seconds.

	Classical Approach	**LLM-based Approach**
True Positives	33	40
False Positives	2	9
False Negatives	11	4
F1 Score	0.84	0.86
Avg. Runtime [s]	0.009	27.9

The results of the evaluation in Table 1 show that the foundational LLM-based NER approach is able to correctly detect 40 unit and observed property references in the annotations and reaches a similar F1 score as the classical approach of fine-tuning a NER model. However, the results also show that the LLM-based NER approach produces more than four times as many false positives as the classical approach, but is able to detect more true positives. The difference in true positives is likely explained by a lack of training data for the classical approach. Adding even more labeled training data to the corpus could allow the model to generalize better and recognize more entities in complex scenarios, such as those involving different contexts or rare entity types.

Overall, the LLM-based NER approach achieves comparable results to a classical fine-tuned NER system, but requires additional processing of the detected entities to reduce the number of false positives. Additionally the runtime of the LLM-based approach is much higher than the classical approach due to higher computational requirements.

5.2 R2: Pipeline Evaluation

To evaluate the effectiveness of our proposed pipeline, which integrates NER, RAG-based entity linking, and LLM-based disambiguation with self-verification through chain-of-thought prompting, we conducted a comparison with two alternative approaches. The first comparison is with a pipeline implementation that uses similar techniques except for the chain-of-thought prompting style. The second comparison is with an implementation that uses prompting methods that rely solely on the LLM's internal data and information to identify and validate detected entities.

For the evaluation, we used a dataset consisting of 50 manually generated example natural language annotations focusing on units of measurement with ambiguities, inspired by sensor data sheets. The task is to identify the name of the unit of measurement entity based on the symbolic unit representation in the context of the sentence. The dataset is publicly available on GitHub[12].

An example from the dataset where disambiguation is needed is the natural language annotation *Time measured in S.* Here, the unit stands for seconds, even though the letter `S` is capitalized. The `S` does not stand for the unit Siemens,

[12] `https://github.com/FreuMi/ner_pipeline/tree/main/evaluation/dataset`

which also uses the letter `S` to describe electrical conductance, because that interpretation does not make sense in the context of the natural language annotation.

Another example where self-verification is required, is the natural language annotation *The length l is 10 m.* The NER system might detect `l`, which typically stands for liters, and `m`, which typically stands for meters, as symbolic units. However, in the context of this annotation, only the unit `m` is relevant, since the letter `l` is simply the name for a certain length and there is no reference to the unit liters.

Table 2. Comparison of the pipeline with chain-of-thought (CoT), the pipeline without CoT, and an approach using an LLM-only method. The average runtime for each approach per annotation is reported in seconds.

	Pipeline w/ CoT	**Pipeline w/o CoT**	**LLM only**
True Positives	39	40	41
False Positives	3	12	25
False Negatives	2	1	0
F1 Score	0.94	0.86	0.77
Avg. Runtime [s]	326.8	42.7	21.3

The results in Table 2 show that the LLM-only approach achieves an F1 score of 0.77 and successfully detects all possible entities, i.e. the 41 true positives. However, it fails to filter out entities that do not fit the context of the sentence, resulting in a high number of false positives.

The pipeline approach without chain-of-thought prompting achieves an F1 score of 0.86, with approximately half the number of false positives compared to the LLM-only approach, while maintaining an almost similar number of 40 true positives. These results suggest that the inclusion of external factual information in the form of RDF data helps the LLM to make more accurate decisions about whether the correct entity has been identified.

Finally, the pipeline approach with chain-of-thought prompting achieves the highest F1 score of 0.94, providing an additional improvement of approximately 0.08 over the pipeline approach without chain-of-thought prompting. This approach further reduces the number of false positives, but detects almost the same number of 39 true positives compared to the 40 and 41 true positives of the previous approaches. These results suggest that the combination of external knowledge and the chain-of-thought reasoning capabilities of the LLM improves its ability to determine whether an entity is correctly used in a sentence.

Overall, the chain-of-though based processing pipeline achieves the best results in our evaluation, but also has the highest average runtime, which is about 7 times slower than the pipeline without chain-of-thought prompting and about 15 times slower than the LLM-only approach.

6 Conclusion and Future Work

In this paper, we presented a processing pipeline for improving the machine-readability and interoperability of RDF data by extracting relevant information contained only in natural language annotations to generate semantic annotations. The pipeline employs LLM-based named entity recognition and semantic similarity-based entity linking. To disambiguate when multiple linkable candidates are found during the EL step, the pipeline uses the context, i.e., the natural language annotation, the candidate, and the RDF definition of the candidate retrieved from the ontology or KG using link traversal as input to the LLM to evaluate if the entity makes sense in the given context using chain-of-though prompting. If only one entity is found, the pipeline uses the same approach as for disambiguation by injecting the detected factual information as input to the LLM to verify the results. In our evaluation, we found that the foundational LLM-based NER approach performs on par with the traditional approach of fine-tuning an NER model for a specific domain. Additionally, we demonstrated the effectiveness of the pipeline using entity linking and the chain-of-though based disambiguation and self-verification steps, compared to a pipeline implementation without chain-of-though prompting and to an implementation relying only on the LLM without external information.

Future work will focus on further optimization of the pipeline, such as exploring the performance of other LLMs, especially smaller models with reduced computational requirements, to reduce the processing time and also enable potential deployment of the pipeline at the network edge in an IoT context.

Acknowledgments. This work was partially funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK) through the Antrieb 4.0 project (Grant No. 13IK015B) and the MANDAT project (Grant No. 16DTM107A).

References

1. Charpenay, V., Käbisch, S.: On modeling the physical world as a collection of things: The w3c thing description ontology. In: European Semantic Web Conference. pp. 599–615. Springer (2020)
2. Freund, M., Rott, J., Dorsch, R., et al.: FAIR Internet of Things Data: Enabling Process Optimization at Munich Airport. In: European Semantic Web Conference. Springer (2024)
3. Kaebisch, S., McCool, M., Korkan, E., Kamiya, T., Charpenay, V., Kovatsch, M.: Web of Things (WoT) Thing Description 1.1. `https://www.w3.org/TR/wot-thing-description/` (2023)
4. Lagally, M., Matsukura, R., McCool, M., et al.: Web of Things (WoT) Architecture 1.1. `https://www.w3.org/TR/wot-architecture/` (2023)
5. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., et al.: Retrieval-augmented generation for knowledge-intensive nlp tasks. Advances in Neural Information Processing Systems **33**, 9459–9474 (2020)

6. Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., et al.: Language models are few-shot learners. arXiv preprint arXiv:2005.14165 **1** (2020)
7. Martino, A., Iannelli, M., Truong, C.: Knowledge injection to counter large language model (llm) hallucination. In: European Semantic Web Conference. pp. 182–185. Springer (2023)
8. Matsumoto, N., Moran, J., Choi, H., Hernandez, M.E., Venkatesan, M., Wang, P., Moore, J.H.: Kragen: a knowledge graph-enhanced rag framework for biomedical problem solving using large language models. Bioinformatics **40**(6) (2024)
9. Mesnard, T., Hardin, C., Dadashi, R., et al.: Gemma: Open models based on gemini research and technology. arXiv preprint arXiv:2403.08295 (2024)
10. Min, B., Ross, H., Sulem, E., Veyseh, A.P.B., Nguyen, T.H., Sainz, O., Agirre, E., Heintz, I., Roth, D.: Recent advances in natural language processing via large pre-trained language models: A survey. ACM Computing Surveys **56**(2), 1–40 (2023)
11. Monajatipoor, M., Yang, J., Stremmel, J., Emami, M., Mohaghegh, F., Rouhsedaghat, M., Chang, K.W.: Llms in biomedicine: A study on clinical named entity recognition. arXiv preprint arXiv:2404.07376 (2024)
12. Nadkarni, P.M., Ohno-Machado, L., Chapman, W.W.: Natural language processing: an introduction. Journal of the American Medical Informatics Association **18**(5), 544–551 (2011)
13. Nasar, Z., Jaffry, S.W., Malik, M.K.: Named entity recognition and relation extraction: State-of-the-art. ACM Computing Surveys (CSUR) **54**(1), 1–39 (2021)
14. Nori, H., Lee, Y.T., Zhang, S., Carignan, D., Edgar, R., Fusi, N., King, N., Larson, J., Li, Y., Liu, W., et al.: Can generalist foundation models outcompete special-purpose tuning? case study in medicine. arXiv preprint arXiv:2311.16452 (2023)
15. Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., Wu, X.: Unifying large language models and knowledge graphs: A roadmap. IEEE Transactions on Knowledge and Data Engineering (2024)
16. Qin, C., Zhang, A., Zhang, Z., et al.: Is chatgpt a general-purpose natural language processing task solver? arXiv preprint arXiv:2302.06476 (2023)
17. Rantala, H., Ikkala, E., Rohiola, V., et al.: Findsampo: A linked data based portal and data service for analyzing and disseminating archaeological object finds. In: European Semantic Web Conference. pp. 478–494. Springer (2022)
18. Satheesh, K., Jahnavi, A., Iswarya, L., Ayesha, K., Bhanusekhar, G., Hanisha, K.: Resume ranking based on job description using spacy ner model. International Research Journal of Engineering and Technology **7**(05), 74–77 (2020)
19. Scheffler, M., Aeschlimann, M., Albrecht, M., et al.: FAIR Data Enabling New Horizons for Materials Research. Nature **604**(7907), 635–642 (2022)
20. Sevgili, Ö., Shelmanov, A., Arkhipov, M., et al.: Neural entity linking: A survey of models based on deep learning. Semantic Web **13**(3), 527–570 (2022)
21. Shen, W., Li, Y., Liu, Y., et al.: Entity linking meets deep learning: Techniques and solutions. IEEE Transactions on Knowledge and Data Engineering **35**(3), 2556–2578 (2021)
22. Wang, S., Zhao, Z., Ouyang, X., Wang, Q., Shen, D.: Chatcad: Interactive computer-aided diagnosis on medical image using large language models. arXiv preprint arXiv:2302.07257 (2023)
23. Wang, S., Sun, X., Li, X., et al.: Gpt-ner: Named entity recognition via large language models. arXiv preprint arXiv:2304.10428 (2023)
24. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Advances in neural information processing systems **35**, 24824–24837 (2022)

25. Whitehouse, C., Choudhury, M., Aji, A.F.: Llm-powered data augmentation for enhanced cross-lingual performance. arXiv preprint arXiv:2305.14288 (2023)
26. Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., Zhong, S., Yin, B., Hu, X.: Harnessing the power of llms in practice: A survey on chatgpt and beyond. ACM Transactions on Knowledge Discovery from Data **18**(6), 1–32 (2024)