# Extensive Benchmark of Frugal Encoder-Decoder Language Models for Datatype Properties Extraction and RDF Knowledge Graph Generation

Ringwald Célian [a,*], Gandon Fabien [a], Faron Catherine [a], Michel Franck [a] and Abi Akl Hanna [a,b]
[a] *Inria, Université Côte d'Azur, CNRS, I3S, France*
*E-mail: firstname.lastname@inria.fr*
[b] *Data ScienceTech Institute, France*

**Abstract.** The choice made for representing the inputs and outputs of generative pre-trained language models (PLMs) can impact their fine-tuning on a new task. This article focuses on the fine-tuning and linearization process to generate facts extracted from text. On a restricted relation extraction (RE) task, we challenged five encoder-decoder models including BART, T5, CodeT5, FlanT5 and PileT5 by fine-tuning them on 13 linearization variations, including RDF standard syntaxes and variations thereof. Our benchmark covers the validity of the produced triples, the model's performance, the training behaviour and the resources needed. We show these PLMs can learn some syntaxes more easily than others, and we identify a promising "Turtle Light" syntax supporting the quick and robust learning of the RE task.

Keywords: Data extraction, RDF, Linearization, Language Model

## 1. Introduction

Relation extraction (RE) – the task of retrieving relations from unstructured text – was drastically improved recently by two main changes: (1) the construction of massive corpora aligning texts and facts from knowledge graphs (KG), using distant supervision [65] e.g. Wikipedia articles with corresponding Wikidata [23, 79] or DBpedia [9, 15] subgraphs, and (2) the availability of pretrained language models that can be used or adapted to solve a vast number of NLP tasks.

However, large KGs such as Wikidata and DBpedia may be affected by several quality issues: they could contain wrong facts, could store inconsistent data, be incomplete or outdated [21, 64, 81, 82]. In this context, extracting information from Wikipedia to fill the gaps in KGs is critical, as neither DBpedia nor Wikidata directly rely on the text of the articles. Aside from that, large language models, exclusively relying on decoder-only Transformer architectures, demonstrated interesting few-shot capacities. Nonetheless, they only marginally outperform finetuned encoder-decoder-based models on the Relation Extraction task, at the cost of significant engineering costs, including integrating RAG systems, prompt design for the few-shot strategy, and higher adaptation costs incurred during

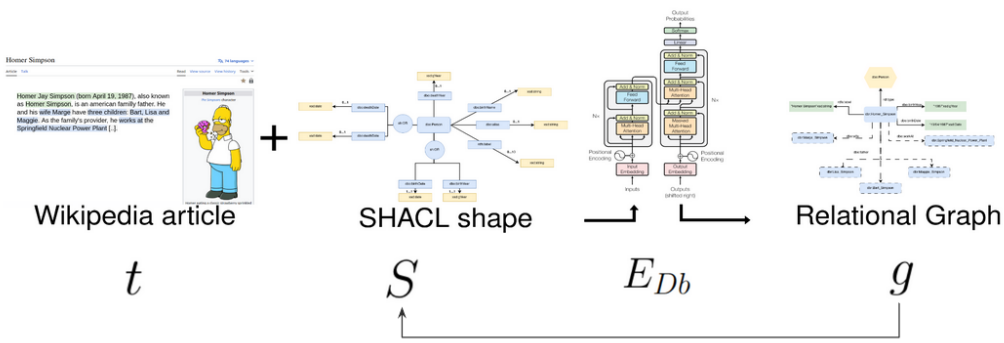
*Corresponding author. E-mail: firstname.lastname@inria.fr.

finetuning. Moreover, a structured extraction task requires the use of structured constraint approaches. These approaches actually require an additional cost at the inference step, which is not suitable in a knowledge-intensive context. A promising research direction is to design frugal models allowing adaptability and fine-grained quality control. Encoder-decoder models can solve extraction as a translation task, offering the option to produce linearised graphs from natural language directly. In this approach, the production of triples in RDF syntax allows for the direct integration of the produced triples into the Semantic Web ecosystem. The usage of constraints (e.g., with SHACL or ShEx) can help formalise both the extraction/learning set target and the validation of the produced output.

However, to the best of our knowledge, no study has been proposed to analyse the impact of the choice of the Encoder-Decoder models and the selection of a specific RDF syntax model to perform RE directly from Wikipedia articles.

*Pattern-based relation extraction* : Formally, let $Db \subseteq W \times G$ be a dual base, where $W$ is a set of Wikipedia articles and $G$ the set of corresponding KGs. Our goal is to learn a pattern-based extractor leveraging generative PLM: $E_{Db}$: $W \times S \rightarrow G; (t, s) \mapsto g$, where $t \in W$ is an input text, $s \in S$ is a set of SHACL shapes, and $g$ is an RDF graph implied by $t$ and valid against $s$. We refer to this task as pattern-based relation extraction, as illustrated in Fig.1

Fig. 1. Pattern-based relation extraction process




Encoder-Decoder models could be easily adapted to solve a seqToseq task, but variations in prompts and output formats can affect their performance. In this paper, we focus on RE for the most common datatype properties of DBpedia resources of type `dbo:Person`. In this specific yet frequent set-up, we challenged encoder-decoder models trained on various RDF syntaxes.

Hence, our research question:

**How does the choice of a linearization syntax impact the generation of RDF triples representing datatype properties for different pre-trained language models?**

We have presented initial experiments to answer this question in [63], where we already challenged Bart and T5 on a set of 12 syntaxes. In this paper, we extend our analysis by considering three additional encoder-decoder models (flanT5, codeT5, and pileT5), and one additional RDF syntax (Turtle Ultra Light). Moreover, we better define the meta-metrics we proposed, which leads us to new insights about the training behavior of the models we finetuned. The produced material is made open and reusable under an open license: the extension of the 12ShadesofRDF Github repository[1] as well as the complete experimental results[2].

The paper is organized as follows. After reviewing the related works (Section 2), we present our method to extract RDF from text with an application to Wikipedia (Section 3). We then report on the experiments and evaluations we carried out (Section 4) before discussing the results (Section 5).

[1] https://github.com/datalogism/12ShadesOfRDFSyntax
[2] https://wandb.ai/celian-ringwald/12ShadesOfRDFExtension, https://wandb.ai/celian-ringwald/12ShadesOfRDF

## 2. Related Works

*Generative language models for RE* : Before investing in generative PLMs, the research community focused on systems built on top of encoder-only PLMs (derived from BERT [5]), where relations were decoded by design in a discriminative manner [49]. Since 2021, generative PLMs have gained interest after demonstrating their ability to solve complex tasks in an end-to-end design. The solutions based on pre-trained generative transformer models rely either on encoder-decoder or decoder-only models:

- (1) **Encoder-Decoder** models traditionally proposed for translation or summarization tasks also demonstrate several successes in Question Answering (QA) and RE tasks which were achieved by finetuning BART [33] and T5 models [58]. For RE we can cite: REBEL [23], TALN [52], DEEPstruct [72] or UIE [41], GenIE [28].
- (2) **Decoder-only models** have interesting generalisation properties when scaled, which paved the way for many model proposals better known today as Large language models (LLM). The Text2KGBench [46] proposes to benchmark LLMs in the context of Ontology-Driven relation extraction, which is closely related to our pattern-based design, demonstrating the difficulties of such models in dealing with more domain-focused RE. More globally, in the classical context of relation extraction tasks, they have shown a small margin of improvement using few-shot learning, which requires a lot of engineering [8, 35, 71, 83]. When it comes to adapting them in a specific context using fine-tuning LLMs, it requires considerable resources. Parameter-efficient fine-tuning (PEFT) approaches [6] allow the adaptation of large models by reducing this cost, but do not necessarily perform as well as encoder-decoder fine-tuned models [14, 24, 32, 36].

The usage of LLMs is debated today [17, 42, 73], as they are costly to train, slow at the inference stage and hard to adapt to a specialised domain [30]. Moreover, they are also not particularly adept at handling hallucinations and structured output, and they raise essential sovereignty issues when used via the APIs via few-shot prompting. Nonetheless, their capacity to understand patterns and generate reliable natural language content gives them a strategic place in intermediary steps such as data-augmentation to later learn smaller, specialised models [55]. This was also experimented in the context of RE [29, 85].

*RDF-Syntaxes for RE* Globally, generative pre-trained models may be adapted to respect a given syntax implicitly from the examples submitted during training [80]. The question of the structure of the output was initially referred to as "Answer Engineering" [38], but in the domain of graph extraction, the community refers to it as the "linearization process" i.e. the transformation of a graph structure into a raw sequence of tokens. This allows using a generative model pre-trained on natural language texts [27].
The two main solutions found in the literature to represent graphs are:

- list of triples [72], e.g. $((s1, p1, o1), (s1, p2, o2), ...)$
- sequence of tagged elements [31], e.g. with tags $H, R, T$ in $\langle H \rangle s1 \langle R \rangle p1 \langle T \rangle o1 \langle H \rangle s1 \langle R \rangle p2 \langle T \rangle o2$.

[23] and [29] proposed a triple linearization method (subject-collapsed) where triples sharing the same subject are grouped to avoid repetition. In this benchmark, we will also consider the syntaxes recommended by the W3C to serialize RDF triples, namely:

- RDF/XML, a historical syntax with the verbosity of XML
- N-Triples, an easy-to-parse line-based format
- Turtle, a lighter and easier-to-read syntax supporting the use of qualified names for compacting URIs, and integrating shortcuts for the writing of triples sharing the same subject or predicate[3]
- JSON-LD, relying on the popular JSON format.
- And variations of the list, tags and Turtle syntaxes

Few research works proposed RDF-generated content with language models. To illustrate it, a recent dedicated survey [59] only covers one model explicitly generating RDF triples, where a majority of the analysed models follow the tags or the list syntaxes presented above. Still, the research interest in producing more complex structured

[3] https://www.w3.org/TR/turtle/#predicate-lists

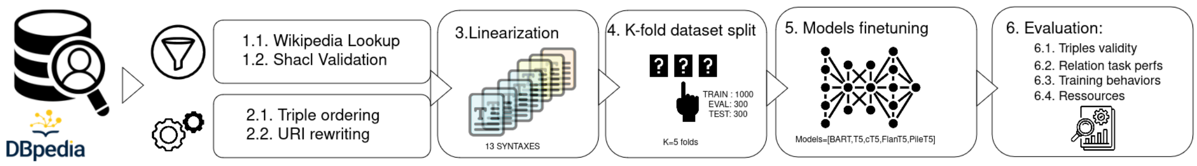
output with language models has grown, notably to perform information extraction [4, 37, 40]. These works mainly focus on the usage of LLMs to benchmark a data-extraction task with JSON syntax, and sometimes other syntaxes such as XML. Other research also demonstrates the potential of constrained generation with a grammar [16], which is reliable on simple syntaxes but expensive on complex ones. The Semantic Web community also proposed research directions using LLMs to assist Knowledge Graph engineering. For instance, [44] conducted to an extended benchmark focused on graph understanding and generation tasks based on the Turtle syntax [12]. The approach was extended to other W3C languages such as SPARQL [43] or RML [20]. However, few initiatives have been proposed to rigorously compare and benchmark smaller and frugal models to produce RDF syntaxes as we do in this extended paper.

*Positioning :* In this article, we propose a benchmark that focuses solely on Encoder-Decoder based PTMs, which offer an interesting performance/cost balance and may be adapted to a variety of output syntaxes via fine-tuning. The goal is to determine the optimal model and syntax for extracting relations most effectively and at minimal cost, while maintaining the highest expressivity.

## 3. Methodological Framework

### 3.1. Overview of the Approach

Fig. 2. Overview of our pipeline: from dataset construction to relation extraction evaluation




Our method is depicted in Figure 2. Our pipeline takes as input a DBpedia dump[4] from which we extract a subset containing specifically targeted triples. This subset is then filtered to check that the values of the selected DBpedia triples are mentioned in the corresponding Wikipedia abstracts (step 1.1) and comply with a predetermined SHACL shape (step 1.2) presented in Section 3.3. The selected triples are then ordered (step 2.1) and the URIs are URL-encoded and cleaned (step 2.2). The resulting dataset is then linearized (step 3) into 13 distinct syntaxes presented in Section 3.4. The obtained corpora are finally split into 5 independent folds (step 4). These subsets are used to fine-tune five SLMs (step 5) presented in Section 3.5. The fine-tuning configuration of the experiment is described in Section 4. The evaluation of the relation extraction (step 6) is conducted at the end of the training step with the computation of the metrics detailed in Section 4.2.

### 3.2. Task definition: a relation extraction focused on a maximal target shape

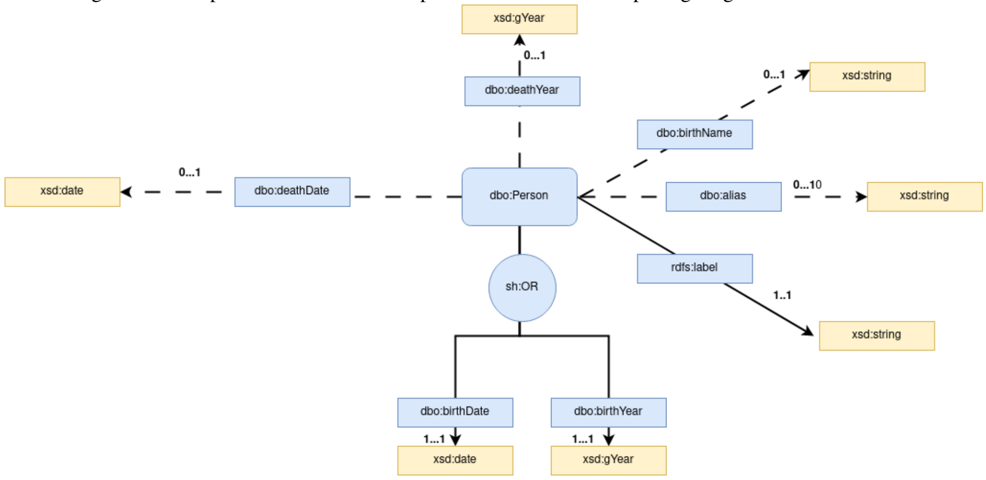
We start by formally defining the relation extraction task introduced in our previous work [63]. It relies on a training set built from the dual base $\mathcal{K}$ defined from the set $\mathcal{W}$ of Wikipedia abstracts associated with the set $\mathcal{G}$ of DBpedia graphs describing ($desc()$) the same resources $e$:

$$\mathcal{K} := \{(w, g) \in \mathcal{W} \times \mathcal{G}, \exists\, e \in IRI \text{ such that } desc_{\mathcal{W}}(e) = w \wedge desc_{\mathcal{G}}(e) = g\} \quad (1)$$

To ensure the quality of the training set, we consider the subset of $\mathcal{K}$ where all the graphs are valid against a SHACL shape $s^*$. We call this shape maximal, as it matches the largest pattern to be extracted. We note $g \models s^*$ this

[4] https://databus.dbpedia.org/dbpedia/collections/dbpedia-snapshot-2022-09

Fig. 3. Visual representation of s* in ShapeVOWL of SHACL shape targeting class `dbo:Person`




validation, and $\mathcal{K}_{s^*}$ the corresponding subset:

$$\mathcal{K}_{s^*} := \{(w, g) \in \mathcal{K}\,,\ g \models s^*\} \tag{2}$$

Finally, to reduce the noise in $\mathcal{K}_{s^*}$ entailed by the mismatch between DBpedia graphs and Wikipedia abstracts, we focus only on the couples $(w, g)$ where the abstract $w$ entails the paired graph $g$, i.e. the triples of $g$ that can effectively be extracted from the paired abstract $w$. We note it $w \models g$ and we denote the dataset by $\mathcal{K}_{s^*}^{\mathcal{W}\models}$:

$$\mathcal{K}_{s^*}^{\mathcal{W}\models} := \{(w, g) \in \mathcal{K}_{s^*}\,,\ w \models g\} \tag{3}$$

$\mathcal{K}_{s^*}^{\mathcal{W}\models}$ is used to train a model expected to predict, from an abstract $w$, a graph $\hat{g}$ valid against $s^*$. We denote by $\mathcal{M}$ this original model, and baseline:

$$\mathcal{M} : \begin{cases} \mathcal{W} \to \mathcal{G} \\ w \mapsto \hat{g}\,,\ \hat{g} \models s^* \wedge w \models \hat{g} \end{cases} \tag{4}$$

### *3.3. Dataset and Ground Truth*

Our experiment focuses on a simplified relation extraction task to better analyse the impact of the syntax. To avoid any entity linking step related to object properties, we limit our experiment to datatype properties with number, string and date values. This is a good starting point because language model hallucinations generally affect these literal values [25]. Moreover, until now, the proposed generative models mostly focus on object properties, allowing for constrained decoding [28] that cannot be envisaged in the case of datatype properties.

We consider the DBpedia subgraphs describing instances of one of the most represented DBpedia classes ($\mathcal{G}$), `dbo:Person`, and their corresponding Wikipedia abstracts ($\mathcal{W}$). The descriptions of instances of this class also include the highest number of datatype properties. We focus on the 7 datatype properties that are most likely to be found in the abstracts describing `dbo:Person` instances: `rdfs:label`, `dbo:alias`, `dbo:birthName`, `dbo:birthDate`, `dbo:deathDate`, `dbo:birthYear`, `dbo:deathYear`, as illustrated in Fig.3 using ShapeVOWL visualizations.

Several works mention the noise caused by the massive alignment of facts with text [66], which also impacts T-Rex or REBEL [36]. More specifically, two problems are pointed out: the values in a triple do not necessarily appear in the corresponding text, and conversely, the facts in the text may not have counterpart triples in the knowledge

Fig. 4. Linearization in Turtle syntax of the SHACL shape s* targeting class `dbo:Person`

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dbo: <http://dbpedia.org/ontology/> .

schema:PersonShape a sh:NodeShape ;
 sh:targetClass dbo:Person ;
 sh:property [
   sh:path rdfs:label;
   sh:minCount 1 ;
   sh:datatype xsd:string;
 ];
sh:or (
    [
      sh:property [
         sh:path dbo:birthDate;
         sh:datatype xsd:date;
         sh:minCount 1;
         sh:maxCount 1;
      ]
    ]
    [
      sh:property [
          sh:path dbo:birthYear;
          sh:datatype xsd:gYear;
          sh:minCount 1;
          sh:maxCount 1;
      ]
    ]
);
sh:property [
   sh:path dbo:deathYear;
   sh:minCount 0;
   sh:maxCount 1;
   sh:datatype xsd:gYear;
 ];
 sh:property [
   sh:path dbo:alias;
   sh:datatype xsd:string ;
   sh:minCount 0;
   sh:maxCount 10;
   sh:nodeKind sh:Literal;
 ];
 sh:property [
   sh:path dbo:birthName ;
   sh:datatype xsd:string ;
   sh:minCount 0;
   sh:maxCount 1;
   sh:nodeKind sh:Literal ;
 ] ;
 sh:property [
   sh:path dbo:deathDate ;
   sh:datatype xsd:date ;
   sh:minCount 0;
   sh:maxCount 1;
 ].
```

base. To solve those issues, we first considered a maximal SHACL shape ($s^*$) targeting class `dbo:Person` (Fig. 4) and specifying which property is mandatory and which one is optional, and we only keep the graphs valid against this shape ($K_{s^*}$ described in Eq. 2). In a second step we keep only the triples whose values can be found in the Wikipedia abstract of a given entity ($\mathcal{K}_{s^*}^{\mathcal{W}\models}$ described in Eq. 3). When applying these two pre-processing steps to a random sample of 1000 entities, we found that 80% of the triples have values that can be found in the corresponding Wikipedia abstract, but only 45% of the entities have a graph description valid against the shape.

Table 1

Datatype properties statistics of the ground truth $D$

| property | freq | variability |
|---|---|---|
| http://www.w3.org/1999/02/22-rdf-syntax-ns#type | 100,0 | 0,0 |
| http://dbpedia.org/ontology/birthYear | 100,0 | 5,3 |
| http://www.w3.org/2000/01/rdf-schema#label | 100,0 | 99,9 |
| http://dbpedia.org/ontology/birthDate | 98,3 | 92,9 |
| http://dbpedia.org/ontology/deathYear | 31,0 | 13,4 |
| http://dbpedia.org/ontology/deathDate | 30,3 | 97,6 |
| http://dbpedia.org/ontology/birthName | 7,0 | 100,0 |
| http://dbpedia.org/ontology/alias | 1,4 | 100,0 |

Our pipeline includes two additional pre-processing steps:

1. Triples ordering: [45, 48, 76] demonstrated the positive impact of typing the entity in the context of the RE task. Therefore, asking a model to first type the main subject of our graphs could be a good idea. However, RDFlib[5] does not guarantee the same order of triples for every syntax. So we implemented a triple-ordering step, which enforces the first triple to be the type of the subject, as illustrated below in Fig. 5.
2. URI encoding: the Turtle syntax uses tokens that can be found in URIs (dots and parenthesis), but their usage is forbidden in local names. We had to encode them systematically. In DBpedia, the Turtle output of URIs containing these special characters is written without prefixes and wrapped into brackets, e.g.

[5]https://rdflib.readthedocs.io/en/stable/

Fig. 5. Example of reordered triples


`<https://dbpedia.org/resource/Tom_Nichols_(footballer)>`. Therefore we rewrite this URI as `dbr:Tom_Nichols_%28footballer%29`.

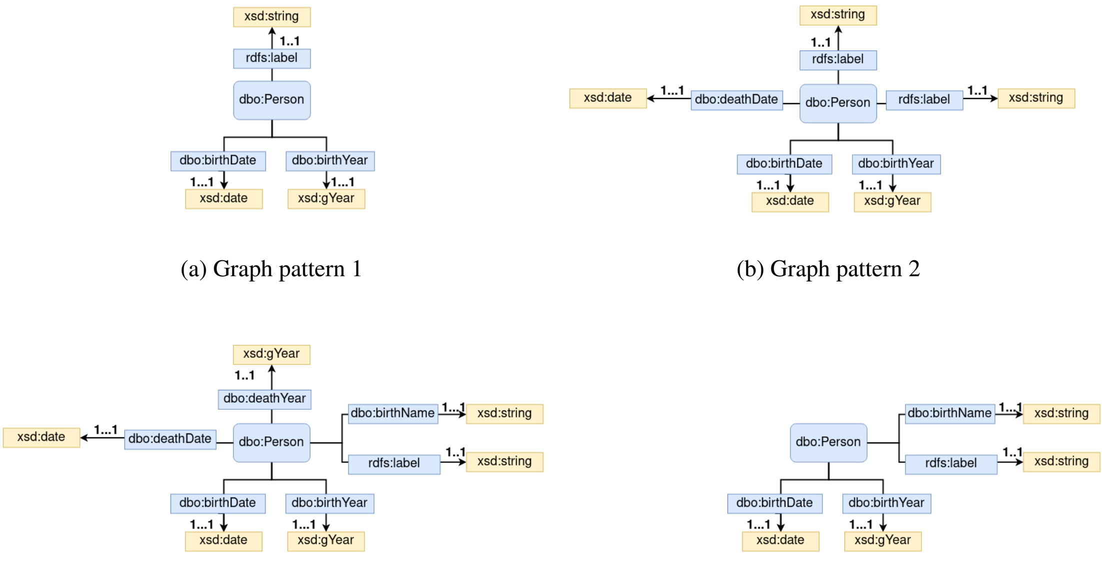
Our ground truth $D$ is randomly sampled from $\mathcal{K}_{s*}^{\mathcal{W}\models}$ and will later be sampled and splited into $D_{train}$, $D_{eval}$ and $D_{test}$. $D$ contains 6000 entities described by 28M triples. The 7 datatype properties we are focusing on are associated with 13 832 distinct values. Table 1 shows the frequency of each property defined as the number of occurrences of a property divided by the number of entities described in $D$, and its variability defined as the number of distinct values for a property divided by the number of occurrences of the property. The frequencies reported in this table show that all the graphs in the dataset contain an occurrence of properties `rdf:type`, `dbo:birthYear`, and `rdfs:label` (which is in line with the shape we defined), and they highlight the low representation of properties `dbo:alias` and `dbo:birthName`. The reported variabilities highlight the historical bias of the data: the low variability of properties `dbo:birthYear` and `dbo:deathYear` means that the entity descriptions in the dataset target some specific historical period. All the combinations of properties identified in $D$ are described in Table 2. Interestingly, a low number of property combinations are represented in the dataset: there are 18 combinations of properties among 127 possible combinations of 7 properties, and their frequencies indicate a long-tail distribution. The four most instantiated graph patterns are shown in Fig. 6.

Table 2

Property combinations observed in the ground truth $D$

| prop_pattern | label | birthYear | birthDate | deathYear | deathDate | alias | birthName | No Prop. | freq |
|---|---|---|---|---|---|---|---|---|---|
| pattern1 | ✓ | ✓ | ✓ | | | | | 3 | 64,43% |
| pattern2 | ✓ | ✓ | ✓ | ✓ | ✓ | | | 5 | 25,47% |
| pattern3 | ✓ | ✓ | ✓ | ✓ | ✓ | | ✓ | 6 | 3,67% |
| pattern4 | ✓ | ✓ | ✓ | | | | ✓ | 4 | 2,85% |
| pattern5 | ✓ | ✓ | ✓ | | | ✓ | | 4 | 0,80% |
| pattern6 | ✓ | ✓ | | | | | | 2 | 0,72% |
| pattern7 | ✓ | ✓ | | ✓ | ✓ | | | 4 | 0,62% |
| pattern8 | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | | 6 | 0,40% |
| pattern9 | ✓ | ✓ | ✓ | ✓ | | | | 4 | 0,37% |
| pattern10 | ✓ | ✓ | | ✓ | ✓ | | | 4 | 0,17% |
| pattern11 | ✓ | ✓ | ✓ | ✓ | | | ✓ | 5 | 0,10% |
| pattern12 | ✓ | ✓ | ✓ | | | ✓ | ✓ | 5 | 0,10% |
| pattern13 | ✓ | ✓ | | | | | ✓ | 3 | 0,10% |
| pattern14 | ✓ | ✓ | | ✓ | ✓ | ✓ | | 5 | 0,03% |
| pattern15 | ✓ | ✓ | ✓ | ✓ | ✓ | | | 5 | 0,03% |
| pattern16 | ✓ | | ✓ | ✓ | | | ✓ | 4 | 0,03% |
| pattern17 | ✓ | ✓ | | | | ✓ | ✓ | 4 | 0,02% |
| pattern18 | ✓ | ✓ | | | | ✓ | | 3 | 0,02% |

Fig. 6. Top 4 graph patterns represented in our training set




(a) Graph pattern 1

(b) Graph pattern 2

(c) Graph pattern 3

(d) Graph pattern 4

### 3.4. RDF Syntaxes and Alternative Linearizations

Our benchmark considers 13 syntaxes belonging to three distinct categories. The first category includes the syntaxes classically found in the literature (Fig. 7): (a) the List (noted $_l$) and (b) the Tags (noted $_g$) and (c/d) their factorized variations (noted $_f$). As underlined by [23] and [29], factorisation enables to avoid the repetition of subjects or predicates. It is naturally integrated into Turtle, JSON-LD and RDF-XML and we also integrated it to the List and Tags syntaxes (noted $_f$).

Fig. 7. Linearization proposed by the literature applied on a graph describing Nelson Mandela.

```
[[Nelson_Mandela,type,Person],[Nelson_Mandela,label,Nelson Mandela],
[Nelson_Mandela,birthDate,1918-07-18],[Nelson_Mandela,birthYear,1918],
[Nelson_Mandela,deathDate,2013-12-05],[Nelson_Mandela,deathYear,2013]]
```

(a) List syntax

```
<subj>Nelson_Mandela<rel>type<obj>Person<et><subj>Nelson_Mandela<rel>label<obj>Nelson Mandela<et>
<subj>Nelson_Mandela<rel>birthDate<obj>1918-07-18<et><subj>Nelson_Mandela<rel>birthYear<obj>1918<et>
<subj>Nelson_Mandela<rel>deathDate<obj>2013-12-05<et><subj>Nelson_Mandela<rel>deathYear<obj>2013<et>
```

(b) Tag syntax

```
[[Nelson_Mandela,[[type,Person],[label,Nelson Mandela],[birthDate,1918-07-18],
[birthYear,1918],[deathDate,2013-12-05],[deathYear,2013]]]]
```

(c) List syntax factorized

```
<subj>Nelson_Mandela<rel>type<obj>Person<rel>label<obj>Nelson Mandela
<rel>birthDate<obj>1918-07-18<rel>birthYear<obj>1918
<rel>deathDate<obj>2013-12-05<rel>deathYear<obj>2013<et>
```

(d) Tag factorized

The second category includes the 4 W3C RDF syntaxes (Fig. 8): (a) XML-RDF (noted $_x$), (b) N-Triples (noted $_n$), (c) Turtle (noted $_T$), and (d) JSON-LD (noted $_j$).

Fig. 8. Linearization proposed by the W3C applied on a graph describing Nelson Mandela.

```
<rdf:RDF xmlns:dbo="http://dbpedia.org/ontology/"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
    <rdf:Description rdf:about="http://dbpedia.org/resource/Nelson_Mandela">
        <rdf:type rdf:resource="http://dbpedia.org/ontology/Person"/>
        <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Nelson Mandela</rdfs:label>
        <dbo:birthDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1918-07-18</dbo:birthDate>
        <dbo:birthYear rdf:datatype="http://www.w3.org/2001/XMLSchema#gYear">1918</dbo:birthYear>
        <dbo:deathDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date">2013-12-05</dbo:deathDate>
        <dbo:deathYear rdf:datatype="http://www.w3.org/2001/XMLSchema#gYear">2013</dbo:deathYear>
    </rdf:Description>
</rdf:RDF>
```

(a) RDF-XML

```
<http://dbpedia.org/resource/Nelson_Mandela><http://www.w3.org/1999/02/22-rdf-syntax-ns#type><http://dbpedia.org/ontology/Person> .
<http://dbpedia.org/resource/Nelson_Mandela><http://dbpedia.org/ontology/birthYear>
"1918"^^<http://www.w3.org/2001/XMLSchema#gYear> .
<http://dbpedia.org/resource/Nelson_Mandela><http://www.w3.org/2000/01/rdf-schema#label>
"Nelson Mandela"^^<http://www.w3.org/2001/XMLSchema#string> .
<http://dbpedia.org/resource/Nelson_Mandela> <http://dbpedia.org/ontology/deathYear>
"2013"^^<http://www.w3.org/2001/XMLSchema#gYear> .
<http://dbpedia.org/resource/Nelson_Mandela> <http://dbpedia.org/ontology/deathDate>
"2013-12-05"^^<http://www.w3.org/2001/XMLSchema#date> .
<http://dbpedia.org/resource/Nelson_Mandela> <http://dbpedia.org/ontology/birthDate>
"1918-07-18"^^<http://www.w3.org/2001/XMLSchema#date> .
```

(b) N-triples

```
@prefix dbo: <http://dbpedia.org/ontology/>.
@prefix dbr: <http://dbpedia.org/resource/>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

dbr:Nelson_Mandela  a  dbo:Person ;
    rdfs:label "Nelson Mandela"^^xsd:string ;
    dbo:birthDate "1918-07-18"^^xsd:date ;
    dbo:birthYear "1918"^^xsd:gYear ;
    dbo:deathDate "2013-12-05"^^xsd:date ;
    dbo:deathYear "2013"^^xsd:gYear.
```

(c) Turtle syntax

```
[{
    "@id": "http://dbpedia.org/resource/Nelson_Mandela",
    "@type": [ "http://dbpedia.org/ontology/Person" ],
    "http://dbpedia.org/ontology/birthDate": [
      { "@type": "http://www.w3.org/2001/XMLSchema#date",
      "@value": "1918-07-18" } ],
    "http://dbpedia.org/ontology/birthYear": [
      { "@type": "http://www.w3.org/2001/XMLSchema#gYear",
      "@value": "1918" } ],
    "http://dbpedia.org/ontology/deathDate": [
      { "@type": "http://www.w3.org/2001/XMLSchema#date",
      "@value": "2013-12-05" } ],
    "http://dbpedia.org/ontology/deathYear": [
      { "@type": "http://www.w3.org/2001/XMLSchema#gYear",
      "@value": "2013" } ],
    "http://www.w3.org/2000/01/rdf-schema#label": [
      { "@value": "Nelson Mandela" } ]
}]
```

(d) JSON-LD

The third category includes additional syntaxes we propose: *Turtle light* (noted $_t$) is a simplified Turtle syntax where namespaces, prefixes, and datatypes are considered as already defined. Turtle light comes with several variations: one with subject factorisation, one with single-line writing (noted $_1$) to evaluate the impact of the line break[6]. Another variation combines factorisation and single-line writing. In addition to these 12 syntaxes considered in our initial experiments, we introduced an Ultra Light Turtle syntax (noted $_u$) which is based on the

[6]the "\n" special token

factorised, inlined Turtle Light where we remove the ":" characters in the qualified names.

Fig. 9. Turtle light variations proposed applied on a graph describing Nelson Mandela.

(a) Turtle Light syntax

```
:Nelson_Mandela a :Person.\n
:Nelson_Mandela :label "Nelson Mandela".\n
:Nelson_Mandela :birthDate "1918-07-18".\n
:Nelson_Mandela :birthYear "1918".\n
:Nelson_Mandela :deathDate "2013-12-05".\n
:Nelson_Mandela :deathYear "2013".
```

(b) Turtle Light syntax on one line

```
:Nelson_Mandela a :Person. :Nelson_Mandela :label "Nelson Mandela". :Nelson_Mandela :birthDate "1918-07-18".
:Nelson_Mandela :birthYear "1918". :Nelson_Mandela :deathDate "2013-12-05". Nelson_Mandela :deathYear "2013".
```

(c) Turtle Light syntax factorized

```
:Nelson_Mandela a :Person ; \n
    :label "Nelson Mandela" ; \n
    :birthDate "1918-07-18"; \n
    :birthYear "1918"; \n
    :deathDate "2013-12-05";\n
    :deathYear "2013".
```

(d) Turtle Light syntax on one line and factorised

```
:Nelson_Mandela a :Person; :label "Nelson Mandela"; :birthDate "1918-07-18"; :birthYear "1918";
:deathDate "2013-12-05"; :deathYear "2013".
```

(e) Turtle Ultra Light

```
Nelson_Mandela a Person; label "Nelson Mandela"; birthDate "1918-07-18"; birthYear "1918";
deathDate "2013-12-05"; deathYear "2013".
```

Finally, we consider the use of **vocabulary extension** (noted $_v$), which is a common practice, notably used for QA SPARQL query generation, that consists in extending the tokenizer vocabulary [2, 60]. This ensures that syntax-related tokens will not be considered as unknown by the tokenizer, and allows the model to learn a vector representation of these tokens during the fine-tuning process, e.g. the representation of the comma in Turtle code replacing the representation of the comma from the pre-trained embedding space. For each W3C syntax, we added all the tokens specified in its recommendation.

### *3.5. Benchmarking frugal Encoder-Decoders*

In our initial experiments described in [63] we focused on the two encoder-decoder models traditionally used in the literature (see Section 2), namely BART (noted $B$) and $T5$. We extended our benchmark by integrating three other T5-based models: FlanT5 [3] noted $T5^F$, PileT5 [69] noted $T5^P$ and CodeT5 [75] noted $T5^C$. These models follow the encoder-decoder transformer architecture. Table 3 summarizes their main differences. We can see from this table several differences: first, the number of pretraining objectives: all of them are pretrained using token masking, whereas CodeT5 and BART were also pretrained, along with other objectives. codeT5 was, for example, pretrained on task-specific code context: 1. identifier prediction, 2. and identifier tagging on both unimodal and bimodal data (so from code to natural language and vice versa). Bart, in this case, was trained to better deal with noisy text via pretraining using 1. token deletion, and 2. Text-Infilling 3. Sentence permutation and 4. Document rotation. In addition, the T5 family of models is pre-trained to solve and gain an initial understanding of specific tasks using prefix tuning in the case of T5 and codeT5, and using Instruction tuning in the cases of FlanT5 and PileT5.

We limited our experiments to pre-trainted models in their “base” versions counting between 140M and 248M parameters, which is frugal compared to decoder-only LLMs that count billions of parameters [47]. Concerning

Table 3
Encoder–decoder model specificities.

| Model | Pretraining Objective | Prefix Tuning | Instruct. Tuning | Pretraining Dataset | Tokenizer | Nb Params | Max Input Len. | Max Output Len. |
|---|---|---|---|---|---|---|---|---|
| **BART** | Token Masking + 4 other obj. | – | – | BookCorpus + Wikipedia 2019 | BPE | 140M | 1024 | 1024 |
| **T5** | Token Masking | ✓ | – | C4 | SentencePiece | 220M | 512 | 512 |
| **CodeT5** | Token Masking + 2 other obj. | ✓ | – | CodeSearchNet | Byte-Level BPE | 220M | 512 | 256 |
| **FlanT5** | Token Masking | ✓ | ✓ | C4 + Muffin + F0-SF + CoT + NaturalInstructionsV2 | SentencePiece | 248M | 1024 | 256 |
| **PileT5** | Token Masking | ✓ | ✓ | The Pile | LLaMA Tokenizer | 248M | 512 | 512 |

the datasets used for pretraining, we can underline that all of them use different corpora, and not all of them are open and available, while it is the case for BART. However, unlike larger LLMs, these models were not pre-trained directly on KG data from Wikidata or DBpedia. But all of them were trained at least on a complete Wikipedia dump from 2019 (for BART, T5, and FlanT5) or 2020 (for PileT5). In contrast, the training dataset for CodeT5 is limited to coding tasks and does not include Wikipedia pages.

Each model uses different tokenizers and has different context sizes related to the specific input/output lengths on which they were pre-trained. For this reason, the input and generation lengths must be adapted to these constraints to avoid generation degradation or early truncation. Concerning the input size, our models is not impacted by truncation of text fitting the input length, since all the values targeted by our shape are generally expressed in the beginning of the abstract. Fig. 10 illustrates the effect of the tokenization of each model tested on the same Turtle graph. We can see on that example that T5-base is the model using the highest number of tokens to represent the sequence, while codeT5 is the model representing the sequence with the lowest number of tokens. We also see that Bart is generally representing words starting with uppercase as a unique unit of meaning while the T5 models tokenize the uppercased letter separately from the rest of the word.

Table 4
Max number of tokens generated over the ground truth $D$, depending on the chosen syntax and tokenizer

| | B | T5 | $T5^P$ | $T5^F$ | $T5^C$ |
|---|---|---|---|---|---|
| $ntriples$ | 541 | <u>583</u>* | <u>610</u>* | <u>583</u>* | <u>566</u>* |
| $XML$ | 479 | 514 | 498 | 514* | 481* |
| $JSON-LD$ | <u>646</u> | 411 | 488 | 411* | 469* |
| $list$ | 392 | 285 | 316 | 285* | 320* |
| $Turtle$ | 286 | 312 | 289 | 312* | 284* |
| $turtleLight$ | 307 | 274 | 310 | 274* | 314* |
| $tags$ | 229 | 265 | 273 | 265* | 259* |
| $turtleLight_1$ | 303 | 225 | 270 | 225 | 255 |
| $list_f$ | 152 | 137 | 171 | 137 | 164 |
| $turtleLight_f$ | 146 | 127 | 153 | 127 | 142 |
| $turtleLight_{1f}$ | 121 | 127 | 140 | 127 | 135 |
| $tags_f$ | **78** | 97 | **99** | 97 | **104** |
| $turtleUltraLight$ | 92 | **85** | 100 | **85** | **104** |

Table 4 shows the longest size (in number of tokens) of the sequences that each model have to generate according to the ground truth $D$ depending on the targeted RDF syntax. Starred values ($*$) indicate that the longest sequence contained in $D$ exceeds the maximum output length of the pre-trained model. In practice, the encoder-decoder models can produce sequences longer than the maximal output length, but then they are more prone to generation degeneration.

This highlights potential issues with CodeT5 and FlanT5 whose maximum output length (256) is exceeded for 7 syntaxes. This table also reveals the fact that light syntaxes allow to divide the output context length by five, compared to complex syntaxes (Ntriple, XML or JSON): for instance, to represent 7 datatype properties, BART with one-line, factorized Turtle light ($B_{vt1f}$) requires 121 tokens while BART with JSON ($B_{vj}$) requires 541 tokens. From that observation, we can estimate that with the factorized, one-line Turtle Light syntax we can generate a graph of almost 60 properties while JSON hardly reaches 14 properties.

Fig. 10. Tokenisation comparison of the same Turtle graph.


In the following sections, we use the notations introduced above to name each possible configuration. For instance, a BART model trained on *Turtle Light* syntax, with factorization and multi-lines will be written $B_{vtf}$, and a codeT5 model trained on lists with a vocabulary will be written $T5^{C}_{vl}$.

## 4. Experimental set-up

### 4.1. Fine-Tuning Details

Our code[7] is based on a fork of REBEL[8] which we extended and adapted to our task. It is published under an open license. For each standard RDF syntax, we developed a specific parser and integrated the metrics we present below.

*Data Split:* We follow a 5-fold cross-validation based on 5 000 examples sampled from $D$ and then used to build rotative splits of 4 000 training examples ($D_{train}$) and 1 000 test examples ($D_{test}$). In addition, 250 disjoint examples also sampled from $D$ are used for the evaluation ($D_{eval}$).

*Configuration:* The BART model was fine-tuned using the inverse square root scheduler with an initial learning rate of 0.00005. For T5 we used the Adafactor scheduler with an initial learning rate of 0.001. Both models were fine-tuned with 1000 warm-up steps and configured with an early stop mode with patience of 5 steps. For FlanT5, codeT5 and PileT5, we used the same configuration as for T5. Models were trained on a single GPU, Tesla V100-SXM2-32GB for BART and NVIDIA A100 80GB PCIe for T5 models (able to manage bf16).

*Handling of Tokenization Inconsistencies:* T5 and BART tokenizers can duplicate or delete spaces before or after special tokens [1, 68]. For this reason, we controlled token consistency during the evaluation with a typographic checker and cleaner. This is applied to the learning examples and to the predicted output when both are compared.

*Finetuning Prompts:* We used two different prompts to finetune the tested models:

- for BART: "`$entity_URI : $Abstract`";
- for the T5 models family (T5, pileT5, codeT5 and Flan-T5):
  "`Translate English to $Syntax: [$entity_URI] $Abstract`",

where `$Abstract` is a Wikipedia abstract, `$Syntax` is the targeted RDF syntax, and `$entity_URI` the URI of the entity in DBpedia.

### 4.2. Evaluation Metrics

The first stage of this experiment is to evaluate the ability of the model to produce a given syntax without generating any parsing error. This is measured by the rate of Parsed Triples $R_{PT}$. We also introduce the rate of Correct Subject $R_{CS}$: the choice of the URI for the subject of a generated triple depends on the ability of the model to copy from the input the targeted entity URI. In addition, we define the rate of SHACL-Validated Triples $R_{SVT}$.

$$R_{PT} = \frac{Nb_{output\ parsed}}{Nb_{output\ generated}} \qquad R_{CS} = \frac{Nb_{URI\ found}}{Nb_{output\ parsed}} \qquad R_{SVT} = \frac{Nb_{output\ Valid}}{Nb_{output\ parsed}} \tag{5}$$

Non-parsable triples are evaluated by computing the Levenstein edit distance $lev(r_g, r_t)$ between the generated RDF code $r_g$ and the targeted one $r_t$. The result is the number of editions needed to transform $r_g$ into $r_t$.

By contrast, when the graphs produced are parsable it is possible to evaluate them with respect to the RE task. Traditionally, RE focuses on *precision* ($P$), *recall* ($R$), $F_1$ score, or *top@k* metrics. But as underlined in [19], these metrics are generally computed at the micro level, that is, they are computed on all the expected and produced triples without taking into account the variable distribution of the properties. As a result, these metrics can hardly account for the performance of a model on under-represented properties, as the micro metrics are dominated by the results obtained for much more frequent properties. In our context, we observed that the property distribution is unbalanced. Therefore we propose to also compute the macro measures ($P^+$, $R^+$, $F_1^+$) which average the metrics recorded on

[7] https://github.com/datalogism/12ShadesOfRDFSyntax
[8] https://github.com/Babelscape/rebel

each properties thus equally representing the performances for each property. These metrics follow the *Strict Mode* evaluation [70], comparing predicted and ground truth values and verifying their strict equality. The strict evaluation-based metrics are not the most appropriate to evaluate datatype properties with values of type `xsd:String`, where we may accept semantically close values. For this reason, we also compute the BLEU score [54]: the closer *BLEU* is to 1, the greater the similarity between string values. As our experimental framework is trained on 10-folds for each configuration, all the metric computed are averaged and noted with an over-lined, for instance, $\overline{F_1^+}$ is the average of $F_1^+$ over the folds.

We define a global grade $G_g$ that will allow us to compare the overall performances of our configurations. It combines the performance of the model in terms of parsability, SHACL validity and subject validity on one side, and in terms of macro $F_1$ on the other side: $G_g = \overline{R_{PT}} \times \overline{R_{CS}} \times \overline{R_{SVT}} \times \overline{F_1^+} \times 100$.

Finally, we also monitored the training time $T_t$ (in minutes) and the carbon footprint[9] $C_c$ (emissions of $CO_2$-equivalents in $g$) for training a model.

To assess the training process itself based on the cross-entropy loss objective, we define three meta-metrics from a metric of interest $\rho$: the velocity $V_\rho$, the stability $S_\rho$ and the divergence $D_\rho$. In our case we will use these meta-metrics to compare and follow the behaviors of the $R_{PT}$ and the $F_1^-$, consequently $\rho \in \{R_{PT}; F_1^-\}$. All meta-metrics consider a specific state of the training called *saturation* which is reached when the metric $\rho$ reaches a threshold $\lambda = 0.9$. The metrics of interest are computed for a fold $f \in [1..n_{fold}]$ and an epoch $e \in [1..n_{epoch}]$, noted $\rho(f, e)$. Let us now detail the computation of each of the meta-metrics:

(1) **The velocity** is the number of epochs needed to reach the first saturation ($\rho(f, e) > \lambda$) on a given fold $f$ by a metric $\rho$.

$$V_\rho(f) = min(\{e|e \in [1; n_{epoch}], \rho(f, e) > \lambda\}) \tag{6}$$

*Interpretation:* A velocity close to 0 indicates that the model saturates at the early stage of the training. Conversely, a velocity close to the number of epochs recorded by the whole training process indicates a late model saturation.
*Aggregation:* The average velocity on all the folds is computed as follows:

$$\overline{V_\rho} = \begin{cases} \dfrac{1}{n_{fold}} \displaystyle\sum_{f=1}^{n_{fold}} V_\rho(f) & \text{if } \forall f \in [1..n_{fold}]\ \exists e; \rho(f, e) > \lambda \\ \varnothing & \text{otherwise} \end{cases} \tag{7}$$

(2.1) **The broken steps set:** To define the stability, we first define the set of the *broken_steps* for a given fold $f$. During a fold $f$, a step is considered as broken if the metric $\rho(f, e)$ is smaller than the defined saturation threshold.

$$broken\_steps(f) = \{e|e \in [V_\rho(f), n_{epoch}], \rho(f, e) < \lambda\} \tag{8}$$

(2.2) **The stability** is the ratio of the number of epochs during which a metric remains stable after the first saturation.

$$S_\rho(f) = 1 - \frac{\|broken\_steps(f)\|}{n_{epoch}} \tag{9}$$

*Interpretation:* During the training, if the metric of interest rarely falls below the saturation threshold, then the stability will be close to 1; conversely, if it often falls below the threshold, the stability will be close to 0.
*Aggregation:* The average stability on the $n_{fold}$ folds is computed as follows:

[9] https://codecarbon.io/

$$\overline{S_\rho} = \begin{cases} \frac{1}{n_{fold}} \sum_{f=1}^{n_{fold}} S_\rho(f) & \text{if } \forall f \in [1..n_{fold}]\ \exists e; \rho(f,e) > \lambda \\ \varnothing & \text{otherwise} \end{cases} \tag{10}$$

(3) **The divergence** indicates if the final value of the metric is lower than the saturation threshold.

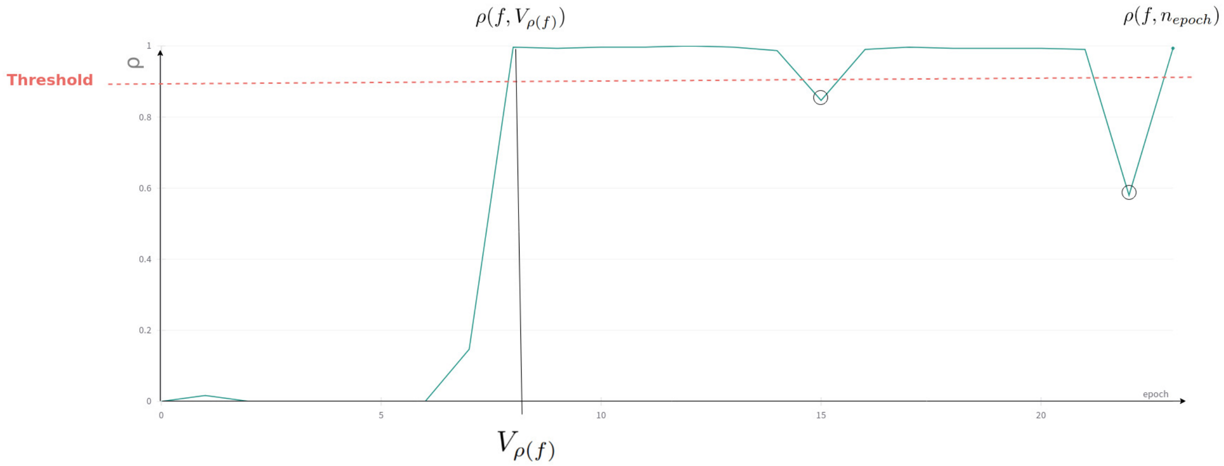
$$D_\rho(f) = \begin{cases} 1 & \text{if } V_\rho(f) \neq \varnothing \wedge \rho(f, n_{epoch}) < \lambda \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

*Interpretation:* The divergence is equal to 1 if the measure $\rho$ recorded at the end of the training is under the saturation threshold. Inversely, the divergence is equal to 0 if the training completes with a value higher than the saturation threshold.
*Aggregation:* We compute the sum on all the folds of each $D_\rho(f)$, and note it as $\sum D_\rho$.

$$\sum D_\rho = \sum_{f=1}^{n_{fold}} D_\rho(f) \tag{12}$$

To illustrate the principle of these meta-metrics, Fig. 11 shows the behaviour of a given metric $\rho(f,e)$ observed at a given fold $f$ and computed at each epoch $e$. In this example, the training took $n_{epoch} = 23$ epochs; $V_\rho(f) = 7$, i.e. $\rho(f,e)$ reaches the saturation threshold of 0.9 at the 7th epoch; $S_\rho(f) = 0.93$, with 2 *broken steps* (circled on the figure) during the training process; $D_\rho(f) = 0$, i.e. the model is not diverging at the end of the training process, as $\rho(f, n_{epoch}) > \lambda$.

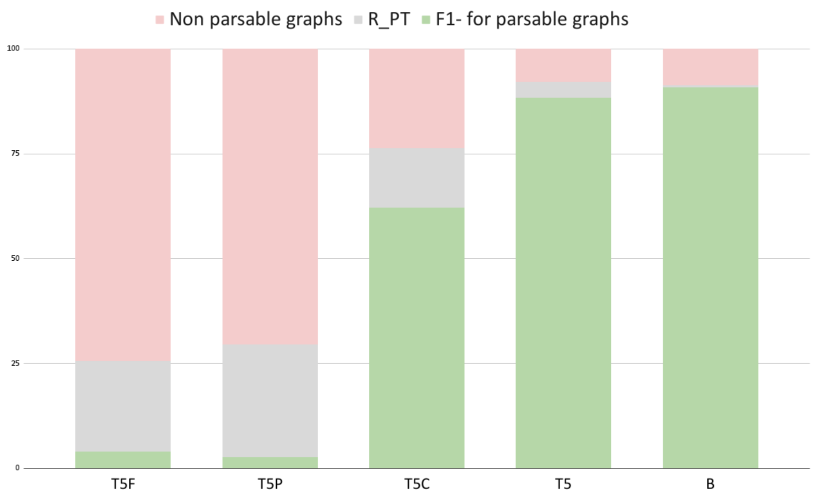
Fig. 11. Example behaviour of a metric $\rho$ during the training of a model on given fold $f$ during 23 epochs




## 5. Results and Discussions: Best models and syntaxes

Table 6, Table 7 and Table 8 compile the results obtained following the experimental framework described in the previous sections, ranked by global grade $G_g$. Table 6 gathers all the models that can effectively generate valid

Fig. 12. Averaged model performances $F_1^-$ relatively to $R_{PT}$




triples; it includes the models based on BART and T5. Table 7 gathers models with lesser performances, mostly based on codeT5, but also some models based on PileT5 and FlanT5. Table 8 gathers the worse models; these are based on FlanT5 and PileT5. RE metrics are computed on valid triples and, in that respect, the best models have a $\overline{F_1^+}$, $\overline{P^+}$ and $\overline{R^+}$ close to 0.95. This is a good result since the macro metrics are generally less optimistic and more informative than the micro ones with which every configuration seems to reach an almost perfect extraction. From that point of view, $T5_{vj}$ is the best model, closely followed by $B_{vgf}$, $B_{vu}$, $T5_{vu}$, $B_{vtf1}$, $B_{vT}$, $T5_{vtf1}$ and $T5_{vgf}$. It is more difficult to discuss the good performances of $T5^P_{vu}$, $T5^P_{vn}$ and $T5^P_{vn}$ in regards to their low triple validity scores.

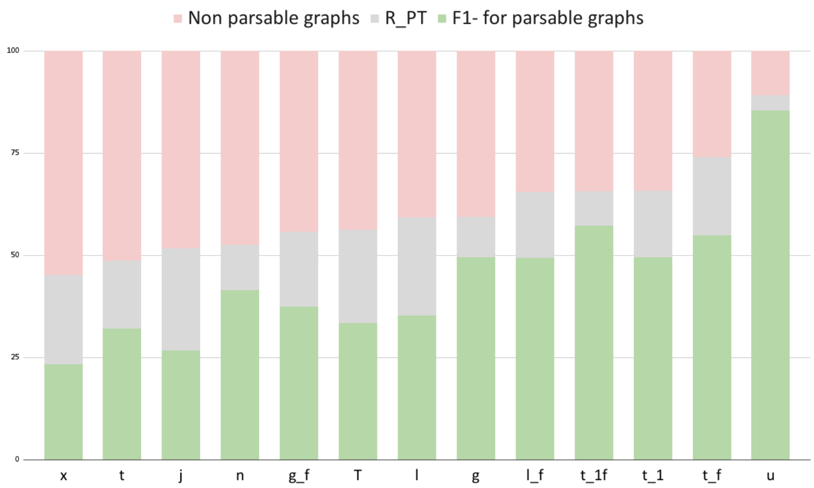
The bar charts of Fig. 12 and Fig. 13 put in perspective the averaged $F_1^-$ metrics obtained relatively to the $R_{PT}$ metric, by plotting the $F_1^-$ stacked over and at the scale of the $R_{PT}$ bar. Fig. 12 confirms the performances of $B$ and $T5$, as well as the relatively good results of $T5^C$. On the other hand, $T5^F$ and $T5^P$ are far behind the others. From the point of view of the syntax and as shown in Fig. 13, we can see the robustness of Turtle Ultra Light which has a high $F_1^-$ and a high $R_{PT}$, notably because this syntax is correctly handled by every model, even by $T5^F$ and $T5^P$. We can also observe globally a poor performance of the models on W3C syntaxes and a good performance of the tag and list syntaxes. Concerning TurtleLight, it seems necessary to apply the variations we proposed (the factorisation and the one-line writing) on this syntax to be able to perform well in terms of both $R_{PT}$ and $F_1^-$.

Coming to the details, and starting with the **triple validity** metrics, almost every configuration related to BART and T5 produces triples that could be parsed ($\overline{R_{PT}}$) with the exception of $T5$ struggling to produce the Turtle and N-Triples syntaxes. Concerning codeT5, only the Turtle (basic version, the Light and Ultra Light version) and the list-based syntaxes are nearly perfectly parsed. The capacity of codeT5 to deal with Turtle is peculiar to our context and could be explained by a syntax close to a code structure, with the definition of the prefixes at the top and the data content written after it. FlanT5 and PileT5 can approximately generate triples only in the simplest syntaxes. Among them, Turtle Ultra Light is the one recording the better $R_{PT}$ for FlanT5 (0,95) as well as for PileT5 (0,59).

Considering the **Levenshtein distance** ($\overline{lev}$) computed on the triples with parsing errors, we observe the ability of some models to extract close to perfect triples, particularly BART. Moreover, many models record negligible $\overline{lev}$ distances ($\overline{lev} \approx 0$) and in these cases, the parsing mainly fails because of forgotten or misplaced tokens that break the syntax. In contrast, high values of the $\overline{lev}$ also allow us to identify models producing triples that are far from well-formed. We identified that these high values are generally related to the generation of incomplete, truncated or empty sequences as in $T5_{vT}$, $T5_{vn}$, $T5_{vlf}$, $T5_{vt1}$ and at a higher level $T5^C_{vlf}$, $T5^C_{vgf}$ and $T5^F_{vu}$. Some typical errors illustrating it are reported in Fig 14.

Regarding the **rate of SHACL-Validated Triples** ($\overline{R_{SVT}}$), all the configurations that lead to parsed triples almost perfectly fit the defined $s^*$ maximal SHACL shape. Moreover, the fact that all the $R_{CS}$ scores are close to 1 in

Fig. 13. Averaged syntax performances $F_1^-$ relatively to the $R_{PT}$




Tables 6, 7 and 8 is a sign that the subject URI given in the prompt is easily copied by the model in all the configurations. But we can point out some exceptions: the Turtle Ultra Light syntax for which FlanT5 and PileT5 fail to properly write the subject URI, forgetting, for example, the prefix token ":" or struggling to produce sequences containing special characters. e.g. $pT5_{vu}$ produces "$\_Roland\_G\%C3\%BAr$" instead of ": $Roland\_G\%C3\%BAr$", as well as only "$Margues$" instead of "$Jos\%C3\%A9\_Margues$".

The **BLUE scores** are generally above 0.9, which shows the global ability of the models to predict the right values when the graphs produced are parsed. In particular, for $T5_{vj}$, $BLEU = 1$, meaning that the model always perfectly predicts string values of datatype properties. $T5_{vlf}$, $T5^C_{vT}$ and $T5^C_{vgf}$ perform less well on that specific aspect.

The **training behaviour** meta-metrics all depend on the saturation of a given metrics. Consequently, we cannot evaluate many of the PileT5 and FlanT5 configurations that never reached saturation on both considered metrics ($T5^F_{vu}$ being an exception here). Conversely BART and T5 configurations always saturate both $R_{PT}$ and $F_1^-$ metrics. Some codeT5 configurations have difficulties saturating on both $R_{PT}$ and $F_1^-$, e.g. $T5^C_{vx}$, $T5^C_{vj}$, $T5^C_{vgf}$.

The average **velocity** values $\overline{V_{R_{PT}}}$ and $\overline{V_{F_1^-}}$ demonstrate that all the BART and T5 configurations saturate at the first epoch, except $T5_{vn}$. We checked the value of the $F1_1^-$ at the epoch where $R_{PT}$ saturates (noted $F1_1^-(V_{R_{PT}})$ in Tables 6, 7 and 8), and we observed that all models saturating the $R_{PT}$ also record high $F1_1^-$ values, which means that learning the syntax is the main challenge when it comes to learn an RDF-extractor. Moreover, we observe that the BART models generally saturate both $R_{PT}$ and $F_1^-$ early, followed by the T5 models. From the syntax point of view, TurtleLight written on one line saturates earlier than other syntaxes. The factorisation of list and tag syntaxes leads to earlier saturation, which is particularly noteworthy in the case of the tag syntax configurations. Regarding the W3C syntaxes, there is a clear difference between JSON and XML on one hand, which have a $\overline{V_{R_{PT}}}$ close to 0, and Turtle and NTriples on the other hand, which have a $\overline{V_{R_{PT}}}$ close to 1. This gap could be due to the pre-training of the language models we used, which are more likely to have seen JSON and XML that are common on the web.

The average **stability** and **divergence** values show that all the saturating models are stable from the $F_1^-$ point of view ($\overline{S_{F_1^-}} = 1$) and never diverge ($\sum D_{F_1^-} = 0$). Only the $\overline{S_{R_{PT}}}$ and $\sum D_\rho$ meta-metrics reveal differences in the performances of the models. These metrics are correlated in the sense that an unstable model always diverges. We see that BART models never diverge, while many configurations of T5 diverge.

The **resource metrics** (time and $CO_2$) also show important discrepancies between models, that could be explained by the verbosity of some syntaxes, and the ability of a model to learn a given syntax without divergence. As shown in Fig. 15, T5 models are greedier than BART models, and simple syntaxes are thriftier than RDF ones. Model training $CO_2$ footprint (radius) vary from 0.029g for $B_{vtf}$, to 0.324g for $T_{vx}$. It is also interesting to see that the Turtle Ultra

Fig. 14. Examples of parsing errors accounted

(a) $B_{vx}$ generated triple with a missing "/" token on the closing `<rdf:Description>` element

```
<?xml version="1.0" encoding="utf-8"?>
 <rdf:RDF
  xmlns:dbo="http://dbpedia.org/ontology/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#" >
 <rdf:Description rdf:about="http://dbpedia.org/resource/Steven_Allan_Boggs">
 <rdf:type rdf:resource="http://dbpedia.org/ontology/Person"/>
 <rdfs:label rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Steven Allan Boggs</rdfs:label>
 <dbo:birthDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date">1946-03-15</dbo:birthDate>
 <dbo:birthYear rdf:datatype="http://www.w3.org/2001/XMLSchema#gYear">1946</dbo:birthYear>
 <rdf:Description>
 </rdf:RDF>
```

(b) $T5_{vgf}$ triples with a missing "<rel>" starting tokens

```
Ofelia_Montesco<rel>type<obj>Person<rel>label<obj>Ofelia Montesco
<rel>birthDate<obj>1936-09-10<rel>birthName<obj> Ofelia Irene Grabowski Edery<et>
```

```
dbo: <http://dbpedia.org/ontology/>.
@prefix dbr: <http://dbpedia.org/resource/>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
dbr:Vernell_Brown_Jr%2E  a  dbo:Person ;
  rdfs:label "Vernell Brown Jr."^^xsd:string ;
  dbo:birthDate "1971-08-13"^^xsd:date ;
  dbo:birthYear "1971"^^xsd:gYear.
```

(c) $T5_{vT}$ triples with a missing "@prefix" declaration

(d) $T5^{C}_{vgf}$ triples with a repeated "Cara_Grzesk" pattern

```
zor%C3%A1d<rel>type<obj>Person<rel>label<obj>Martin Boszorád<rel> birthDate<obj>1989-11-13
<rel>birthYear<obj>1989<et><subj> Cara_GrzeskCara_GrzeskCara_GrzeskCara
_GrzeskCara_GrzeskCara_GrzeskCara_GrzeskCara_GrzeskCara_GrzeskCara...
```

Light syntax, despite conciseness and good overall results, finally requires more training and, consequently, more resources.

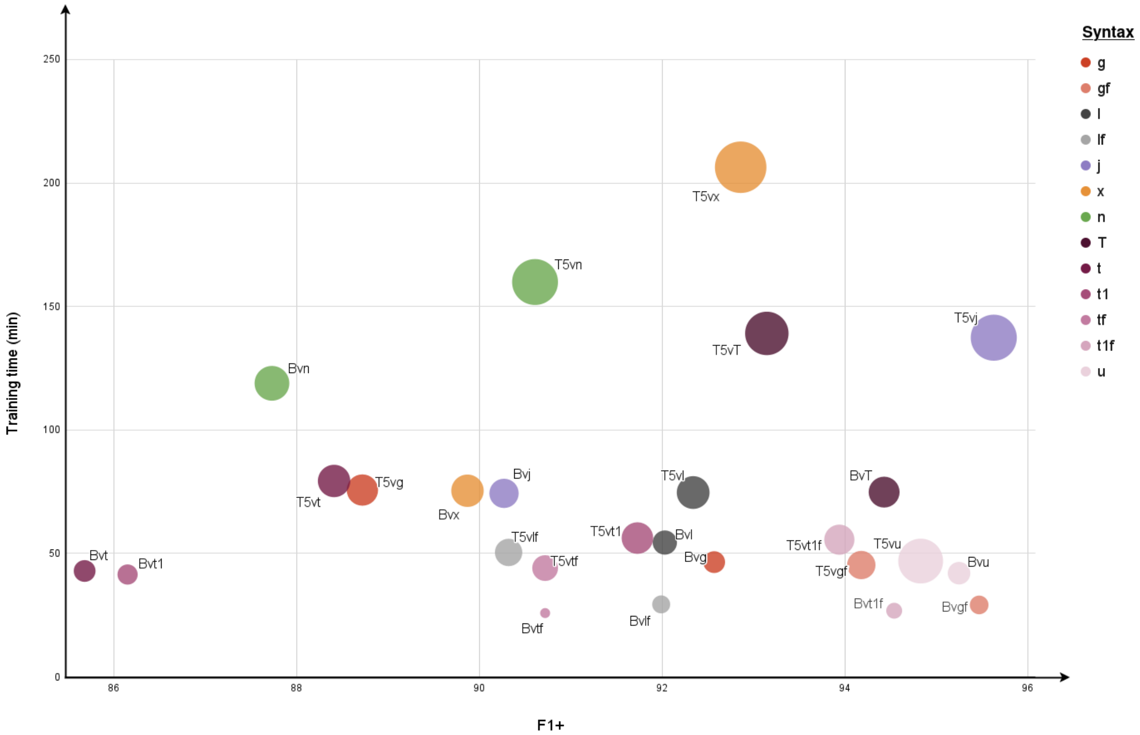
To sum up, our experiments show that the quality of the generated RDF triples depends on both the choice of the model and of the syntax. In terms of models, BART generally writes syntactically better triples than T5, while T5 needs fewer training epochs but requires more resources. Interestingly, codeT5 is in third place without being pre-trained on Wikipedia data. FlanT5 and pileT5 are unable to solve our task.

In terms of syntaxes, factorization applied on the list, tags and Turtle light syntaxes positively impacts the models' performance, except on $T5_{vl}$. The one-line variation improves the quality of Turtle Light variations, and the best configuration combines factorization and one-line writing. In the end, $B_{vtf1}$ (BART with Turtle light, factorization, one-line) offers good performances at a low cost with a standard and human-readable syntax.

## 6. Evaluation of Specific Model Capacities

Two potential solutions can be considered to address the syntactic errors produced by our models. **(1)** The first relies on the use of *constrained decoding* guided by an Extended Backus–Naur Form (EBNF) grammar. As demonstrated by [16], grammar-based decoding can constrain a model to generate outputs that strictly follow a predefined syntax, thereby reducing structural errors. Their work shows that constrained generation can be effectively applied

Fig. 15. Best performing models (BART and T5) configurations against $\overline{F_1^+}$ (x axis), $\overline{T_t}$ (y axis) and $\overline{C_c}$ (radius)




at inference time in the context of large language models (LLMs); however, it remains to be verified whether similar benefits can be achieved for *small language models (SLMs)*.

**(2)** The second solution exploits the capabilities of the *CodeT5* model, which was pretrained on code understanding and correction tasks [75]. This family of models could be leveraged to automatically correct malformed textual sequences that deviate from a target syntax.

### *6.1. Constrained Decoding Based on Grammar*

Following discussions with the maintainers of the *transformers-CFG* library[10] and building on the work of [16], we extended this package to support the BART and T5 models. To evaluate the effectiveness and potential of constrained decoding, we reused our fine-tuned model $B_{vtf1}$ and adapted the EBNF grammar defined by the W3C [11] to manage the proposed *Turtle Light* syntax, including both its one-line and factorized forms.

However, none of our tests yielded conclusive results. In practice, constrained decoding led to a substantial increase in inference time, requiring on average approximately 250 ms per token on a personal computer, which is about six times slower than unconstrained decoding. Moreover, the generated outputs were often incomplete and syntactically incorrect, indicating that grammar-constrained decoding could not be effectively applied within our experimental context.

### *6.2. CodeT5 Coding Capacities*

The **CodeT5** model was pretrained to perform multiple code-related tasks: (1) code summarization, (2) code generation, (3) code refinement and correction, (4) code vulnerability detection (defect detection), and (5) code

[10] https://github.com/epfl-dlab/transformers-CFG/issues/58
[11] https://www.w3.org/TR/turtle/

Fig. 16. EBNF syntax of `t1f` Turtle Light syntax

```
############## turtle light one-line factorized
root ::= triples+
triples ::= WS? triple WS? "."
triple ::= subj WS? predicateObjectList
predicateObjectList ::= pred objectList ( WS?  ";"  WS? ( pred   WS? objectList)? )*
objectList ::= obj  (  WS? ","  WS? obj  )*
subj ::=  iri
pred ::= iri | "a"
obj ::= iri | string
string ::= WS? "\"" [ \t!#-\[\]-~]* "\"" WS?
iri ::= ":" PN_LOCAL+
WS ::= [ \t\n]
PN_CHARS_BASE ::= [A-Z] | [a-z] | [#x00C0-#x00D6] | [#x00D8-#x00F6] | [#x00F8-#x02FF] |
[#x0370-#x037D] | [#x037F-#x1FFF] | [#x200C-#x200D] | [#x2070-#x218F] | [#x2C00-#x2FEF] |
[#x3001-#xD7FF] | [#xF900-#xFDCF] | [#xFDF0-#xFFFD] | [#x10000-#xEFFFF]
PN_CHARS_U ::= PN_CHARS_BASE | "_"
PN_LOCAL ::= ( PN_CHARS_U | ":" | [0-9] | PLX )
( ( PN_CHARS | "." | ":" | PLX )*
( PN_CHARS | ":" | PLX ) ) ?
PLX  ::= PERCENT | PN_LOCAL_ESC
PN_CHARS  ::= PN_CHARS_U | "-" | [0-9] | [#x00B7] | [#x0300-#x036F] | [#x203F-#x2040]
PERCENT  ::= "%" HEX HEX
HEX ::= [0-9] | [A-F] | [a-f]
PN_LOCAL_ESC ::= "\\" ( "_" | "~" | "." | "-" | "!" | "$" | "&" |
"'" | "(" | ")" | "*" | "+" | "," | ";" | "=" | "/" | "?" | "#" | "@" | "%" )
```

Fig. 17. Example of broken sequence generated using the EBNF grammar

*Expected triple:*

```
:Pavel_Kuptsov a a :Person; :label "Pavel Kuptsov"; :birthDate
"1998-01-06"; :birthYear "1998".
```

*Generated triple:*

```
::Pavel_Kuptsov a :Person; ::label a
```

translation [75]. In our case, the *code refinement* capability is of particular interest, as it can help correct syntactic errors within generated outputs. It may also assist in fixing minor encoding inconsistencies that often penalize models from the T5 family.

*Using the Pretrained CodeT5 Directly:* A straightforward approach is to employ the general pretrained CodeT5 model to automatically fix common syntactic or encoding errors observed in our generated outputs. To evaluate this potential, we tested CodeT5 on a recurrent error made by T5-family models when generating Turtle graphs—specifically, the omission of the `@prefix` declaration. We used the following `"Refine:"` prompt to instruct CodeT5 to correct such cases. This preliminary example showed that the pretrained **CodeT5** model is not capable of handling RDF data or understanding the specific syntactic conventions of Turtle. As a result, it was unable to correct even simple structural errors such as missing declarations.

*Using the Finetuned CodeT5 $T5^{C}_{tf1}$ model* : We hypothesized that a fine-tuned model, denoted as $T5^{C}_{tf1}$, could potentially manage such cases within the *Turtle Light* syntax. To evaluate this hypothesis, we reused the same refinement prompt and tested the model on four artificially corrupted versions of an original triple. Each corrupted instance represented a different type of syntactic distortion—such as omitted prefixes, misplaced delimiters, invalid datatype annotations, and missing brackets—allowing us to assess the model's ability to restore the valid graph given in the prompt.

The fine-tuned model appeared capable of correcting two of the syntactic errors illustrated in Fig. 19 and 20. However, this improvement came at a significant cost: a noticeable reduction in the number of triples successfully produced. Regarding the date format, the model indeed generated a syntactically valid correction, but it altered the original value, thereby compromising the factual integrity of the triple. Finally, when instructed to infer the `year` properties from a complete date—as shown in Fig. 22—the model succeeded only in generating the `birthYear` property, while omitting the corresponding `deathYear`.

Fig. 18. Example of codeT5 *Refine* sequence fixing

*Refine:*

```
dbo: <http://dbpedia.org/ontology/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>
dbr:Evelyn_Sterling a dbo:Person ;
    rdfs:label "Evelyn Sterling"^^xsd:string;
    dbo:birthDate "1978-09-12"^^xsd:date ;
    dbo:birthYear "1978"^^xsd:gYear ;
    dbo:deathYear "1978"^^xsd:gYear.
```

*Expected output:*

```
@prefix dbo: <http://dbpedia.org/ontology/> .
@prefix dbr: <http://dbpedia.org/resource/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>
dbr:Evelyn_Sterling a dbo:Person ;
    rdfs:label "Evelyn Sterling"^^xsd:string;
    dbo:birthDate "1978-09-12"^^xsd:date ;
    dbo:birthYear "1978"^^xsd:gYear ;
    dbo:deathYear "1978"^^xsd:gYear.
```

*Generated output:*

```
dbo:deathDate "1978-09-12"^^xsd:date ;dbo:deathDate "1978-09-12"^^xsd:date;
dbo:birthDate "1978-09-12"^^xsd:date ;dbo:deathDate "1978-09-12"^^xsd:date ;dbo:deathDate "1978-09-12"^^xsd:date
```

Fig. 19. Example of $T5^{C}_{tf1}$ *Refine* on a missing dot broken sequence

*Missing terminal dot:*

```
:Erwan_Falwad a :Person; :label "Erwan Falwad"; :birthDate "1730-07-31";
:birthYear "1730"; :deathDate "1793-06-28"; :deathYear "1793"
```

*Generated output:*

```
:Erwan_Falwad a :Person; :label "Erwan Falwad"; :birthDate "1730-01-30";
:birthYear "1730".
```

Fig. 20. Example of $T5^{C}_{vtf1}$ *Refine* applied on a missing ":" broken sequence

*Missing initial ":" token:*

```
Erwan_Falwad a :Person; :label "Erwan Falwad"; :birthDate "1730-07-31";
:birthYear "1730"; :deathDate "1793-06-28"; :deathYear "1793".
```

*Generated output:*

```
:Erwan_Falwad a :Person; :label "Erwan Falwad"; :birthYear "1730".
```

### *6.3. Robustness of the Models to Prompt Noise*

To further evaluate the contextual sensitivity of our models, we investigated how variations in prompt design and input structure affect relation extraction performance.

*Sensitivity to Sentences Order in Abstracts:* Both BART and T5 fine-tuned models demonstrate a strong capacity to extract factual knowledge from reordered abstracts. The BART model, in particular, shows near-perfect robustness to sentence reordering, achieving almost identical performance across the baseline and reordered test sets.

*Impact of Entity URIs in the Prompt:* A recurrent limitation observed in both models concerns their inability to infer entity URIs autonomously, even when the extracted relations are correct. By comparing the *MaskedEntity* and *NoEntity* configurations, we observe that omitting the entity entirely yields better performance than masking it. This suggests that explicit masking introduces noise into the encoding process, whereas the absence of the entity allows the model to generalize more effectively.

Fig. 21. Example of $T5^{C}_{vtf1}$ *Refine* on malformed date

*Date format error:*

```
:Erwan_Falwad a :Person; :label "Erwan Falwad"; :birthDate "1730-0731";
:birthYear "1730"; :deathDate "1793-06-28"; :deathYear "1793".
```

*Generated output:*

```
:Erwan_Falwad a :Person; :label "Erwan Falwad"; :birthDate "1730-07-02";
:birthYear "1730".
```

Fig. 22. Example of $T5^{C}_{vtf1}$ *Refine* on sequence missing the year properties

*Missing Year properties:*

```
:Erwan_Falwad a :Person; :birthDate "1730-07-31"; :deathDate "1793-06-28".
```

*Generated output:*

```
:Erwan_Falwad a :Person; :label "Erwan Falwad"; :birthDate "1730-01-30";
:birthYear "1730".
```

*Impact of Explicit Instructions in T5:* Although the T5 model was fine-tuned to produce triples directly from Wikipedia abstracts, the presence of an explicit instruction prompt (e.g., "`Extract RDF triples from the following abstract:`") remains essential for high-quality extraction. When this instruction was removed, performance degraded sharply, as shown in the "NoInstruct" configuration.

Table 5

Impact of context variations on BART and T5 performances.

| **Model** | **Test Set** | $F1_-$ | $F1_+$ | $R_{PT}$ | $R_{CS}$ | $R_{SVT}$ | **Gg** |
|---|---|---|---|---|---|---|---|
| $B_{vtf1}$ | Baseline | 98.86 | 78.22 | 1.00 | 1.00 | 1.00 | 78.22 |
| $B_{vtf1}$ | ReOrder | 98.65 | 80.23 | 1.00 | 1.00 | 1.00 | 80.23 |
| $B_{vtf1}$ | MaskedEntities | 90.72 | 72.48 | 1.00 | 0.00 | 1.00 | 0.00 |
| $B_{vtf1}$ | NoEntity | 97.39 | 74.42 | 1.00 | 0.00 | 1.00 | 0.00 |
| $T5_{vtf1}$ | Baseline | 88.56 | 68.84 | 0.90 | 0.89 | 0.90 | 49.57 |
| $T5_{vtf1}$ | ReOrder | 88.68 | 73.67 | 0.90 | 0.88 | 0.90 | 52.34 |
| $T5_{vtf1}$ | MaskedEntities | 84.27 | 66.21 | 0.90 | 0.00 | 0.90 | 0.00 |
| $T5_{vtf1}$ | NoEntities | 86.93 | 67.89 | 0.90 | 0.00 | 0.90 | 0.00 |
| $T5_{vtf1}$ | NoInstruct | 70.44 | 54.78 | 0.70 | 0.00 | 0.70 | 0.00 |

## 7. Discussion

Our benchmark first demonstrated that all encoders-decoders are highly sensitive to the choice of a syntax for the RE task. Moreover, we observed that the choice of the model itself is crucial in terms of performance. Bart is supposed to be trained on plain text, where the T5 family models were all trained on noisier sources, potentially containing XML or JSON input data. These specificities related to the pretraining corpora help explain why T5 is better at handling JSON and XML than BART, but these good performances come at a higher training cost for T5. In that context, we showed that the closer the syntax is to natural language, the easier it is for a pretrained model to learn it.

Many hypotheses can be drawn to explain these discrepancies. (1) BART was pretrained on a larger set of basic subtasks, which may encourage the flexibility of the model. (2) The multitask learning design of the T5 family enables the resolution of high-level objectives. However, all of these predefined tasks expect short, or natural language answers that are far from being structured, which may, in our context, penalise these models.

Considering the supplementary experiment we conducted on the usage of grammar-based constrained generation, we showed that this solution does not significantly help encoder-decoder-based models generate well-formed output. First, the constrained generation incurs a significant cost at inference time, and the actual library developed struggles to handle complex grammars. Aside from this, the experiments conducted with codeT5 showed that the model's refinement capacities are limited when used directly. Nonetheless, these results are encouraging since the models are producing parsable triples and can still be further finetuned using explicit examples of refinement.

Of course, one potential explanation for the good results of T5 and BART may be a memorisation effect. In a recent work [62], we tested this hypothesis. We finetuned BART models using only facts that were already seen during the model's pretraining, i.e., those existing in the 2021 Wikipedia dumps. We then tested it on two sets of datasets: one comprising Wikipedia pages published before 2021 and the other comprising new Wikipedia pages published after this date. Our results showed no significant difference between these two configurations, which allows us to refute the memorisation hypothesis today.

## 8. Conclusion: BART and a Turtle Light Go a Long Way

In this article, we evaluated the impact of the choice of a syntax on the fine-tuning of small language models for the generation of RDF triples, focusing on extracting datatype properties from text. We demonstrated the ability of the BART and T5 models to solve the RE task compared to codeT5, pileT5 and FlanT5. To do so, we proposed several metrics that allowed us to characterise the behaviours of the given training configuration. Our results show that syntax understanding is the main challenge language models face when they are finetuned to solve a relation extraction task: all the configurations able to generate well-written graphs also highly perform in terms of $F_1^-$. Moreover, the choice of the syntax also has a significant impact on the performance of the extraction, as well as on the resources needed to learn it (time and $CO_2$ footprint). Basic syntaxes (list and tags) are generally easily learned but lead to average performances. While learning W3C RDF syntaxes is more resource-consuming, and the RDF potential (managing ontologies, datatypes) must generally be paid by a higher cost in terms of resource. The best-performing configuration $T5_{vj}$ (T5 with JSON) outperforms the others at the cost of 2 hours of training on an A100 GPU and a 0.250g $CO_2$ footprint. An interesting compromise is the use of simplified standard syntaxes (as the Turtle Light syntaxes proposed in this work) that are robust and quick to learn. However, despite its simplicity, we also show that the Turtle Ultra Light syntax (Turtle variation omitting the prefixes and their declarations) could be costly to learn. For all these reasons, the finetuning of BART models using the **inline factorised Turtle Light** ($B_{vtf1}$) is a good tradeoff between efficiency and frugality.

Beyond syntax optimization, we evaluated the capacity of models to handle and repair syntactic errors. Constrained decoding using EBNF grammars, although theoretically promising, proved impractical in our context due to a sixfold increase of inference time and incomplete outputs. In contrast, experiments with the CodeT5 model and its fine-tuned variant ($T5^C_{vtf1}$) showed partial success in correcting malformed triples within the Turtle Light syntax, demonstrating potential for integrating code-oriented pretraining into graph refinement tasks. However, these approaches also revealed a trade-off between syntactic correction and semantic fidelity: models can repair structure but occasionally corrupt values or omit facts.

Finally, our study of **contextual robustness** emphasized the prompt sensitivity of the models analysed. Both BART and T5 remained stable under reordered textual inputs, confirming limited order sensitivity. However, entity masking and missing led to degraded results, underlying some limitations related to the fine-tuning models since they could not generalize enough to be able to guess the subject of the graph, when conditioned to use it from the prompt to compose the output. Moreover we saw that the instruction also has an important impact on the T5 model, underlying the importance to define a new instruction for a new task, but also to refer to the right instruction when the model is used for inference.

In future work, we will focus on systematizing the task proposed at a Knowledge base scale, notably by extending our proposal to Shape containing both data and object properties.

***Acknowledgments*** This work has been supported by the French government through the 3IA Côte d'Azur Investments (ANR-23-IACL-0001) and by the UCAJEDI (ANR-15-IDEX-01), the OPAL infrastructure and Université Côte d'Azur's Center for High-Performance Computing.

Table 6 First part of the results for the best-performing configurations. Configurations are listed in descending order of $G_g$. In bold are the best results. In italics are the second-best results. The worse results are underlined. As a reminder the syntax notation is: XML-RDF ($_x$), Turtle ($_T$), Turtle Light ($_t$), Turtle Ultra Light ($_u$), N-Triples ($_n$), JSON-LD ($_j$), list ($_l$) and tags ($_g$).

| Rank | Config | Triple Validity | | | RE performance ×100 | | | | Edition m. | | Training behavior | | | | | | | | Resources | | $G_g$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\overline{R_{PT}}$ | $\overline{R_{CS}}$ | $\overline{R_{SVT}}$ | $\overline{F_1^-}$ | $\overline{F_1^+}$ | $\overline{P^+}$ | $\overline{R^+}$ | $\overline{BLEU}$ | $\overline{lev}$ | $\overline{Nb_{epochs}}$ | $\overline{V_{R_{PT}}}$ | $\overline{S_{R_{PT}}}$ | $\sum D_{R_{PT}}$ | $\overline{V_{F_1^-}}$ | $\overline{S_{F_1^-}}$ | $\sum D_{F_1^-}$ | $\overline{F_1^-(V_{R_{PT}})}$ | $\overline{C_c}$ | $\overline{T_t}$ | |
| 1 | $T5_{vj}$ | **1,00** | **1,00** | **1,00** | *99,75* ± 0.3 | **95,63** ± 8.1 | **100,00** | **94,37** | **1** | **0,000** | 13 | *0,2* | 0,973 | **0** | *0,2* | **1,000** | **0** | *98,91* | 0,252 | 137,362 | **96** |
| 2 | $B_{vgf}$ | **1,00** | **1,00** | **1,00** | 99,69 ± 0.4 | *95,47* ± 8.1 | 99,29 | *94,28* | *0,967* | **0,000** | 15 | **0,0** | **1,000** | **0** | **0,0** | **1,000** | **0** | 98,59 | 0,042 | 29,149 | *95* |
| 3 | $B_{vu}$ | **1,00** | **1,00** | **1,00** | **99,79** ± 0.3 | 95,25 ± 6.7 | *99,41* | 94,04 | 0.970 | **0,000** | 22 | **0,0** | **1,000** | **0** | **0,0** | **1,000** | **0** | 98,78 | 0,056 | 42,000 | *95* |
| 4 | $T5_{vu}$ | **1,00** | **1,00** | **1,00** | 99,63 ± 0.6 | *94,83* ± 7.7 | 99,01 | 93,53 | 0,967 | **0,000** | 15 | **0,0** | **1,000** | **0** | **0,0** | **1,000** | **0** | 98,90 | 0,236 | 46,905 | *95* |
| 5 | $B_{vt1f}$ | **1,00** | **1,00** | **1,00** | 99,72 ± 0.3 | *94,54* ± 7.3 | 97,09 | 93,20 | 0,933 | **0,000** | *12* | **0,0** | **1,000** | **0** | **0,0** | **1,000** | **0** | 98,80 | *0,035* | *26,801* | *95* |
| 6 | $B_{vT}$ | **1,00** | **1,00** | **1,00** | 99,73 ± 0.4 | *94,43* ± 8.2 | 96,39 | 93,42 | 0,972 | 11,750 | 23 | **0,0** | **1,000** | **0** | **0,0** | **1,000** | **0** | 98,68 | 0,104 | 74,831 | 94 |
| 7 | $T5_{vt1f}$ | **1,00** | **1,00** | **1,00** | 99,51 ± 0.6 | 93,94 ± 9.7 | 95,48 | 93,13 | 0,961 | **0,000** | 14 | *0,2* | **1,000** | **0** | **0,0** | **1,000** | **0** | **98,98** | 0,099 | 55,563 | 94 |
| 8 | $T5_{vx}$ | **1,00** | **1,00** | **1,00** | 99,58 ± 0.5 | 92,86 ± 7.3 | 96,81 | 91,91 | 0,946 | 1,600 | 18 | 0,4 | 0,990 | **0** | 0,4 | **1,000** | **0** | 98,09 | 0,324 | 206,378 | 93 |
| 9 | $B_{vg}$ | **1,00** | **1,00** | **1,00** | 99,62 ± 0.4 | 92,57 ± 8.8 | 96,34 | 91,08 | 0,940 | **0,000** | 18 | **0,0** | **1,000** | **0** | **0,0** | **1,000** | **0** | 98,42 | 0,053 | 46,486 | 93 |
| 10 | $T5_{vl}$ | **1,00** | **1,00** | **1,00** | 99,55 ± 0.6 | 92,34 ± 9.4 | 95,19 | 91,40 | 0,972 | *0,200* | *11* | 0,8 | **1,000** | **0** | 0,6 | **1,000** | **0** | 98,27 | 0,118 | 74,664 | 92 |
| 11 | $B_{vlf}$ | **1,00** | **1,00** | **1,00** | 99,63 ± 0.4 | 91,99 ± 9.4 | 96,68 | 90,49 | *0,996* | 1,650 | *12* | **0,0** | **1,000** | **0** | **0,0** | **1,000** | **0** | 98,49 | 0,040 | 29,360 | 92 |
| 12 | $B_{vl}$ | **1,00** | **1,00** | **1,00** | 99,62 ± 0.4 | 92,03 ± 8.6 | 94,75 | 90,37 | 0,896 | 18,086 | 18 | **0,0** | **1,000** | **0** | **0,0** | **1,000** | **0** | 98,23 | 0,064 | 54,408 | 92 |
| 13 | $B_{vtf}$ | **1,00** | **1,00** | **1,00** | 99,49 ± 0.6 | 90,72 ± 10.7 | 95,45 | 89,13 | 0,968 | **0,000** | 12 | **0,0** | **1,000** | **0** | **0,0** | **1,000** | **0** | 98,33 | **0,029** | **25,849** | 91 |
| 14 | $T5_{vgf}$ | **1,00** | 0,96 | **1,00** | 99,57 ± 0.4 | *94,18* ± 7.8 | 96,76 | 92,51 | 0,983 | 1,400 | 13 | *0,2* | 0,975 | **0** | *0,2* | **1,000** | **0** | 98,49 | 0,087 | 45,302 | 90 |
| 15 | $T5_{vtf}$ | **1,00** | **1,00** | **1,00** | 99,33 ± 0.7 | 90,72 ± 9.8 | 95,81 | 88,72 | 0,955 | 0,560 | **10** | 0,8 | 0,986 | **0** | **0,0** | **1,000** | **0** | 93,54 | 0,072 | 44,072 | 90 |
| 16 | $B_{vj}$ | **1,00** | **1,00** | **1,00** | 99,52 ± 0.5 | 90,27 ± 11.3 | 95,14 | 88,85 | 0,962 | 47,200 | *11* | *0,2* | **1,000** | **0** | **0,0** | **1,000** | **0** | 98,88 | 0,093 | 74,324 | 90 |
| 17 | $B_{vx}$ | **1,00** | **1,00** | **1,00** | 99,46 ± 0.7 | 89,87 ± 13 | 96,69 | 88,85 | 0,974 | 17,400 | 14 | **0,0** | **1,000** | **0** | **0,0** | **1,000** | **0** | 96,72 | 0,115 | 75,387 | 90 |
| 18 | $T5_{vt1}$ | 0,97 | **1,00** | **1,00** | 99,34 ± 0.7 | 91,73 ± 9.8 | 95,32 | 89,68 | 0,968 | 99,400 | **10** | **0,0** | 0,980 | **0** | **0,0** | **1,000** | **0** | 97,36 | 0,109 | 56,223 | 89 |
| 19 | $T5_{vlf}$ | **1,00** | 0,98 | **1,00** | 99,32 ± 0.7 | 90,32 ± 10.9 | 94,28 | 89,43 | 0,875 | 205,511 | *11* | 1,0 | 0,960 | 1 | **0,0** | **1,000** | **0** | 98,71 | 0,081 | 50,397 | 88 |
| 20 | $B_{vn}$ | **1,00** | **1,00** | **1,00** | 99,36 ± 0.6 | 87,73 ± 9.5 | 97,01 | 85,48 | 0,986 | 81,800 | 24 | **0,0** | *0,991* | **0** | 0,4 | **1,000** | **0** | 95,96 | 0,134 | 118,891 | 88 |
| 21 | $T5_{vg}$ | *0,98* | **1,00** | **1,00** | 99,29 ± 0.6 | 88,72 ± 9.6 | 96,15 | 86,28 | 0,943 | 18,305 | 16 | 0,4 | 0,970 | **0** | **0,0** | **1,000** | **0** | 98,58 | 0,107 | 75,670 | 87 |
| 22 | $T5_{vt}$ | 0,97 | **1,00** | **1,00** | 99,25 ± 0.5 | 88,41 ± 11.4 | 92,60 | 86,61 | 0,935 | 29,200 | 14 | *0,2* | **1,000** | **0** | **0,0** | **1,000** | **0** | 98,65 | 0,115 | 79,342 | 85 |
| 23 | $B_{vt1}$ | 0,97 | **1,00** | **1,00** | 99,32 ± 0.5 | 86,15 ± 11.0 | 94,13 | 83,89 | 0,993 | 20,000 | 16 | **0,0** | **1,000** | **0** | **0,0** | **1,000** | **0** | 97,91 | 0,047 | 41,463 | 83 |
| 24 | $B_{vt}$ | 0,97 | **1,00** | **1,00** | 99,34 ± 0.5 | 85,68 ± 12.2 | 93,52 | 83,64 | 0,975 | 12,933 | 15 | **0,0** | **1,000** | **0** | **0,0** | **1,000** | **0** | 97,44 | 0,053 | 42,898 | 83 |

Table 7 Second part of the results for the best-performing configurations. Configurations are listed in descending order of $G_g$. In bold are the best results. In italics are the second-best results. The worse results are underlined. As a reminder the syntax notation is: XML-RDF ($_x$), Turtle ($_T$), Turtle Light ($_t$), Turtle Ultra Light ($_u$), N-Triples ($_n$), JSON-LD ($_j$), list ($_l$) and tags ($_g$).

| Rank | Config | Triple Validity | | | RE performance ×100 | | | | Edition m. | | Training behavior | | | | | | | | Resources | | $G_g$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\overline{R_{PT}}$ | $\overline{R_{CS}}$ | $\overline{R_{SVT}}$ | $\overline{F_1^-}$ | $\overline{F_1^+}$ | $\overline{P^+}$ | $\overline{R^+}$ | $\overline{BLEU}$ | $\overline{lev}$ | $\overline{Nb_{epochs}}$ | $\overline{V_{R_{PT}}}$ | $\overline{S_{R_{PT}}}$ | $\sum D_{R_{PT}}$ | $\overline{V_{F_1^-}}$ | $\overline{S_{F_1^-}}$ | $\sum D_{F_1^-}$ | $\overline{F_1^-(V_{R_{PT}})}$ | $\overline{C_c}$ | $\overline{T_t}$ | |
| 25 | $T5^C_{vT}$ | **1,00** | 0,98 | **1,00** | 98,82 ± 0.8 | 83,47 ± 9.7 | 87,99 | 82,15 | 0,877 | 269,573 | *12* | 1,6 | 0,923 | **0** | *0,2* | **1,000** | **0** | 97,68 | 0,135 | 80,982 | **82** |
| 26 | $T5^C_{vt1f}$ | **1,00** | 0,98 | **1,00** | 98,51 ± 0.4 | 83,18 ± 7.9 | 91,51 | 81,64 | 0,929 | 19,789 | 14 | *0,2* | 0,913 | **0** | **0,0** | **1,000** | **0** | 98,14 | *0,082* | *55,403* | *81* |
| 27 | $T5^C_{vl}$ | *0,97* | *0,99* | **1,00** | 98,55 ± 0.8 | 83,21 ± 9.1 | 89,41 | 81,25 | 0,930 | 47,107 | 17 | 0,6 | 0,949 | *1* | **0,0** | **1,000** | **0** | 94,19 | 0,138 | 89,753 | 79 |
| 28 | $T5^C_{vt1}$ | *0,97* | 0,98 | **1,00** | 98,29 ± 0.4 | 82,02 ± 7.9 | 89,42 | 78,81 | 0,927 | 122,218 | 15 | **0,0** | 0,900 | **0** | **0,0** | **1,000** | **0** | 94,70 | 0,131 | 77,993 | *78* |
| 29 | $T5^C_{vlf}$ | 0,94 | 0,97 | **1,00** | 98,60 ± 0.6 | 83,68 ± 7.4 | 92,13 | 81,59 | 0,881 | 80844,715 | 19 | *0,2* | 0,741 | 2 | **0,0** | **1,000** | **0** | 96,77 | 0,101 | 73,919 | 77 |
| 30 | $T5_{vT}$ | 0,82 | **1,00** | **1,00** | *99,38* ± 0.8 | **93,15** ± 10.3 | 95,55 | **91,93** | 0,968 | 810,135 | 15 | 0,8 | 0,620 | 2 | 0,4 | **1,000** | **0** | *98,62* | 0,221 | 139,095 | *76* |
| 31 | $T5^C_{vtf}$ | 0,96 | 0,95 | **1,00** | 97,66 ± 1.8 | 80,41 ± 11.2 | 85,87 | 80,08 | 0,911 | 208,583 | 13 | 1,2 | 0,925 | *1* | **0,0** | **1,000** | **0** | 95,11 | 0,083 | *54,966* | 73 |
| 32 | $T5_{vn}$ | 0,75 | *0,99* | **1,00** | **99,39** ± 0.4 | *90,61* ± 7.9 | **97,98** | *88,17* | 0,972 | 137,233 | 13 | 1,6 | 0,747 | 2 | *0,2* | **1,000** | **0** | 97,68 | 0,250 | 159,895 | 67 |
| 33 | $T5^C_{vt}$ | 0,82 | *0,99* | **1,00** | 98,51 ± 0.8 | 78,92 ± 7.8 | 86,83 | 76,55 | *0,977* | 40,978 | 17 | *0,2* | 0,813 | 2 | **0,0** | **1,000** | **0** | 97,26 | 0,174 | 90,203 | 64 |
| 34 | $T5^C_{vn}$ | 0,84 | *0,99* | **1,00** | 98,41 ± 0.4 | 74,54 ± 7.9 | 84,07 | 73,27 | 0,949 | 421,512 | *12* | 1,8 | *0,857* | 2 | 1,0 | **1,000** | **0** | 97,96 | 0,220 | 139,832 | 62 |
| 35 | $T5^C_{vg}$ | 0,70 | *0,99* | **1,00** | 98,58 ± 0.8 | 81,41 ± 8.4 | 87,09 | 79,85 | 0,926 | 71,099 | 17 | 2,6 | **0,65** | 3 | **0,0** | **1,000** | **0** | 97,73 | 0,119 | 83,253 | 56 |
| 36 | $T5^C_{vgf}$ | 0,61 | 0,97 | **1,00** | 97,12 ± 1.9 | 77,73 ± 7.8 | 85,89 | 76,72 | 0,882 | 287087,870 | *11* | ∅ | ∅ | ∅ | **0,0** | **1,000** | **0** | ∅ | **0,079** | **44,384** | 47 |
| 37 | $T5^C_{vu}$ | 0,92 | 0,41 | *0,99* | 98,84 ± 0.8 | 87,83 ± 8.9 | 91,33 | 86,54 | 0,927 | 77,72 | 16 | *0,2* | 0,89 | *1* | **0,0** | **1,000** | **0** | 97,66 | 0,109 | 57,632 | 32 |
| 38 | $T5^C_{vj}$ | 0,59 | **1,00** | **1,00** | 59,60 ± 54.4 | 52,11 ± 47.7 | 60,00 | 49,64 | ∅ | **17,800** | 14 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 0,174 | 132,535 | 31 |
| 39 | $T5^P_{vt1}$ | 0,39 | **1,00** | **1,00** | 79,27 ± 0.9 | 72,61 ± 13.8 | 76,25 | 71,24 | ∅ | 33,056 | **10** | ∅ | ∅ | ∅ | **0,0** | **1,000** | **0** | ∅ | 0,159 | 77,297 | 28 |
| 40 | $T5^F_{vlf}$ | 0,33 | **1,00** | **1,00** | 59,67 ± 0.6 | 53,78 ± 7.4 | 55,72 | 52,95 | ∅ | 66,548 | 14 | *0,2* | 0,74 | 2 | **0,0** | **1,000** | **0** | **98,77** | 0,115 | 69,921 | 18 |
| 41 | $T5^F_{vu}$ | 0,94 | 0,251 | 0,77 | 88,49 ± 10.0 | 85,59 ± 8.9 | 96,40 | 79,16 | 0,970 | 7042933 | 12 | **0,0** | 0,89 | *1* | **0,0** | **1,000** | **0** | 96,07 | 0,083 | *42,996* | 15 |
| 42 | $T5^C_{vx}$ | 0,26 | 0,97 | **1,00** | 59,59 ± 54.4 | 43,92 ± 44.3 | 46,08 | 42,85 | ∅ | *18,316* | 18 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 0,205 | 168,913 | 11 |
| 43 | $T5^P_{vn}$ | 0,04 | *0,99* | **1,00** | 97,19 ± 4.1 | 79,29 ± 12.9 | 81,01 | *79,36* | **0,983** | 67,679 | 18 | ∅ | ∅ | ∅ | 0,4 | **1,000** | **0** | ∅ | 0,279 | 241,861 | 3 |
| 44 | $T5^P_{vu}$ | 0,59 | 0,00 | 0,87 | 93,32 ± 7.6 | 90,29 ± 7.2 | *97,37* | 87,00 | 0,972 | 69,29 | 18 | ∅ | ∅ | ∅ | **0,0** | **1,000** | **0** | ∅ | 0,122 | 76,741 | 0 |
| 45 | $T5^P_{vt1f}$ | 0,02 | **1,00** | **1,00** | 79,84 ± 44.6 | 61,59 ± 39.8 | 62,86 | 60,86 | ∅ | 55,849 | 16 | ∅ | ∅ | ∅ | 2,6 | **1,000** | **0** | ∅ | 0,109 | 71,955 | 0 |
| 46 | $T5^P_{vg}$ | 0,00 | **1,00** | **1,00** | 39,51 ± 54.1 | 31,28 ± 43.1 | 31,14 | 31,43 | ∅ | 21,812 | *11* | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 0,113 | 69,022 | 0 |
| 47 | $T5^P_{vlf}$ | 0,00 | **1,00** | **1,00** | 20,00 ± 44.7 | 8,57 ± 19.2 | 8,57 | 8,57 | ∅ | 23,183 | 13 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | *0,088* | 73,299 | 0 |

Table 8 Third part of the results for the best-performing configurations. In bold are the best results. In italics are the second-best results. The worse results are underlined. As a reminder the syntax notation is: XML-RDF ($_x$), Turtle ($_T$), Turtle Light ($_t$), Turtle Ultra Light ($_u$), N-Triples ($_n$), JSON-LD ($_j$), list ($_l$) and tags ($_g$).

| Rank | Config | Triple Validity | | | RE performance ×100 | | | | Edition m. | | Training behavior | | | | | | | | Resources | | $G_g$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $\overline{R_{PT}}$ | $\overline{R_{CS}}$ | $\overline{R_{SVT}}$ | $\overline{F_1^-}$ | $\overline{F_1^+}$ | $\overline{P^+}$ | $\overline{R^+}$ | $\overline{BLEU}$ | $\overline{lev}$ | $\overline{Nb_{epochs}}$ | $\overline{V_{R_{PT}}}$ | $\overline{S_{R_{PT}}}$ | $\sum D_{R_{PT}}$ | $\overline{V_{F_1^-}}$ | $\overline{S_{F_1^-}}$ | $\sum D_{F_1^-}$ | $\overline{F_1^-(V_{R_{PT}})}$ | $\overline{C_c}$ | $\overline{T_t}$ | |
| 48 | $T5^F_{vg}$ | **0,29** | 0,00 | 0,00 | **79,68** ± 54.1 | **72,56** ± 43.1 | **74,07** | **71,54** | ∅ | **12,462** | **11** | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | *0,087* | *59,423* | 0 |
| 49 | $T5^F_{vt1f}$ | *0,26* | 0,00 | 0,00 | *59,43* ± 54.3 | 51,66 ± 47.8 | *52,96* | *51,78* | ∅ | 25,253 | 13 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | *0,080* | *55,165* | 0 |
| 50 | $T5^F_{vgf}$ | 0,17 | 0,00 | 0,00 | 39,92 ± 54.7 | 39,51 ± 54.1 | 39,95 | 39,16 | ∅ | 47,084 | 13 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | **0,067** | **45,982** | 0 |
| 51 | $T5^P_{vt}$ | 0,15 | 0,00 | 0,00 | *59,67* ± 54.5 | 40,58 ± 37.6 | 42,78 | 40,29 | ∅ | 28,467 | 15 | ∅ | ∅ | ∅ | 0,6 | **1,000** | **0** | ∅ | 0,154 | 100,197 | 0 |
| 52 | $T5^F_{vtf}$ | 0,01 | 0,00 | 0,00 | 39,63 ± 54.3 | 29,28 ± 40.1 | 31,35 | 28,75 | ∅ | 24,247 | 14 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | *0,089* | 61,239 | 0 |
| 53 | $T5^P_{vx}$ | 0,00 | 0,00 | 0,00 | 0,00 ± 0.0 | 0,00 ± 0.0 | 0,00 | 0,00 | ∅ | 384,492 | 13 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 0,250 | 164,578 | 0 |
| 54 | $T5^P_{vtf}$ | 0,00 | 0,00 | 0,00 | 0,00 ± 0.0 | 0,00 ± 0.0 | 0,00 | 0,00 | ∅ | 21,538 | 15 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 0,124 | 77,378 | 0 |
| 55 | $T5^P_{vT}$ | 0,00 | 0,00 | 0,00 | 0,00 ± 0.0 | 0,00 ± 0.0 | 0,00 | 0,00 | ∅ | 107,384 | *12* | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 0,137 | 114,290 | 0 |
| 56 | $T5^P_{vl}$ | 0,00 | 0,00 | 0,00 | 0,00 ± 0.0 | 0,00 ± 0.0 | 0,00 | 0,00 | ∅ | 17,443 | 14 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 0,189 | 108,608 | 0 |
| 57 | $T5^P_{vj}$ | 0,00 | 0,00 | 0,00 | 0,00 ± 0.0 | 0,00 ± 0.0 | 0,00 | 0,00 | ∅ | 79,172 | 14 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 0,403 | 200,460 | 0 |
| 58 | $T5^P_{vgf}$ | 0,00 | 0,00 | 0,00 | 0,00 ± 0.0 | 0,00 ± 0.0 | 0,00 | 0,00 | ∅ | 20,584 | 14 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 0,109 | 62,134 | 0 |
| 59 | $T5^F_{vx}$ | 0,00 | 0,00 | 0,00 | 0,00 ± 0.0 | 0,00 ± 0.0 | 0,00 | 0,00 | ∅ | 310,462 | 17 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 0,270 | 186,559 | 0 |
| 60 | $T5^F_{vt1}$ | 0,00 | 0,00 | 0,00 | 0,00 ± 0.0 | 0,00 ± 0.0 | 0,00 | 0,00 | ∅ | 118,134 | 17 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 0,183 | 89,052 | 0 |
| 61 | $T5^F_{vT}$ | 0,00 | 0,00 | 0,00 | 0,00 ± 0.0 | 0,00 ± 0.0 | 0,00 | 0,00 | ∅ | 195,167 | *12* | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 0,217 | 114,521 | 0 |
| 62 | $T5^F_{vt}$ | 0,00 | 0,00 | 0,00 | 0,00 ± 0.0 | 0,00 ± 0.0 | 0,00 | 0,00 | ∅ | 97,534 | 15 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 0,122 | 83,726 | 0 |
| 63 | $T5^F_{vn}$ | 0,00 | 0,00 | 0,00 | 0,00 ± 0.0 | 0,00 ± 0.0 | 0,00 | 0,00 | ∅ | 97,423 | 13 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 0,263 | 160,633 | 0 |
| 64 | $T5^F_{vl}$ | 0,00 | 0,00 | 0,00 | 0,00 ± 0.0 | 0,00 ± 0.0 | 0,00 | 0,00 | ∅ | 18,166 | 16 | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 0,169 | 107,316 | 0 |
| 65 | $T5^F_{vj}$ | 0,00 | 0,00 | 0,00 | 0,00 ± 0.0 | 0,00 ± 0.0 | 0,00 | 0,00 | ∅ | 77,319 | **11** | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | ∅ | 0,160 | 115,238 | 0 |

## References


[1] D. Banerjee, P.A. Nair, J.N. Kaur, R. Usbeck and C. Biemann, Modern Baselines for SPARQL Semantic Parsing, in: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22, Association for Computing Machinery, New York, NY, USA, 2022, pp. 2260–2265. ISBN 978-1-4503-8732-3. doi:10.1145/3477495.3531841.

[2] D. Banerjee, P. Nair, R. Usbeck and C. Biemann, The Role of Output Vocabulary in T2T LMs for SPARQL Semantic Parsing, in: *Findings of the Association for Computational Linguistics: ACL 2023*, A. Rogers, J. Boyd-Graber and N. Okazaki, eds, Association for Computational Linguistics, Toronto, Canada, 2023, pp. 12219–12228. doi:10.18653/v1/2023.findings-acl.774. https://aclanthology.org/2023.findings-acl.774.

[3] H.W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, Y. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S.S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, A. Castro-Ros, M. Pellat, K. Robinson, D. Valter, S. Narang, G. Mishra, A. Yu, V. Zhao, Y. Huang, A. Dai, H. Yu, S. Petrov, E.H. Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q.V. Le and J. Wei, Scaling Instruction-Finetuned Language Models, 2022. https://arxiv.org/abs/2210.11416.

[4] J. Dagdelen, A. Dunn, S. Lee, N. Walker, A.S. Rosen, G. Ceder, K.A. Persson and A. Jain, Structured information extraction from scientific text with large language models, *Nature Communications* **15**(1) (2024), 1418. doi:10.1038/s41467-024-45563-x. https://www.nature.com/articles/s41467-024-45563-x.

[5] J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, in: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran and T. Solorio, eds, Association for Computational Linguistics, Minneapolis, Minnesota, 2019, pp. 4171–4186. doi:10.18653/v1/N19-1423. https://aclanthology.org/N19-1423.

[6] N. Ding, Y. Qin, G. Yang, F. Wei, Z. Yang, Y. Su, S. Hu, Y. Chen, C.-M. Chan, W. Chen, J. Yi, W. Zhao, X. Wang, Z. Liu, H.-T. Zheng, J. Chen, Y. Liu, J. Tang, J. Li and M. Sun, Parameter-efficient fine-tuning of large-scale pre-trained language models, *Nat Mach Intell* **5**(3) (2023), 220–235, Number: 3 Publisher: Nature Publishing Group. doi:10.1038/s42256-023-00626-4. https://www.nature.com/articles/s42256-023-00626-4.

[7] J. Dodge, M. Sap, A. Marasović, W. Agnew, G. Ilharco, D. Groeneveld, M. Mitchell and M. Gardner, Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus, arXiv, 2021, arXiv:2104.08758 [cs]. doi:10.48550/arXiv.2104.08758. http://arxiv.org/abs/2104.08758.

[8] S. Efeoglu and A. Paschke, Retrieval-Augmented Generation-based Relation Extraction, *ArXiv* **abs/2404.13397** (2024). https://api.semanticscholar.org/CorpusID:269292881.

[9] H. Elsahar, P. Vougiouklis, A. Remaci, C. Gravier, J. Hare, E. Simperl and F. Laforest, T-REx: A Large Scale Alignment of Natural Language with Knowledge Base Triples, in: *LREC2018*, European Language Resources Association (ELRA), Miyazaki, Japan, 2018, p. nan. https://aclanthology.org/L18-1544.

[10] H. Elsahar, P. Vougiouklis, A. Remaci, C. Gravier, J. Hare, F. Laforest and E. Simperl, T-REx: A Large Scale Alignment of Natural Language with Knowledge Base Triples, in: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, N. Calzolari, K. Choukri, C. Cieri, T. Declerck, S. Goggi, K. Hasida, H. Isahara, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis and T. Tokunaga, eds, European Language Resources Association (ELRA), Miyazaki, Japan, 2018. https://aclanthology.org/L18-1544.

[11] J. Frey, L.-P. Meyer, N. Arndt, F. Brei and K. Bulert, Benchmarking the Abilities of Large Language Models for RDF Knowledge Graph Creation and Comprehension: How Well Do LLMs Speak Turtle?, in: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG 2023)*, M. Alam, D. Buscaldi, M. Cochez, F. Osborne and D.R. Recupero, eds, CEUR Workshop Proceedings, Vol. 3559, CEUR, ISSN: 1613-0073. https://ceur-ws.org/Vol-3559/#paper3.

[12] J. Frey, L.-P. Meyer, N. Arndt, F. Brei and K. Bulert, Benchmarking the Abilities of Large Language Models for RDF Knowledge Graph Creation and Comprehension: How Well Do LLMs Speak Turtle?, *ArXiv* **abs/2309.17122** (2023). https://api.semanticscholar.org/CorpusID:263310661.

[13] J. Frey, L. Meyer, N. Arndt, F. Brei and K. Bulert, Benchmarking the Abilities of Large Language Models for RDF Knowledge Graph Creation and Comprehension: How Well Do LLMs Speak Turtle?, in: *Proceedings of the Workshop on Deep Learning for Knowledge Graphs (DL4KG 2023) co-located with the 21th International Semantic Web Conference (ISWC 2023), Athens, November 6-10, 2023*, M. Alam and M. Cochez, eds, CEUR Workshop Proceedings, Vol. 3559, CEUR-WS.org, 2023. https://ceur-ws.org/Vol-3559/paper-3.pdf.

[14] A.P. Gallardo, S. Consoli, M. Ceresa, R. Hulsman and L. Bertolini, On Constructing Biomedical Text-to-Graph Systems with Large Language Models, in: *Joint proceedings of the 3rd International workshop on knowledge graph generation from text (TEXT2KG) and Data Quality meets Machine Learning and Knowledge Graphs (DQMLKG) co-located with the Extended Semantic Web Conference ( ESWC 2024), Hersonissos, Greece, May 26-30, 2024*, S. Tiwari, N. Mihindukulasooriya, F. Osborne, D. Kontokostas, J. D'Souza, M. Kejriwal, M.A. Pellegrino, A. Rula, J.E.L. Gayo, M. Cochez and M. Alam, eds, CEUR Workshop Proceedings, Vol. 3747, CEUR-WS.org, 2024, p. 12. https://ceur-ws.org/Vol-3747/text2kg_paper10.pdf.

[15] C. Gardent, A. Shimorina, S. Narayan and L. Perez-Beltrachini, Creating Training Corpora for NLG Micro-Planners, in: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, ACL, Vancouver, Canada, 2017, pp. 179–188. doi:10.18653/v1/p17-1017. http://aclweb.org/anthology/P17-1017.

[16] S. Geng, M. Josifoski, M. Peyrard and R. West, Grammar-Constrained Decoding for Structured NLP Tasks without Finetuning, in: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino and K. Bali, eds, Association for Computational Linguistics, Singapore, 2023. doi:10.18653/v1/2023.emnlp-main.674. https://aclanthology.org/2023.emnlp-main.674/.

[17] D. Grangier, A. Katharopoulos, P. Ablin and A. Hannun, Need a Small Specialized Language Model? Plan Early!, 2024. https://arxiv.org/abs/2402.01093.

[18] R. Han, T. Peng, C. Yang, B. Wang, L. Liu and X. Wan, Is Information Extraction Solved by ChatGPT? An Analysis of Performance, Evaluation Criteria, Robustness and Errors, arXiv, 2023, arXiv:2305.14450 [cs]. doi:10.48550/arXiv.2305.14450. http://arxiv.org/abs/2305.14450.

[19] D. Harbecke, Y. Chen, L. Hennig and C. Alt, Why only Micro-F1? Class Weighting of Measures for Relation Classification, in: *Proceedings of NLP Power! The First Workshop on Efficient Benchmarking in NLP*, T. Shavrina, V. Mikhailov, V. Malykh, E. Artemova, O. Serikov and V. Protasov, eds, Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 32–41. doi:10.18653/v1/2022.nlppower-1.4. https://aclanthology.org/2022.nlppower-1.4.

[20] M. Hofer, J. Frey and E. Rahm, Towards self-configuring Knowledge Graph Construction Pipelines using LLMs - A Case Study with RML, in: *Proceedings of the 5th International Workshop on Knowledge Graph Construction co-located with 21th Extended Semantic Web Conference (ESWC 2024), Hersonissos, Greece, May 27, 2024*, D. Chaves-Fraga, A. Dimou, A. Iglesias-Molina, U. Serles and D.V. Assche, eds, CEUR Workshop Proceedings, Vol. 3718, CEUR-WS.org, 2024. https://ceur-ws.org/Vol-3718/paper6.pdf.

[21] M. Hofer, D. Obraczka, A. Saeedi, H. Köpcke and E. Rahm, Construction of Knowledge Graphs: State and Challenges, arXiv, 2023, arXiv:2302.11509 [cs]. doi:10.48550/arXiv.2302.11509. http://arxiv.org/abs/2302.11509.

[22] A. Hotho, J.L. Martinez-Rodriguez, A. Hogan and I. Lopez-Arevalo, Information extraction meets the Semantic Web: A survey, *Semant. Web* **11**(2) (2020), 255–335–. doi:10.3233/SW-180333.

[23] P.-L. Huguet Cabot and R. Navigli, REBEL: Relation Extraction By End-to-end Language generation, in: *Findings of the Association for Computational Linguistics: EMNLP 2021*, M.-F. Moens, X. Huang, L. Specia and S.W.-t. Yih, eds, Association for Computational Linguistics, Punta Cana, Dominican Republic, 2021, pp. 2370–2381. doi:10.18653/v1/2021.findings-emnlp.204. https://aclanthology.org/2021.findings-emnlp.204.

[24] C.C. Hussam Ghanem, Fine-Tuning vs. Prompting: Evaluating the Knowledge Graph Construction with LLMs, 2024. https://ceur-ws.org/Vol-3747/text2kg_paper7.pdf.

[25] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y.J. Bang, A. Madotto and P. Fung, Survey of Hallucination in Natural Language Generation, *ACM Comput. Surv.* **55**(12) (2023). doi:10.1145/3571730.

[26] L. Jiang, X. Yan and R. Usbeck, A Structure and Content Prompt-based Method for Knowledge Graph Question Answering over Scholarly Data., in: *QALD/SemREC@ ISWC*, 2023.

[27] B. Jin, G. Liu, C. Han, M. Jiang, H. Ji and J. Han, Large Language Models on Graphs: A Comprehensive Survey, arXiv, 2023, arXiv:2312.02783 [cs]. http://arxiv.org/abs/2312.02783.

[28] M. Josifoski, N. De Cao, M. Peyrard, F. Petroni and R. West, GenIE: Generative Information Extraction, in: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, M. Carpuat, M.-C. de Marneffe and I.V. Meza Ruiz, eds, Association for Computational Linguistics, Seattle, United States, 2022, pp. 4626–4643. doi:10.18653/v1/2022.naacl-main.342. https://aclanthology.org/2022.naacl-main.342.

[29] M. Josifoski, M. Sakota, M. Peyrard and R. West, Exploiting Asymmetry for Synthetic Training Data Generation: SynthIE and the Case of Information Extraction, in: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino and K. Bali, eds, Association for Computational Linguistics, Singapore, 2023, pp. 1555–1574. doi:10.18653/v1/2023.emnlp-main.96. https://aclanthology.org/2023.emnlp-main.96.

[30] N. Kandpal, H. Deng, A. Roberts, E. Wallace and C. Raffel, Large language models struggle to learn long-tail knowledge, in: *Proceedings of the 40th International Conference on Machine Learning*, ICML'23, JMLR.org, 2023.

[31] P. Ke, H. Ji, Y. Ran, X. Cui, L. Wang, L. Song, X. Zhu and M. Huang, JointGT: Graph-Text Joint Representation Learning for Text Generation from Knowledge Graphs, in: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, C. Zong, F. Xia, W. Li and R. Navigli, eds, Association for Computational Linguistics, Online, 2021, pp. 2526–2538. doi:10.18653/v1/2021.findings-acl.223. https://aclanthology.org/2021.findings-acl.223.

[32] J. Lehmann, A. Meloni, E. Motta, F. Osborne, D.R. Recupero, A.A. Salatino and S. Vahdati, Large Language Models for Scientific Question Answering: An Extensive Analysis of the SciQA Benchmark, in: *The Semantic Web*, A. Meroño Peñuela, A. Dimou, R. Troncy, O. Hartig, M. Acosta, M. Alam, H. Paulheim and P. Lisena, eds, Springer Nature Switzerland, Cham, 2024, pp. 199–217. ISBN 978-3-031-60626-7.

[33] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov and L. Zettlemoyer, BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension, arXiv, 2019, arXiv:1910.13461 [cs, stat]. doi:10.48550/arXiv.1910.13461. http://arxiv.org/abs/1910.13461.

[34] D. Li, Z. Chen, E. Cho, J. Hao, X. Liu, F. Xing, C. Guo and Y. Liu, Overcoming Catastrophic Forgetting During Domain Adaptation of Seq2seq Language Generation, in: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Seattle, United States, 2022, pp. 5441–5454. doi:10.18653/v1/2022.naacl-main.398. https://aclanthology.org/2022.naacl-main.398.

[35] G. Li, P. Wang, W. Ke, Y. Guo, K. Ji, Z. Shang, J. Liu and Z. Xu, Recall, Retrieve and Reason: Towards Better In-Context Relation Extraction, in: *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, K. Larson, ed., International Joint Conferences on Artificial Intelligence Organization, 2024, pp. 6368–6376, Main Track. doi:10.24963/ijcai.2024/704.

[36] X. Li, F. Polat and P. Groth, Do Instruction-tuned Large Language Models Help with Relation Extraction?, in: *Joint proceedings of the 1st workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM) and the 2nd challenge on Language Models for Knowledge Base Construction (LM-KBC)*, S. Razniewski, J.-C. Kalo, S. Singhania and J.Z. Pan, eds, CEUR Workshop Proceedings, Vol. 3577, CEUR, ISSN: 1613-0073. https://ceur-ws.org/Vol-3577/#paper4.

[37] M.X. Liu, F. Liu, A.J. Fiannaca, T. Koo, L. Dixon, M. Terry and C.J. Cai, "We Need Structured Output": Towards User-centered Constraints on Large Language Model Output, in: *Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, CHI EA '24, Association for Computing Machinery, New York, NY, USA, 2024. ISBN 9798400703317. doi:10.1145/3613905.3650756.

[38] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi and G. Neubig, Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing, *ACM Comput. Surv.* **55**(9) (2023). doi:10.1145/3560815.

[39] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi and G. Neubig, Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing, *ACM Comput. Surv.* **55**(9) (2023), 195:1–195:35. doi:10.1145/3560815.

[40] Y. Liu, D. Li, K. Wang, Z. Xiong, F. Shi, J. Wang, B. Li and B. Hang, Are LLMs good at structured outputs? A benchmark for evaluating structured output capabilities in LLMs, *Information Processing & Management* **61**(5) (2024), 103809. doi:https://doi.org/10.1016/j.ipm.2024.103809. https://www.sciencedirect.com/science/article/pii/S0306457324001687.

[41] Y. Lu, Q. Liu, D. Dai, X. Xiao, H. Lin, X. Han, L. Sun and H. Wu, Unified Structure Generation for Universal Information Extraction, in: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov and A. Villavicencio, eds, Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 5755–5772. doi:10.18653/v1/2022.acl-long.395. https://aclanthology.org/2022.acl-long.395.

[42] Z. Lu, X. Li, D. Cai, R. Yi, F. Liu, X. Zhang, N.D. Lane and M. Xu, Small Language Models: Survey, Measurements, and Insights, 2024. https://arxiv.org/abs/2409.15790.

[43] L. Meyer, J. Frey, F. Brei and N. Arndt, Assessing SPARQL capabilities of Large Language Models, *CoRR* **abs/2409.05925** (2024). doi:10.48550/ARXIV.2409.05925. https://doi.org/10.48550/arXiv.2409.05925.

[44] L.-P. Meyer, C. Stadler, J. Frey, N. Radtke, K. Junghanns, R. Meissner, G. Dziwis, K. Bulert and M. Martin, LLM-assisted Knowledge Graph Engineering: Experiments with ChatGPT, in: *First Working Conference on Artificial Intelligence Development for a Resilient and Sustainable Tomorrow*, C. Zinke-Wehlmann and J. Friedrich, eds, Springer Fachmedien Wiesbaden, Wiesbaden, 2024, pp. 103–115, Series Title: Informatik aktuell. ISBN 978-3-658-43704-6 978-3-658-43705-3. doi:10.1007/978-3-658-43705-3_8. https://link.springer.com/10.1007/978-3-658-43705-3%5F8.

[45] N. Mihindukulasooriya, M. Sava, G. Rossiello, M.F.M. Chowdhury, I. Yachbes, A. Gidh, J. Duckwitz, K. Nisar, M. Santos and A. Gliozzo, Knowledge Graph Induction Enabling Recommending and Trend Analysis: A Corporate Research Community Use Case, in: *The Semantic Web – ISWC 2022: 21st International Semantic Web Conference, Virtual Event, October 23–27, 2022, Proceedings*, Springer-Verlag, Berlin, Heidelberg, 2022, pp. 827–844–. ISBN 978-3-031-19432-0. doi:10.1007/978-3-031-19433-7_47. https://doi.org/10.1007/978-3-031-19433-7%5F47.

[46] N. Mihindukulasooriya, S. Tiwari, C.F. Enguix and K. Lata, Text2KGBench: A Benchmark for Ontology-Driven Knowledge Graph Generation from Text, 2023. https://arxiv.org/abs/2308.02357.

[47] S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain and J. Gao, Large Language Models: A Survey, 2024.

[48] C. Möller and R. Usbeck, *Analyzing the Influence of Knowledge Graph Information on Relation Extraction*, in: *The Semantic Web*, Springer Nature Switzerland, 2025, pp. 460–480–. ISSN 1611-3349. ISBN 9783031945755. doi:10.1007/978-3-031-94575-5$_2$5.

[49] T. Nayak, N. Majumder, P. Goyal and S. Poria, Deep Neural Approaches to Relation Triplets Extraction: a Comprehensive Survey, *Cognitive Computation* **13** (2021), 1215–1232. https://api.semanticscholar.org/CorpusID:232427782.

[50] J.Z. Pan, S. Razniewski, J.-C. Kalo, S. Singhania, J. Chen, S. Dietze, H. Jabeen, J. Omeliyanenko, W. Zhang, M. Lissandrini, R. Biswas, G. de Melo, A. Bonifati, E. Vakaj, M. Dragoni and D. Graux, Large Language Models and Knowledge Graphs: Opportunities and Challenges, arXiv, 2023, arXiv:2308.06374 [cs]. http://arxiv.org/abs/2308.06374.

[51] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang and X. Wu, Unifying Large Language Models and Knowledge Graphs: A Roadmap, *ArXiv* **abs/2306.08302** (2023). https://api.semanticscholar.org/CorpusID:259165563.

[52] G. Paolini, B. Athiwaratkun, J. Krone, J. Ma, A. Achille, R. Anubhai, C.N. dos Santos, B. Xiang and S. Soatto, Structured Prediction as Translation between Augmented Natural Languages, in: *9th International Conference on Learning Representations, ICLR 2021*, 2021.

[53] G. Paolini, B. Athiwaratkun, J. Krone, J. Ma, A. Achille, R. Anubhai, C.N.d. Santos, B. Xiang and S. Soatto, Structured Prediction as Translation between Augmented Natural Languages, arXiv, 2021, arXiv:2101.05779 [cs]. doi:10.48550/arXiv.2101.05779. http://arxiv.org/abs/2101.05779.

[54] K. Papineni, S. Roukos, T. Ward and W.-J. Zhu, Bleu: a Method for Automatic Evaluation of Machine Translation, in: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, P. Isabelle, E. Charniak and D. Lin, eds, Association for Computational Linguistics, Philadelphia, Pennsylvania, USA, 2002, pp. 311–318. doi:10.3115/1073083.1073135. https://aclanthology.org/P02-1040.

[55] A. Patel, C. Raffel and C. Callison-Burch, DataDreamer: A Tool for Synthetic Data Generation and Reproducible LLM Workflows, in: *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, L.-W. Ku, A. Martins and V. Srikumar, eds, Association for Computational Linguistics, Bangkok, Thailand, 2024, pp. 3781–3799. doi:10.18653/v1/2024.acl-long.208. https://aclanthology.org/2024.acl-long.208/.

[56] A. Patel, B. Li, M.S. Rasooli, N. Constant, C. Raffel and C. Callison-Burch, Bidirectional Language Models Are Also Few-shot Learners, *ArXiv* **abs/2209.14500** (2022). https://api.semanticscholar.org/CorpusID:252595927.

[57] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P.J. Liu, Exploring the limits of transfer learning with a unified text-to-text transformer, *J. Mach. Learn. Res.* **21**(1) (2020).

[58] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P.J. Liu, Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer, arXiv, 2023, arXiv:1910.10683 [cs, stat]. http://arxiv.org/abs/1910.10683.

[59] A.G. Regino, A. Rossanez, R. da Silva Torres and J.C. dos Reis, A systematic literature review on rdf triple generation from natural language text, *Semantic Web Journal* (2025).

[60] S. Reyd and A. Zouaq, Assessing the Generalization Capabilities of Neural Machine Translation Models for SPARQL Query Generation, in: *The Semantic Web - ISWC 2023 - 22nd International Semantic Web Conference, Athens, Greece, November 6-10, 2023, Proceedings, Part I*, T.R. Payne, V. Presutti, G. Qi, M. Poveda-Villalón, G. Stoilos, L. Hollink, Z. Kaoudi, G. Cheng and J. Li, eds, Lecture Notes in Computer Science, Vol. 14265, Springer, 2023, pp. 484–501. doi:10.1007/978-3-031-47240-4_26. https://doi.org/10.1007/978-3-031-47240-4%5F26.

[61] C. Ringwald, F. Gandon, C. Faron, F. Michel and H. Abi Akl, Well-Written Knowledge Graphs: Most Effective RDF Syntaxes for Triple Linearization in End-to-End Extraction of Relations from Texts (Student Abstract), in: *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38, 2024, pp. 23631–23632.

[62] C. Ringwald, F. Gandon, C. Faron, F. Michel and H.A. Akl, Overcoming the Generalization Limits of SLM Finetuning for Shape-Based Extraction of Datatype and Object Properties, in: *KCAP 2025 - The Thirteenth International Conference on Knowledge Capture- 10- Association for Computing Machinery*, ACM, Dayton, OH, USA,, United States, 2025, Accepted at KCAP2025 - Preprint version not updated with reviews. https://hal.science/hal-05285428.

[63] C. Ringwald, F. Gandon, C. Faron, F. Michel and H.A. Akl, 12 Shades of RDF: Impact of Syntaxes on Data Extraction with Language Models, in: *The Semantic Web: ESWC 2024 Satellite Events*, A. Meroño Peñuela, O. Corcho, P. Groth, E. Simperl, V. Tamma, A.G. Nuzzolese, M. Poveda-Villalón, M. Sabou, V. Presutti, I. Celino, A. Revenko, J. Raad, B. Sartini and P. Lisena, eds, Springer Nature Switzerland, Cham, 2025, pp. 81–91. ISBN 978-3-031-78952-6.

[64] K. Shenoy, F. Ilievski, D. Garijo, D. Schwabe and P. Szekely, A study of the quality of Wikidata, *Journal of Web Semantics* **72** (2022), 100679. doi:10.1016/j.websem.2021.100679. https://www.sciencedirect.com/science/article/pii/S1570826821000536.

[65] A. Smirnova and P. Cudré-Mauroux, Relation Extraction Using Distant Supervision: A Survey, *ACM Comput. Surv.* **51**(5) (2018). doi:10.1145/3241741.

[66] A. Smirnova and P. Cudré-Mauroux, Relation Extraction Using Distant Supervision: A Survey, *ACM Comput. Surv.* **51**(5) (2018). doi:10.1145/3241741.

[67] W. Su, Q. Ai, X. Li, J. Chen, Y. Liu, X. Wu and S. Hou, Wikiformer: Pre-training with Structured Information of Wikipedia for Ad-hoc Retrieval, arXiv, 2024, arXiv:2312.10661 [cs]. http://arxiv.org/abs/2312.10661.

[68] K. Sun, P. Qi, Y. Zhang, L. Liu, W.Y. Wang and Z. Huang, Tokenization Consistency Matters for Generative Models on Extractive NLP Tasks, arXiv, 2023, arXiv:2212.09912 [cs]. doi:10.48550/arXiv.2212.09912. http://arxiv.org/abs/2212.09912.

[69] L. Sutawika, A. Komatsuzaki and C. Raffel, Pile-T5, 2024, Blog post. https://blog.eleuther.ai/pile-t5/.

[70] B. Taillé, V. Guigue, G. Scoutheeten and P. Gallinari, Let's Stop Incorrect Comparisons in End-to-end Relation Extraction!, in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He and Y. Liu, eds, Association for Computational Linguistics, Online, 2020, pp. 3689–3701. doi:10.18653/v1/2020.emnlp-main.301. https://aclanthology.org/2020.emnlp-main.301.

[71] S. Wadhwa, S. Amir and B. Wallace, Revisiting Relation Extraction in the era of Large Language Models, in: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, A. Rogers, J. Boyd-Graber and N. Okazaki, eds, Association for Computational Linguistics, Toronto, Canada, 2023, pp. 15566–15589. doi:10.18653/v1/2023.acl-long.868.

[72] C. Wang, X. Liu, Z. Chen, H. Hong, J. Tang and D. Song, DeepStruct: Pretraining of Language Models for Structure Prediction, in: *Findings of the Association for Computational Linguistics: ACL 2022*, S. Muresan, P. Nakov and A. Villavicencio, eds, Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 803–823. doi:10.18653/v1/2022.findings-acl.67. https://aclanthology.org/2022.findings-acl.67.

[73] F. Wang, Z. Zhang, X. Zhang, Z. Wu, T. Mo, Q. Lu, W. Wang, R. Li, J. Xu, X. Tang, Q. He, Y. Ma, M. Huang and S. Wang, A Comprehensive Survey of Small Language Models in the Era of Large Language Models: Techniques, Enhancements, Applications, Collaboration with LLMs, and Trustworthiness, 2024. https://arxiv.org/abs/2411.03350.

[74] R. Wang, Z. Zhang, L. Rossetto, F. Ruosch and A. Bernstein, NLQxform: A Language Model-based Question to SPARQL Transformer, arXiv, 2023, arXiv:2311.07588 [cs]. doi:10.48550/arXiv.2311.07588. http://arxiv.org/abs/2311.07588.

[75] Y. Wang, W. Wang, S. Joty and S.C.H. Hoi, CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation, 2021. https://arxiv.org/abs/2109.00859.

[76] C. Wu and L. Chen, Utber: Utilizing Fine-Grained Entity Types to Relation Extraction with Distant Supervision, in: *2020 IEEE International Conference on Smart Data Services (SMDS)*, 2020, pp. 63–71. doi:10.1109/SMDS49396.2020.00015.

[77] D. Xu, W. Chen, W. Peng, C. Zhang, T. Xu, X. Zhao, X. Wu, Y. Zheng and E. Chen, Large Language Models for Generative Information Extraction: A Survey, arXiv, 2023, arXiv:2312.17617 [cs] version: 1. doi:10.48550/arXiv.2312.17617. http://arxiv.org/abs/2312.17617.

[78] D. Yang, X. Wang and R. Celebi, Expanding the Vocabulary of BERT for Knowledge Base Construction, arXiv, 2023, arXiv:2310.08291 [cs]. http://arxiv.org/abs/2310.08291.

[79] Y. Yao, D. Ye, P. Li, X. Han, Y. Lin, Z. Liu, Z. Liu, L. Huang, J. Zhou and M. Sun, DocRED: A Large-Scale Document-Level Relation Extraction Dataset, in: *Proceedings of the 57th Annual Meeting of the ACL*, ACL, Florence, Italy, 2019, pp. 764–777. doi:10.18653/v1/p19-1074. http://arxiv.org/abs/1906.06127.

[80] H. Ye, N. Zhang, H. Chen and H. Chen, Generative Knowledge Graph Construction: A Review, *CoRR* **abs/2210.12714** (2022). doi:10.48550/arXiv.2210.12714.

[81] A. Zaveri, D. Kontokostas, M.A. Sherif, L. Bühmann, M. Morsey, S. Auer and J. Lehmann, User-driven quality evaluation of DBpedia, in: *Proceedings of the 9th International Conference on Semantic Systems*, I-SEMANTICS '13, Association for Computing Machinery, New York, NY, USA, 2013, pp. 97–104–. ISBN 9781450319720. doi:10.1145/2506182.2506195.

[82] A. Zaveri, D. Kontokostas, S. Hellmann, J. Umbrich, M. Färber, F. Bartscherer, C. Menne, A. Rettinger, A. Zaveri, D. Kontokostas, S. Hellmann and J. Umbrich, Linked data quality of DBpedia, Freebase, OpenCyc, Wikidata, and YAGO, *Semant. Web* **9**(1) (2018), 77–129–. doi:10.3233/SW-170275.

[83] B. Zhang, I. Reklos, N. Jain, A.M. Peñuela and E. Simperl, Using Large Language Models for Knowledge Engineering (LLMKE): A Case Study on Wikidata, 2023. https://arxiv.org/abs/2309.08491.
[84] S.D. Zhang, C. Tigges, S. Biderman, M. Raginsky and T. Ringer, Can Transformers Learn to Solve Problems Recursively?, arXiv, 2023, arXiv:2305.14699 [cs]. http://arxiv.org/abs/2305.14699.
[85] M. Šakota and R. West, Combining Constrained and Unconstrained Decoding via Boosting: BoostCD and Its Application to Information Extraction, 2025. https://arxiv.org/abs/2506.14901.