12 shades of RDF: Impact of Syntaxes on Data Extraction with Language Models

Célian Ringwald, Fabien Gandon, Catherine Faron, Franck Michel, Hanna Abi Akl

12 shades of RDF: Impact of Syntaxes on Data Extraction with Language Models

Célian Ringwald[1 [0000−0002−7302−9037]], Fabien Gandon[1,2 [0000−0003−0543−1232]], Catherine Faron[1 [0000−0001−5959−5561]], Franck Michel[1 [0000−0001−9064−0463]], Hanna Abi Akl[1,2 [0000−0001−9829−7401]]

[1] Université Côte d'Azur, Inria, CNRS, I3S
[2] Data ScienceTech Institute

Abstract. The fine-tuning of generative pre-trained language models (PLMs) on a new task can be impacted by the choice made for representing the inputs and outputs. This article focuses on the linearization process used to structure and represent, as output, facts extracted from text. On a restricted relation extraction (RE) task, we challenged T5 and BART by fine-tuning them on 12 linearizations, including RDF standard syntaxes and variations thereof. Our benchmark covers: the validity of the produced triples, the performance of the model, the training behaviours 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: Targeted Data Properties Extraction

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) e.g. Wikipedia articles with corresponding Wikidata or DBpedia subgraphs, and (2) the usage of pre-trained language models (PLM) to carry out this task. However, Wikidata and DBpedia still struggle with coverage and quality issues [24,6]. In this context, extracting from Wikipedia the missing information in KGs is an important task. A promising research direction is to design a system allowing adaptability and fine-grained quality control. Now that we have end-to-end off-the-shelf methods, we have the opportunity to directly produce RDF serialization from natural language, and specify and control the output with constraints (e.g. with SHACL, ShEx). However, to the best of our knowledge, no LLM-based system currently performs RE directly from Wikipedia articles with a specific RDF syntax. Formally, let $Db \subseteq W \times G$ be a dual base, where W is a set of Wikipedia articles and G a 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.

Generative PLMs are very flexible but variations in prompts and output formats can impact their performances. In this paper, we focus on RE for the most common datatype properties of DBpedia resources of type `dbo:Person`. In this simplified setup, we challenged two encoder-decoder models trained on twelve RDF syntax flavours. Hence our research question: *How does the choice of a syntax impact the generation of RDF triples using datatype properties?*

After reviewing the related works (Section 2) we present a method to extract RDF from Wikipedia (Section 3) and the experiments we conducted (Section 4) before discussing the results (Section 5).

2 Related Works: RDF Extraction with Language Models

Before investing in generative PLMs, the research community focused on systems built on top of encoder-only PLMs (derived from BERT [2]), where relations were decoded by design in a discriminative manner [19]. 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 [13] and T5 models [22]. For RE we can cite: REBEL [7], TALN [20], DEEPstruct [28] or UIE [16]. (2) Decoder-only models have interesting generalization properties but generally work at large scale and need dedicated resources to be adapted to a specific task. Few-shot and zero-shot approaches were studied for these reasons. But few-shot learning does not seem sufficient to solve the relation extraction task [4]. Parameter-efficient fine-tuning (PEFT) approaches [3] allow the adaptation of large models to a specific task but do not necessarily perform as well as fine-tuned models [14].

The use of generative pre-trained models allows us to learn the triple syntax implicitly from the examples submitted during training [29]. The question of the structure of the output was initially referred to as "Answer Engineering" [15], 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 the usage of a generative model pre-trained on natural language texts [9]. Until now, different methods have been investigated but they were not rigorously compared. The two main solutions proposed represent a relation as a list of triples [28]: $((s1, p1, o1), (s2, p1, o2), ...)$ or a sequence of tags [12] where each element of the triple is preceded by a special token e.g. H, R, T in $\langle H\rangle s1\langle R\rangle p1\langle T\rangle o1\langle H\rangle s2\langle R\rangle p1\langle T\rangle o2$. [7] and [11] proposed a triple linearization method (subject-collapsed) where triples sharing the same subject are grouped to avoid repetition.

In this article, we will also consider the syntaxes recommended by the W3C to serialize RDF triples, namely, RDF/XML, N-Triples, Turtle, and JSON-LD.

3 Methodological Framework: Definitions and Notations

Our pipeline takes as input a DBpedia dump[3] which is filtered to check that the values of the triples we target are mentioned in the corresponding Wikipedia abstracts and comply with a SHACL shape (Section 3.1). The selected triples are ordered and the URIs they use are cleaned. The dataset is then linearized into 12 syntaxes (Section 3.2) and each version is used in a K-fold approach. (Sections 3.3 and 4).

3.1 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 only focused on datatype properties that relate to numbers, string values and dates. This is a good starting point because LLM hallucination generally affects these literal values [8]. Moreover, until now, the proposed generative models mostly focused on object properties, allowing for constrained decoding [10] that cannot be envisaged in the case of datatypes properties.

We focused on the DBpedia subgraphs describing instances of one of the most represented DBpedia classes, `dbo:Person`, and their corresponding Wikipedia abstracts. The instances of this class include the highest number of datatype properties, among which: `rdfs:label`, `dbo:alias`, `dbo:birthName`, `dbo:birthDate`, `dbo:deathDate`, `dbo:birthYear`, `dbo:deathYear`. Our original set was composed of 1 833 493 entities and 3 249 446 related triples, but this is over-scaled compared to our task. Preliminary trials [23] shown that a smaller set could be sufficient to learn the graph pattern captured by a SHACL shape.

Several works mention the noise caused by the massive alignment of facts with text [25], which also impacts T-Rex or REBEL [14]. More specifically, two problems are pointed out: the triple values do not necessarily appear in the text and, conversely, the facts of the text may not have counterpart triples in the knowledge base. To solve the first one, we keep only the triples describing values that could be found in the Wikipedia abstract of a given entity. To answer the second problem, we designed a SHACL shape targeting `dbo:Person` and specifying which property is mandatory and which is optional, and we kept only the graphs valid against this shape. By applying these two pre-processing steps to a random sample of 1000 entities, we found that 80% of the triples contain values that can be found in the Wikipedia abstract, but that only 45% of the entities have a description graph valid against the shape.

Our pipeline includes two additional pre-processing steps: (1) Triple ordering: [17] demonstrated the importance of having in the first place the triples typing the entity. As RDFlib [4] does not ensure this on every syntax, we added an ordering step. (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

[3] `https://databus.dbpedia.org/dbpedia/collections/dbpedia-snapshot-2022-09`
[4] https://rdflib.readthedocs.io/en/stable/

3.2 RDF Syntaxes and Alternative Linearizations

Our benchmark covers three types of syntaxes. First we consider the four W3C RDF syntaxes: XML-RDF (noted $_x$), Turtle (noted $_T$), N-Triples (noted $_n$) and JSON-LD (noted $_j$). Second, we include the classical syntaxes of the literature: the List (noted $_l$) and the Tags (noted $_g$). Finally, we propose *Turtle light*, a simplified Turtle syntax where namespaces, prefixes, and datatypes are considered as already defined (noted $_t$). We also consider two variations. The first one is the triple subject factorisation (noted $_f$). It is naturally integrated into Turtle, JSON-LD and RDF-XML and we also apply to the Turtle Light, the List and the Tags. A second variation is the single-line writing (noted $_1$) to evaluate the impact of the carriage return [5]. Finally, we consider the use of vocabulary extension (noted $_v$) which first ensures that syntax-related tokens will not be considered as unknown by the tokenizer, but also allows us to detach these tokens from the pre-trained embedding space because they relate to another semantic space, e.g. a comma in Turtle vs. a comma in a text in natural language. For each W3C syntax, we added all the tokens specified in its recommendation.

3.3 Pre-trained Language Models: the Choice of Frugal Sizes

We focused our benchmark on the two encoder-decoder models traditionally used in the literature (see Section 2), BART (noted B) and $T5$, and we limited our experiment to the "base" size of these pre-trained models that can be seen as small or frugal LLMs compared to decoder-only models: today's LLMs count billions of parameters [18], where BART base uses 140M parameters and T5 base 220M. When comparing BART and T5, they were pre-trained on different datasets and in a different manner. Each model is given a specific Task Prompt, where $Abstract is a Wikipedia abstract and $Syntax the targeted RDF syntax: (1) BART: "`$entity_URI : $Abstract`"; and (2) T5: "`Translate English to $Syntax: [$entity_URI] $Abstract`". In the next sections, we use the notations introduced in this section to name each possible configuration. For instance, a BART model trained on *Turtle Light* syntax, with factorization and multi-lines will be written B_{tf}, and a $T5$ model trained on lists with a vocabulary will be written $T5_{vl}$.

4 Experimental set-up

4.1 Fine-Tuning Details

Our code[6] is published under an open license and based on a fork of REBEL[7], which we extended and adapted to our task. For each standard RDF syntax, we developed a specific parser and integrated the metrics we present below.

[5] the "\n" special token

[6] `https://github.com/datalogism/12ShadesOfRDFSyntax`

[7] `https://github.com/Babelscape/rebel`

Data Split: we follow a 5-fold cross-validation based on 5 000 rotated examples split into 4 000 training examples and 1 000 test examples. In addition, 250 disjoint examples are used for the evaluation. **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 steps of warmup and configured with an early stop mode with patience of 5 steps. Both models were trained on a single GPU, Tesla V100-SXM2-32GB for BART and NVIDIA A100 80GB PCIe for T5 (able to manage bf16). **Management of Tokenization Inconsistencies:** As underlined in [26,1], both T5 and BART tokenizers may duplicate or delete spaces before or after special tokens. For this reason, we controlled the 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.

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 a model to copy from the input the targeted entity. In addition, we define the rate of SHACL-Validated Triples R_{SVT}.

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

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

Traditionally, RE focuses on *precision* (P), *recall* (R), F_1 score, or *top@k* metrics. Following [5], only parsed outputs are evaluated with these metrics and we focus on macro-measures (P^+, R^+, F_1^+) that better account for the imbalanced distribution of properties.

These metrics follow the *Strict Mode* evaluation [27], 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 [21] (B): the closer B is to 1, the greater the similarity between string values.

To assess the training process itself based on cross-entropy loss objective, we define meta-metrics to monitor the behaviour of the R_{PT} and F_1 metrics. The three meta-metrics are defined as: (1) the learning velocity V is the number of epochs needed to reach the first saturation (> 0.9) of a given metric, e.g. $V_{F_1^-}$ is the number of epochs needed to reach the first saturation when $F_1^- > 0.9$; (2) the stability of a learning process is defined as the ratio of epochs during which a

metric remains stable after the first saturation, e.g. for F_1^- we note the stability $S_{F_1^-}$; (3) the final divergence of the learning process is defined as the number of folds for which there is a final divergence, e.g. the divergence $D_{F_1^-}$ is the number of folds for which the final F_1^- is lower than the value of its first saturation. In some folds, the learning behaviours metrics may have no value. First, when saturation never happens on a fold, the average velocity ($\overline{V}$) and stability ($\overline{S}$) cannot be computed. For this reason, we focus on the micro-F1 (F_1^-), because the macro-F1 (F_1^+) metric never saturates[8]

Finally, 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: $G_g = \overline{R_{PT}} \times \overline{R_{CS}} \times \overline{R_{SVT}} \times \overline{F_1^+} \times 100$ where, for instance, $\overline{F_1^+}$ is the average of F_1^+ over the splits.

Additionally, we monitored the training time T_t (in minutes) and the carbon cost[9] C_c (emissions of CO_2-equivalents in kg) for training a model.

5 Results and Discussions: the Best Syntaxes

Table 5 compiles the results for the best-performing configurations ; additional details are online[10]. As the configurations using a vocabulary systematically perform better, we only report these in the table. Starting with the **triple validity** metric, almost every configuration produces triples that could be parsed ($\overline{R_{PT}}$); except $T5$ that struggles to produce the Turtle and N-Triples syntaxes.

Considering the $\overline{lev}$ computed on the triples with syntax errors, we observe the ability of some models to extract close to perfect triples. Moreover, a lot of 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 (see examples online[10]). In contrast, high values of the $\overline{lev}$ also allow us to identify models producing triples that can be far from the well-formed triples ($T5_{vT}$, $T5_{vn}$, $T5_{vlf}$, $T5_{vt1}$). Once the results are parsable, they are always valid against the shape ($\overline{R_{SVT}}$). The subject URI is also generally easily copied from the prompt by the model, even if we can find some exceptions ($T5_{vgf}$ and $T5_{vlf}$).

The **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, where every configuration seems to reach an almost perfect extraction. From that point of view, $T5_{vj}$ is our best result, closely followed by B_{vgf}, B_{vtf1}, B_{vT}, $T5_{vtf1}$ and $T5_{vgf}$. Considering the BLUE score B, we can see that $T5_{vj}$ is always perfectly predicting string values of datatype properties, and other models generally perform well, except $T5_{vtl}$.

[8] A formalisation of the computation of those three metrics is detailed on GitHub: `https://github.com/datalogism/12ShadesOfRDFSyntax/tree/main/eval`

[9] `https://codecarbon.io/`

[10] `https://wandb.ai/celian-ringwald/12ShadesOfRDF`

The **training behaviour** metrics show that the models generally saturate at the first epoch. Velocity metrics ($\overline{V_{R_{PT}}}$ and $\overline{V_{F_1^-}}$) also demonstrate that models learn the relation extraction task slightly before they learn to produce syntactically correct triples. Considering the stability ($\overline{S_{R_{PT}}}$ and $\overline{S_{F_1^-}}$), we observed that two models $T5_{vT}$ and $T5_{vn}$ experience difficulties to converge. As for the divergence metrics ($\sum D_{R_{PT}}$ and $\sum D_{F_1^-}$), we see that the forgetting effect could be reached, but BART-based models are less impacted.

The **resources metrics** also show important discrepancies between models, that could be explained by the verbosity of some syntaxes, the resources needed for each model, and the ability of the latter to learn a given syntax without divergences. Indeed T5 models are greedier than BART models and simple syntaxes are thriftier than RDF ones. Model training costs vary from 29g of CO_2, reached by B_{vtf} to 300g of CO_2 emitted by T_{vx}.

Globally, BART generally writes syntactically better triples than T5, where T5 needs less training epochs but requires more resources. The factorisation variation has shown a positive impact on the performance of the models, except on $T5_{vl}$ configurations. On the Turtle Light variations, the one-line option also improves quality but the best configuration seems to be the combination of both factorisation and one-line writing. In the end, B_{vtf1} offers good performances, at a low cost with a standard and human-readable syntax.

Finally, the experiment conducted has some limitations. T5 and BART were pre-trained partially on Wikipedia, which means they may already have been exposed to some of the knowledge we want to extract. The second limitation is our dependency on the tokenization method which, if changed, could impact the effectiveness of a given syntax to capture relations.

6 Conclusion: a Light Turtle Goes a Long Way

In this article, we evaluated how the choice of a syntax impacts the generation of RDF triples focusing on datatype properties extraction from text. We showed that basic syntaxes (list and tags) are generally easily parsed but lead to average performances. While learning W3C RDF syntaxes is more resource-consuming, the best-performing configuration $T5_{vj}$ outperforms the others at the cost of 2 hours of training on an A100 GPU and 250g of CO_2 produced. An interesting compromise is the use of simplified syntaxes, close to standards, robust and quick to learn, in particular **inline factorised Turtle Light** (B_{vtf1} and $T5_{vtf1}$).

Our experiments also showed the limits of full fine-tuning in some training configurations: $T5_{vn}$ or $T5_{vT}$, underlining that Turtle and N-Triples may require better-fitted adaptation. Several directions could be explored, including the use of a loss or an iterative learning process designed to take into account the syntax and the task, as well as models specialized on code.

Acknowledgments This work is supported by 3IA Côte d'Azur (ANR-19-P3IA-0002) and UCAJEDI (ANR-15-IDEX-01) and the OPAL infrastructure and Université Côte d'Azur's Center for High-Performance Computing.

Config	Triple Validity			RE performances ×100				Edition m.		Training behaviors							Resources		G_g
	$\overline{R_{PT}}$	$\overline{R_{CS}}$	$\overline{R_{SVT}}$	$\overline{F_1^-}$	$\overline{F_1^+}$	$\overline{P^+}$	$\overline{R^+}$	$\overline{B}$	$\overline{lev}$	$\overline{Nb_{epochs}}$	$\overline{V_{R_{PT}}}$	$\overline{S_{R_{PT}}}$	$\sum D_{R_{PT}}$	$\overline{V_{F_1^-}}$	$\overline{S_{F_1^-}}$	$\sum D_{F_1^-}$	$\overline{C_c}$	$\overline{T_t}$	
$T5_{vj}$	**1**	**1**	**1**	**99.75**	**95.63**	**100.00**	**94.37**	**1.00**	**0**	13	*0.2*	∅	**0**	*0.2*	∅	2	0.252	137	**96**
B_{vgf}	**1**	**1**	**1**	99.69	*95.47*	*99.29*	94.28	0.97	**0**	15	**0**	∅	**0**	**0**	∅	*1*	0.042	29	**95**
B_{vtf1}	**1**	**1**	**1**	99.72	94.54	97.09	93.20	0.93	**0**	12	**0**	∅	**0**	*0.2*	∅	2	*0.035*	*27*	**95**
B_{vT}	**1**	**1**	**1**	*99.73*	94.43	96.39	*93.42*	0.97	11	22	0	∅	**0**	**0**	∅	*1*	0.104	75	*94*
$T5_{vtf1}$	**1**	**1**	**1**	99.51	93.94	95.48	93.13	0.96	**0**	14	*0.2*	∅	**0**	**0**	∅	3	0.099	56	*94*
$T5_{vx}$	**1**	**1**	**1**	99.58	92.86	96.81	91.91	0.95	2	18	0.4	∅	*1*	0.4	∅	2	0.324	206	93
B_{vg}	**1**	**1**	**1**	99.62	92.57	96.34	91.08	0.94	**0**	17	**0**	∅	**0**	**0**	∅	**0**	0.053	46	93
$T5_{vl}$	**1**	**1**	**1**	99.55	92.34	95.19	91.40	0.97	**0**	*11*	0.8	∅	*1*	0.6	∅	*1*	0.118	75	92
B_{vlf}	**1**	**1**	**1**	99.63	91.99	96.68	90.49	**1.00**	2	12	**0**	∅	**0**	**0**	∅	*1*	0.04	29	92
B_{vl}	**1**	**1**	**1**	99.62	92.03	94.75	90.37	0.90	18	18	**0**	∅	**0**	**0**	∅	1	0.064	54	92
B_{vtf}	**1**	**1**	**1**	99.49	90.72	95.45	89.13	0.97	**0**	12	**0**	∅	**0**	**0**	∅	**0**	**0.029**	**26**	91
$T5_{vgf}$	**1**	0.96	**1**	99.57	94.18	96.76	92.51	0.98	*1*	13	*0.2*	∅	**0**	*0.2*	∅	*1*	0.087	45	91
$T5_{vtf}$	**1**	**1**	**1**	99.33	90.72	95.81	88.72	0.96	*1*	**10**	0.8	∅	*1*	0.4	∅	*1*	0.072	44	90
B_{vj}	**1**	**1**	**1**	99.52	90.27	95.14	88.85	0.96	47	*11*	*0.2*	∅	**0**	**0**	∅	**0**	0.093	74	90
B_{vx}	**1**	**1**	**1**	99.46	89.87	96.69	88.85	0.97	17	14	**0**	∅	**0**	*0.2*	∅	*1*	0.092	75	90
$T5_{vt1}$	0.97	**1**	**1**	99.34	91.73	95.32	89.68	0.97	99	**10**	**0**	∅	**0**	*0*	∅	2	0.109	56	89
$T5_{vlf}$	**1**	*0.98*	**1**	99.32	90.32	94.28	89.43	0.88	205	*11*	1	∅	*1*	**0**	∅	4	0.081	49	88
B_{vn}	**1**	**1**	**1**	99.36	87.73	97.01	85.48	*0.99*	81	24	**0**	∅	**0**	0.4	∅	*1*	0.134	119	88
$T5_{vg}$	*0.98*	**1**	**1**	99.29	88.72	96.15	86.28	0.94	18	15	0.4	∅	*1*	**0**	∅	*1*	0.107	76	87
$T5_{vt}$	0.97	**1**	**1**	99.24	88.41	92.60	86.61	0.94	29	14	*0.2*	∅	*1*	*0.2*	∅	3	0.115	79	85
B_{vt1}	0.97	**1**	**1**	99.32	86.15	94.13	83.89	*0.99*	20	16	**0**	∅	**0**	**0**	∅	*1*	0.047	41	83
B_{vt}	0.97	**1**	**1**	99.34	85.68	93.52	83.64	0.98	13	14	**0**	∅	**0**	**0**	∅	*1*	0.053	43	83
$T5_{vT}$	0.82	**1**	**1**	99.25	88.41	92.60	86.61	0.97	810	15	0.8	0.5	2	0.4	∅	3	0.221	139	72
$T5_{vn}$	0.75	**1**	**1**	99.39	90.61	97.98	88.17	0.97	137	13	1.6	0.4	3	*0.2*	0.5	3	0.25	160	67
μ	0.98	1.00	1	99.49	91.42	96.02	89.87	0.96	63	14	0.3	∅	0.5	0.1	∅	1.5	0.11	73	89
σ	0.1	0.0	0	0.2	2.7	1.7	3	0.0	163	3	0.4	∅	0.7	0.2	∅	1	0.1	46	6.8

Table 1. Results for the best-performing configurations. This table is ordered based on the G_g score taking into account both triple validity and performances. In bold are the best results. In italics are the second-best results. The worse results are underlined. Averages are calculated over the 5 folds. The mean μ and standard deviation σ are provided for each metric. As a reminder the syntax notation is: XML-RDF ($_x$), Turtle ($_T$), Turtle Light ($_t$), N-Triples ($_n$), JSON-LD ($_j$), list ($_l$) and tags ($_g$).

References

1. Banerjee, D., Nair, P.A., Kaur, J.N., Usbeck, R., Biemann, C.: Modern Baselines for SPARQL Semantic Parsing. In: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 2260–2265. SIGIR '22, Association for Computing Machinery, New York, NY, USA (Jul 2022). https://doi.org/10.1145/3477495.3531841, https://doi.org/10.1145/3477495.3531841
2. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Burstein, J., Doran, C., Solorio, T. (eds.) 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). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1423, https://aclanthology.org/N19-1423
3. Ding, N., Qin, Y., Yang, G., Wei, F., Yang, Z., Su, Y., Hu, S., Chen, Y., Chan, C.M., Chen, W., Yi, J., Zhao, W., Wang, X., Liu, Z., Zheng, H.T., Chen, J., Liu, Y., Tang, J., Li, J., Sun, M.: Parameter-efficient fine-tuning of large-scale pre-trained language models. Nat Mach Intell **5**(3), 220–235 (Mar 2023). https://doi.org/10.1038/s42256-023-00626-4, https://www.nature.com/articles/s42256-023-00626-4, number: 3 Publisher: Nature Publishing Group
4. Han, R., Peng, T., Yang, C., Wang, B., Liu, L., Wan, X.: Is Information Extraction Solved by ChatGPT? An Analysis of Performance, Evaluation Criteria, Robustness and Errors (May 2023). https://doi.org/10.48550/arXiv.2305.14450, http://arxiv.org/abs/2305.14450, arXiv:2305.14450 [cs]
5. Harbecke, D., Chen, Y., Hennig, L., Alt, C.: Why only micro-f1? class weighting of measures for relation classification. In: Shavrina, T., Mikhailov, V., Malykh, V., Artemova, E., Serikov, O., Protasov, V. (eds.) Proceedings of NLP Power! The First Workshop on Efficient Benchmarking in NLP. pp. 32–41. Association for Computational Linguistics, Dublin, Ireland (May 2022). https://doi.org/10.18653/v1/2022.nlppower-1.4, https://aclanthology.org/2022.nlppower-1.4
6. Hofer, M., Obraczka, D., Saeedi, A., Köpcke, H., Rahm, E.: Construction of Knowledge Graphs: State and Challenges (Oct 2023). https://doi.org/10.48550/arXiv.2302.11509, http://arxiv.org/abs/2302.11509, arXiv:2302.11509 [cs]
7. Huguet Cabot, P.L., Navigli, R.: REBEL: Relation Extraction By End-to-end Language generation. In: Moens, M.F., Huang, X., Specia, L., Yih, S.W.t. (eds.) Findings of the Association for Computational Linguistics: EMNLP 2021. pp. 2370–2381. Association for Computational Linguistics, Punta Cana, Dominican Republic (Nov 2021). https://doi.org/10.18653/v1/2021.findings-emnlp.204, https://aclanthology.org/2021.findings-emnlp.204
8. Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y.J., Madotto, A., Fung, P.: Survey of hallucination in natural language generation. ACM Comput. Surv. **55**(12) (mar 2023). https://doi.org/10.1145/3571730, https://doi.org/10.1145/3571730
9. Jin, B., Liu, G., Han, C., Jiang, M., Ji, H., Han, J.: Large Language Models on Graphs: A Comprehensive Survey (Dec 2023), http://arxiv.org/abs/2312.02783, arXiv:2312.02783 [cs]

10. Josifoski, M., De Cao, N., Peyrard, M., Petroni, F., West, R.: GenIE: Generative information extraction. In: Carpuat, M., de Marneffe, M.C., Meza Ruiz, I.V. (eds.) Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 4626–4643. Association for Computational Linguistics, Seattle, United States (Jul 2022). https://doi.org/10.18653/v1/2022.naacl-main.342, https://aclanthology.org/2022.naacl-main.342
11. Josifoski, M., Sakota, M., Peyrard, M., West, R.: Exploiting asymmetry for synthetic training data generation: SynthIE and the case of information extraction. In: Bouamor, H., Pino, J., Bali, K. (eds.) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing. pp. 1555–1574. Association for Computational Linguistics, Singapore (Dec 2023). https://doi.org/10.18653/v1/2023.emnlp-main.96, https://aclanthology.org/2023.emnlp-main.96
12. Ke, P., Ji, H., Ran, Y., Cui, X., Wang, L., Song, L., Zhu, X., Huang, M.: JointGT: Graph-text joint representation learning for text generation from knowledge graphs. In: Zong, C., Xia, F., Li, W., Navigli, R. (eds.) Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021. pp. 2526–2538. Association for Computational Linguistics, Online (Aug 2021). https://doi.org/10.18653/v1/2021.findings-acl.223, https://aclanthology.org/2021.findings-acl.223
13. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., Zettlemoyer, L.: BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension (Oct 2019). https://doi.org/10.48550/arXiv.1910.13461, http://arxiv.org/abs/1910.13461, arXiv:1910.13461 [cs, stat]
14. Li, X., Polat, F., Groth, P.: Do Instruction-tuned Large Language Models Help with Relation Extraction?
15. Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., Neubig, G.: Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. ACM Comput. Surv. **55**(9) (jan 2023). https://doi.org/10.1145/3560815, https://doi.org/10.1145/3560815
16. Lu, Y., Liu, Q., Dai, D., Xiao, X., Lin, H., Han, X., Sun, L., Wu, H.: Unified structure generation for universal information extraction. In: Muresan, S., Nakov, P., Villavicencio, A. (eds.) Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 5755–5772. Association for Computational Linguistics, Dublin, Ireland (May 2022). https://doi.org/10.18653/v1/2022.acl-long.395, https://aclanthology.org/2022.acl-long.395
17. Mihindukulasooriya, N., Sava, M., Rossiello, G., Chowdhury, M.F.M., Yachbes, I., Gidh, A., Duckwitz, J., Nisar, K., Santos, M., Gliozzo, A.: 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. p. 827–844. Springer-Verlag, Berlin, Heidelberg (2022). https://doi.org/10.1007/978-3-031-19433-7_47, https://doi.org/10.1007/978-3-031-19433-7_47
18. Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., Gao, J.: Large language models: A survey (2024)

19. Nayak, T., Majumder, N., Goyal, P., Poria, S.: Deep neural approaches to relation triplets extraction: a comprehensive survey. Cognitive Computation **13**, 1215 – 1232 (2021), https://api.semanticscholar.org/CorpusID:232427782
20. Paolini, G., Athiwaratkun, B., Krone, J., Ma, J., Achille, A., Anubhai, R., dos Santos, C.N., Xiang, B., Soatto, S.: Structured prediction as translation between augmented natural languages. In: 9th International Conference on Learning Representations, ICLR 2021 (2021)
21. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: Bleu: a method for automatic evaluation of machine translation. In: Isabelle, P., Charniak, E., Lin, D. (eds.) Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics. pp. 311–318. Association for Computational Linguistics, Philadelphia, Pennsylvania, USA (Jul 2002). https://doi.org/10.3115/1073083.1073135, https://aclanthology.org/P02-1040
22. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer (Sep 2023), http://arxiv.org/abs/1910.10683, arXiv:1910.10683 [cs, stat]
23. Ringwald, C., Gandon, F., Faron, C., Michel, F., Abi Akl, H.: 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, pp. 23631–23632 (2024)
24. Shenoy, K., Ilievski, F., Garijo, D., Schwabe, D., Szekely, P.: A study of the quality of Wikidata. Journal of Web Semantics **72**, 100679 (Apr 2022). https://doi.org/10.1016/j.websem.2021.100679, https://www.sciencedirect.com/science/article/pii/S1570826821000536
25. Smirnova, A., Cudré-Mauroux, P.: Relation extraction using distant supervision: A survey. ACM Comput. Surv. **51**(5) (nov 2018). https://doi.org/10.1145/3241741, https://doi.org/10.1145/3241741
26. Sun, K., Qi, P., Zhang, Y., Liu, L., Wang, W.Y., Huang, Z.: Tokenization Consistency Matters for Generative Models on Extractive NLP Tasks (Oct 2023). https://doi.org/10.48550/arXiv.2212.09912, http://arxiv.org/abs/2212.09912, arXiv:2212.09912 [cs]
27. Taillé, B., Guigue, V., Scoutheeten, G., Gallinari, P.: Let's Stop Incorrect Comparisons in End-to-end Relation Extraction! In: Webber, B., Cohn, T., He, Y., Liu, Y. (eds.) Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 3689–3701. Association for Computational Linguistics, Online (Nov 2020). https://doi.org/10.18653/v1/2020.emnlp-main.301, https://aclanthology.org/2020.emnlp-main.301
28. Wang, C., Liu, X., Chen, Z., Hong, H., Tang, J., Song, D.: DeepStruct: Pretraining of language models for structure prediction. In: Muresan, S., Nakov, P., Villavicencio, A. (eds.) Findings of the Association for Computational Linguistics: ACL 2022. pp. 803–823. Association for Computational Linguistics, Dublin, Ireland (May 2022). https://doi.org/10.18653/v1/2022.findings-acl.67, https://aclanthology.org/2022.findings-acl.67
29. Ye, H., Zhang, N., Chen, H., Chen, H.: Generative knowledge graph construction: A review. CoRR **abs/2210.12714** (2022). https://doi.org/10.48550/arXiv.2210.12714, https://doi.org/10.48550/arXiv.2210.12714