HyperPIE: Hyperparameter Information Extraction from Scientific Publications

Tarek Saier[1], Mayumi Ohta[2], Takuto Asakura[3], and Michael Färber[1]

[1] Karlsruhe Institute of Technology, Karlsruhe, Germany
{tarek.saier,michael.faerber}@kit.edu
[2] Fraunhofer Institute for Systems and Innovation Research, Karlsruhe, Germany
mayumi.ohta@isi.fraunhofer.de
[3] The University of Tokyo, Tokyo, Japan
takuto@is.s.u-tokyo.ac.jp

Abstract. Automatic extraction of information from publications is key to making scientific knowledge machine-readable at a large scale. The extracted information can, for example, facilitate academic search, decision making, and knowledge graph construction. An important type of information not covered by existing approaches is hyperparameters. In this paper, we formalize and tackle hyperparameter information extraction (HyperPIE) as an entity recognition and relation extraction task. We create a labeled data set covering publications from a variety of computer science disciplines. Using this data set, we train and evaluate BERT-based fine-tuned models as well as five large language models: GPT-3.5, GALACTICA, Falcon, Vicuna, and WizardLM. For fine-tuned models, we develop a relation extraction approach that achieves an improvement of 29% F_1 over a state-of-the-art baseline. For large language models, we develop an approach leveraging YAML output for structured data extraction, which achieves an average improvement of 5.5% F_1 in entity recognition over using JSON. With our best performing model we extract hyperparameter information from a large number of unannotated papers, and analyze patterns across disciplines. All our data and source code is publicly available at https://github.com/IllDepence/hyperpie.

Keywords: Information Extraction, Scientific Text, Hyperparameter

1 Introduction

Models capable of extracting fine-grained information from publications can make scientific knowledge machine-readable at a large scale. Aggregated, such information can fuel platforms like Papers with Code[4] and the Open Research Knowledge Graph [26,3], and thereby facilitate academic search, recommendation, and reproducibility. Accordingly, a variety of approaches for information extraction (IE) from scientific text have been proposed [19,13,12,16,11].

[4] See https://paperswithcode.com/.

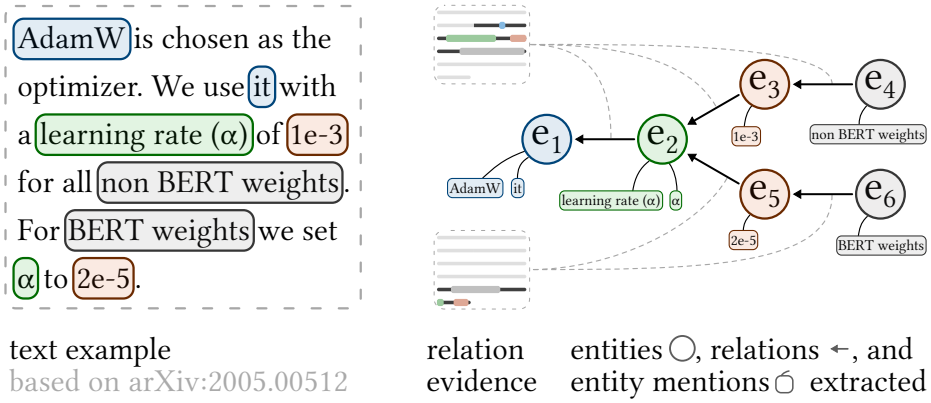


Fig. 1: Illustration of hyperparameter information in a text example alongside the extracted entities and relations. Entity types are research artifact, parameter, value, and context. Relations are indicated by arrows.

However, to the best of our knowledge, no approaches exist for the extraction of structured information on hyperparameter use from publications. That is, information on *with which parameters* researchers use methods and data. We refer to this information as "hyperparameter information" (see Fig. 1). Hyperparameter information is important for several reasons. (1) First, its existence in a paper is an indicator for reproducibility [22] and, when extracted automatically, can improve automated reproduction of results [25]. (2) Second, in aggregate it can inform on both conventions in a field as well as trends over time. (3) Lastly, it enables more fine-grained paper representations benefiting downstream applications based on document similarity, such as recommendation and search. Hyperparameter information is challenging to extract, because (1) it is usually reported in a dense format, (2) often includes special notation, and (3) operates on domain specific text (e.g. "For Adam we set α and β to 1e-3 and 0.9 respectively.").

To address the lack of approaches for extracting this type of information, we define the task of "hyperparameter information extraction" (HyperPIE) and develop several approaches to it. Specifically, we formalize HyperPIE as an entity recognition (ER) and relation extraction (RE) task. We create a labeled data set spanning a variety of computer science disciplines from machine learning (ML) and related areas. The data set is created by manual annotation of paper full-texts, which is accelerated by a pre-annotation mechanism based on an external knowledge base. Using our data set, we train and evaluate both BERT-based [10] fine-tuned models as well as large language models (LLMs). For the former, we develop a dedicated relation extraction model that achieves an improvement of 29% F_1 compared to a state-of-the-art baseline. For LLMs, we develop an approach leveraging YAML output for structured data extraction, which achieves a consistent improvement in entity recognition across all tested models, averag-

ing at 5.5% F_1. Using our best performing model, we extract hyperparameter information from 15,000 unannotated papers, and analyze patterns across ML disciplines of how authors report hyperparameters. All our data and source code is made publicly available.[5] In summary, we make the following contributions.

1. We formalize a novel and relevant IE task (HyperPIE).
2. We create a high quality, manually labeled data set from paper full-texts, enabling the development and study of approaches to the task.
3. We develop two lines of approaches to HyperPIE and achieve performance improvements in both of them over solutions based on existing work.
4. We demonstrate the utility of our approaches by application on large-scale, unannotated data, and analyze the extracted hyperparameter information.

2 Related Work

Fine-Tuned Models Named entity recognition (NER) and RE from publications in ML and related fields have been tackled by SciERC [19] and subsequently SciREX [13]. The scope considered are methods, tasks, data sets, and evaluation metrics. Proposed methods for the task utilize BiLSTMs, BERT and SciBERT [5]. With both approaches, there is a partial overlap in entity types to our task, as we also extract methods and data sets. The key difference arises though the nature of parameters and values we relate them to, and the challenges in extracting those as briefly laid out in the introduction.

IE models aiming to relate natural language to numerical values and mathematical symbols have been introduced at SemEval 2021 Task 8 [12] and SemEval 2022 Task 12 [16] respectively. Most of the proposed models base their processing of natural language on BERT or SciBERT. To handle numbers and symbols rendered in LaTeX, as well as to accomplish RE between entity types with highly regular writing conventions (e.g. numbers and units such as "5 ms"), rule-based approaches or dedicated smaller neural networks are commonly used.

Similarly, we find a level of regularity in how authors report parameters and values, and make use of that in our approach accordingly. In line with related work using fine-tuned models, we also use BERT and SciBERT for contextualized token embeddings.

LLMs With the recent advances in LLMs, there has been a surge in efforts to utilize them for IE from scientific text. Nevertheless, their performance is not on par with dedicated models for NER and RE yet [32].

An improtant concept for IE with LLMs is introduced by Agrawal et al. [1]: a "resolver" is a function that maps the potentially ambiguous output of an LLM to a defined, task specific output space. In their work, the authors extract singular values and lists from clinical notes using GPT-3. They use a variety of resolvers that perform steps like tokenization, removal of specific symbols or words, and pattern matching using regular expressions.

[5] See https://github.com/IllDepence/hyperpie.

Work with similar output data complexity (values and lists) has also been done in the area of material science. Xie et al. [30] use GPT-3.5 to extract information on solar cells from paper full-text. Similarly, Polak et al. [20] use ChatGPT to extract material, value, and unit information from sentences of material science papers. They define a conversational progression, in which they prompt the model generate tables, which are processed using simple string parsing rules.

An approach for IE of more complex information is proposed by Dunn et al. [11]. They use GPT-3.5 to extract material information from materials chemistry papers. Given the hierarchical nature of the information to be extracted, the authors find simple output formats insufficient. To overcome this, they prompt the model to output the data in JSON format.

Given hyperparameter information also is hierarchical (see Fig. 1), we adopt prompting LLMs to output data in a text based data serialization format. Different from the related work introduced above, we do not limit our experiments to API access based closed source LLMs, but also evaluate various open LLMs, because we recognize the importance of contributing efforts to the advancement of the more transparent, accountable, and reproducibility friendly side of this new and rapidly evolving area of research [17].

Besides IE from scientific publications, there have been efforts to extract hyperparameter schemata and constraints from Python docstrings [4] using CNL grammars [15], and from Python code [23] using static analysis. Compared to our task setting, these rely on a known context (e.g. a `fit` method) and operate on constrained input (generated docstrings and source code instead).

3 Hyperparameter Information Extraction

3.1 Task Definition

We define HyperPIE as an ER+RE task with four entity classes "research artifact", "parameter", "value", and "context", and a single relation type. Briefly illustrated by a minimal example, in the sentence *"During fine-tuning, we use the Adam optimizer with* $\alpha = 10^{-4}$*."*, the research artifact *Adam* has the parameter α which is set to the value 10^{-4} in the context *During fine-tuning*.

The entity classes are characterized as follows. A "research artifact", within the scope of our task, is an entity used for a specific purpose with a set of variable aspects that can be chosen by the user. These include methods, models, and data sets.[6] A "parameter" is a variable aspect of an artifact. This includes model parameters, but also, for example, the size of a sub-sample of a data set. A "value" expresses a numerical quantity and in our task is treated like an entity rather than a literal. Lastly, a "context" can be attached to a value if the value is only valid in that specific context. The single relation type relates entities as follows: parameter → research artifact, value → parameter, and context → value.

[6] Broader definitions in other contexts also include software in general, empirical laws, and ideas [18]. For our purposes, however, above specific definition is more useful.

Co-reference relations implicitly exist between the mentions of a common entity (e.g. "AdamW" and "it" in Fig. 1). That is, if an entity has multiple mentions within the text, they are considered co-references to each other.

The scope of the IE task comprises the extraction of entities, their relations, and the identification of all their mentions in the text (and thereby implicitly co-references). Furthermore, we specifically consider IE from text, and not from tables, graphs, or source code.[7]

3.2 Data Set Construction

Because HyperPIE is a novel task, we cannot rely on existing data sets for training and evaluating our approaches. We therefore create a new data set by manually annotating papers. As our data source we chose unarXive [24], because it includes paper full-texts and, most importantly, retains mathematical notation as LaTeX. This is crucial because parsing such notation from PDFs is prone to noise, which would be problematic for our parameter and value entities.

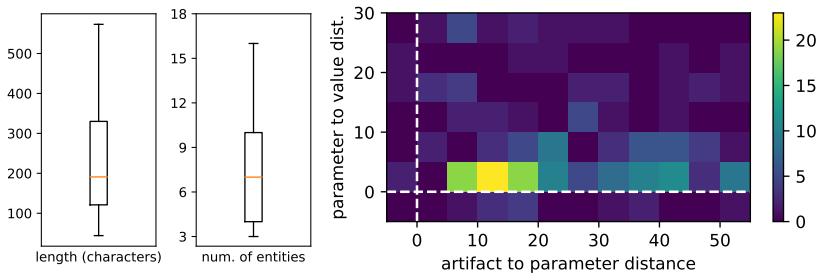
To ensure we cover a wide variety of artifacts and discipline specific writing conventions, we use papers from multiple ML related fields. Specifically, these are Machine Learning (ML), Computation and Language (CL), Computer Vision (CV), and Digital Libraries (DL), which make up 143,203 papers in unarXive.[8]

We base our annotation guidelines on the widely used ACL RD-TEC guideline[9] [21]. To make sure our resulting annotations are able to properly capture how authors report hyperparameters in text, we perform two annotation rounds: (1) an initial exploratory round, the results of which are used to refine the annotation guidelines and inform later model development, and (2) the main annotation round, the results of which constitute our data set used for model training and evaluation. In the following, both steps are described in more detail.

Initial Annotation Round We heuristically pre-filter our ML paper corpus for sections reporting on hyperparameters.[10] Annotators then inspect these sections, select a continuous segment of text that contain hyperparameter information, and make their annotations. This task is performed independently by two annotators and results in a total of 151 text segments (131 unique, 2×10 annotated by both). The annotated text segments contain 1,345 entities and 1,110 relations.

As shown in Figure 2a, we observe text segments reporting on hyperparameters to generally have a length below 600 characters. We furthermore see that most text segments contain between 3 and 15 entities. Lastly, in Figure 2b, show distances between artifacts and their parameters, as well as parameters and their values. We see that artifacts usually are mentioned before their parameters (78%), and parameters before their values (93%). The reverse cases

[7] We leave investigating multi-modal IE pipelines (text/code/graphs) for future work.
[8] The respective arXiv categories are cs.LG, cs.CL, cs.AI, and cs.DL. See https://arxiv.org/category_taxonomy for a more detailed description.
[9] See http://pars.ie/publications/papers/pre-prints/acl-rd-tec-guidelines-ver2.pdf.
[10] We filter based on key phrases ("use", "set", etc.), numbers, and LaTeX math content.



(a) text segments (b) relation distances (#chars)

Fig. 2: Observations of initial annotation round

also exists, but are less common. Additionally, we can see that values are most commonly reported right after their parameter, while there is a higher variability in distances between parameters and artifacts. Based on above observations we determine the unit of annotation for the final round to be one paragraph (on average 563.4 characters long in our corpus), as it is sufficient to capture hyperparameters being reported.

The inter annotator agreement (IAA, reported as Cohen's kappa) of the text segments annotated by both annotators is 0.867 for entities and 0.737 for relations[11] (strong to almost perfect agreement) which is compares favorably to SciERC [19] with an IAA of 0.769 for entities and 0.678 for relations.

Main Annotation Round In our main annotation round we annotate whole papers (paragraph by paragraph) instead of pre-filtered text-segments. This is done to ensure that the final annotation result reflects data as it will be encountered by a model during inference—that is, containing a realistic amount of paragraphs that have no information on hyperparameters, or, for example, only mention research artifacts but no parameters.

Similar to related work [13], we use Papers with Code as an external knowledge base to pre-annotate entity candidates to make the annotation process more efficient. In a similar fashion, we use annotator's previously annotated entity mentions for pre-annotation. Pre-annotated text spans are, as the name suggests, set automatically, but need to be checked by annotators manually.

Through this process we annotate 444 paragraphs, which contain 1,971 entities and 614 relations. The entity class distribution is 1,134 research artifacts, 131 parameters, 662 values, and 44 contexts. The annotation data is provided in a JSON structure as shown in Figure 1, as well as in the W3C Web Annotation Data Model[12] to facilitate easy re-use and compatibility with existing systems.

[11] Measured by the character level entity class and character level relation target span agreement respectively.

[12] See https://www.w3.org/TR/annotation-model/.

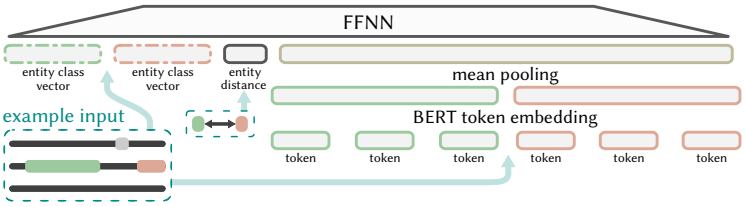


Fig. 3: RE with emphasis on entity candidate pair types and distance.

4 Methods

We approach hyperparameter information extraction in two ways. First, we build upon established ER+RE methods and develop an approach using a fine-tuned model in a supervised learning setting. Second, given the recent promising advances with LLMs, we develop an approach utilizing LLMs in a zero-shot and few-shot setting.

4.1 Fine-Tuned Models

We base our fine-tuned model approach on PL-Marker [33], the currently best performing model on SciERC. Specifically, we use the ER component of PL-Marker. Our reason is that (1) the text our model will be applied on is of the same type as in SciERC (ML publications), and (2) there is some correspondence between the entities to be identified—namely our entity class "research artifact" including methods and datasets, which are both entity classes in SciERC.

For RE we develop an approach that utilizes token embeddings as well as relative entity distance and entity class pairings. This is motivated by the fact that (a) we observed a high level or regularity in the relative distance of research artifact, parameter, and value mentions[13] (see Fig. 2), and (b) relations only exist between specific pairs of entity types.

In Figure 3 we show a schematic depiction of our new relation extraction component. Entity candidate pair classes as well as the relative distance between the entities in the text are used as a dedicated model input, BERT token embeddings of the entity mentions are combined using mean pooling. These inputs are fed into a feed-forward neural network FFNN for prediction. Formally, the model performs pairwise binary classification as $\text{FFNN}(E_0^c, E_1^c, E^d, E^T)$, where E_i^c are class vectors, E^d encodes candidate distance, and E^T is the token pair embedding calculated as $E^T = \frac{1}{|T|}\sum_{i=0}^{|T|} \text{BERT}(t_i)$, the mean of the pair's tokens $t_i \in |T|$.

During the development of our model we also experiment with concatenation in favor of mean pooling to preserve information on the order of the entities,

[13] We note that these observations where made during the initial exploratory annotation round (Sec. 3.2) and not during annotation of the evaluation data.

but find that mean pooling results in better performance. Furthermore, we investigated the use of SciBERT instead of BERT, but find that regular BERT embeddings give us better results, despite our model handling scientific text.

4.2 LLM

We develop our LLM approach for a zero-shot and a few-shot setting. This means the models perform the IE task based on either instructions only (zero-shot), or instructions and a small number of examples (few-shot).

Performing IE using LLMs in zero-shot or few-shot settings requires the desired structure of the output data to be specified within the model input. In simple cases (e.g. numbers or yes/no decisions) this can be achieved by an inline specification of the format in natural language (e.g. "The answer (arabic numerals) is") [14]. IE from scientific publications, however, often seeks to extract more complex information. To achieve this, the model can be tasked to produce output in a text based data serialization format such as JSON, as done in previous work [11]. Especially for complex structured predictions, few-shot prompting has been shown to further boost in-context learning (ICL) accuracy and consistency at inference time [6].

Drawing from techniques used in previous work approaching other IE tasks, we investigate several prompting strategies to build our approach.

1. *Multi-stage prompting* [20]: first determine the presence of hyperparameters information; if present, extract the list of entities; lastly, determine relations.
2. *In-text annotation* [29]: let the input text be repeated with entity annotations, e.g. repeat "We use BERT for ..." as "We use [a1|BERT] for ...".
3. *Data serialization format* [11]: specify a serialization format in the promt that is parsed afterwards; then match in-text mentions in the input.
4. *(3)+(2)*: prompt as in (3); then match in-text mentions using (2).

We find (1) to lead to problems with errors propagation along steps, and with (2) and (4) we frequently see alterations in the reproduced text. Accordingly, we use prompt type (3) for our approach—specifying a data serialization format in the prompt. While existing work uses the JSON format for this [11], we use YAML, as it is less prone to "delimiter collision" problems due to its minimal requirements for structural characters.[14] In doing so, we expect to avoid problems with LLM output not being parsable. Our overall LLM approach looks as follows.

Zero-shot: We build our zero-shot prompts from the following consecutive components: `[task]` `[input text]` `[format]` `[completion prefix]` In `[task]` we specify the information to extract, i.e. research artifacts, their parameters, etc. `[input text]` is the paragraph from which to extract the information. `[format]` defines the output YAML schema. `[completion prefix]` is a piece of text that directly precedes the LLM's output, such as *"ASSISTANT: "*. To then generate predictions based on LLM output, we pass it to a standard YAML parser after cleansing (e.g. removing text around the YAML block). For each used LLM

[14] See https://yaml.org/spec/1.2.2/.

model, we individually perform prompt tuning. Here we determine, for example, if a model gives better results when the `[completion prefix]` includes the beginning of the serialized output data (e.g, "`---\ntext_contains_entities: `") or if this leads to a deterioration in output quality.

Few-shot: Our few shot approach makes the following adjustments to the method described above. Prompts additionally include a component `[examples]`, which are valid input output pairs sampled by their similarity to the input text. Specifically, for an input text from a document X, we sample the 5 most similar paragraphs from all ground truth documents excluding X. As these examples can be confused with the input text, we place it after the examples, resulting in the structure `[task][format][examples][input text][completion prefix]`.

LLMs reaching a sufficient context size for a few-shot approach to our task are a recent development. We can therefore additionally make use of other recently added capabilities. Specifically, we make use of generation constrains via a gBNF grammar[15] to enforce LLM output according to our data scheme, allowing us to mitigate parsing errors.

5 Experiments

We evaluate the fine-tuned models and LLM approach against baselines from existing work. Both evaluations are performed on our data set described in Section 3.2. Metrics used to measure prediction performance are precision, recall and F_1 score, abbreviated as P, R and F_1 respectively.

5.1 Fine-Tuned Models

We use PL-Marker, the currently best performing model on SciERC, as our baseline. Models are trained and evaluated using 5-fold cross validation (3 folds training, 1 dev, 1 test). We train the ER component of PL-Marker as done in [33], using *scibert-scivocab-uncased* as the encoder, Adam as the optimizer, a learning rate of 2e-5, and for 50 epochs. The PL-Marker RE component is trained using *bert-base-uncased*, Adam, a learning rate of 2e-5, and for 10 epochs. Our own RE component also uses *bert-base-uncased*, Adam as the optimizer, and is trained with a learning rate of 1e-3 for 90 epochs. Models are trained and evaluated on a single GeForce RTX 3090.

Results In Figure 4 we show the results of PL-Marker ER (used for both models) as well as the PL-Marker RE component and our RE model. For ER we evaluate exact matches (no partial token overlap). In the case of RE, each entity pair is predicted as having a relation or not—as there is just one relation type.

Mean ER performance is 81.5, 76.8, and 79.0 (P, R, F_1). For RE, the precision of PL-Marker and our model are similar at 33.5 and 30.7 respectively, but our model performs more consistent. PL-Marker only achieves a very low recall of

[15] See https://github.com/ggerganov/llama.cpp/pull/1773.

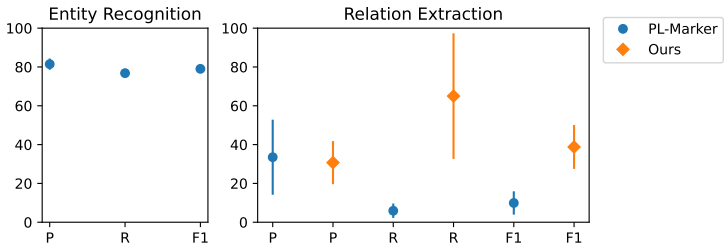


Fig. 4: Fine-tuned model evaluation (5-fold cross validation).

5.9, whereas our model, while showing large variability, achieves a mean of 65.0. The resulting F_1 scores are 9.9 for PL-Marker and 38.8 for our model.

Analysis Token level ER performance across entity classes (none, artifact, parameter, value, context) is at 98.5%, 77.8%, 47.9%, 84.4%, 0% F_1. That is, the model does predict contexts and struggles with parameters, but artifacts and values are predicted reliably. For our RE model, we observe that value-parameter relations are more reliably predicted than parameter-artifact relations.

Table 1: Ablation study

Used	P [%]	R [%]	F_1 [%]
␣CD	15.5	8.8	11.1
T␣D	16.6	29.8	19.6
TC␣	26.5	65.0	35.5
TCD	**30.7**	**65.0**	**38.8**

To assess the impact of the different components in our RE model, we perform an ablation study with the same 5-fold cross-validation setup as above. In Table 1, showing its results, we can see that removing the BERT token embeddings (T) results in the largest performance loss, followed by entity class embeddings (C) and entity distance (D). Removing any of the inputs results in worse predictions.

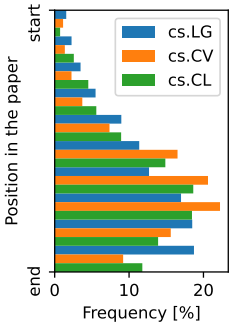


Fig. 5: Mentioning position in papers.

Finally, we apply our full model to a random sample of 15,000 papers. Analyzing the results, we find hyperparameters (artifact, parameter, value triples) are reported in 36% of ML papers, 42% of CV papers, 36% of CL papers, and 7% of DL papers. In Figure 5 we further look at the distribution of the information across the length of papers (excluding DL as not being representative). We can see a clear tendency towards the latter half of papers.

5.2 LLMs

For our LLM experiments we chose a variety of models, with sizes ranging from 13 B to 175 B parameters, as shown in Table 2. We chose WizardLM [31] as it is meant to handle complex instructions, Vicuna [9] due to its performance relative

Table 2: LLM selection (size in number of parameters).

Model	Variant	Size
WizardLM [31]	`WizardLM-13B-V1.1`	13 B
Vicuna$_{4k}$ [9]	`vicuna-13b-v1.3`	13 B
Vicuna$_{16k}$ [9]	`vicuna-13b-v1.5-16k`	13 B
Falcon [2]	`falcon-40b-instruct`	40 B
GALACTICA [28]	`galactica-120b`	120 B
GPT-3.5 [7]	`text-davinci-003`	175 B

Table 3: Prediction performance of LLM models. Subscripts ($_{\Delta\pm n}$) show the delta in F_1 from JSON to YAML output of each model. Format: **best**, second.

Zero-shot		Entity Recognition			Relation Extraction		
Model	Output	P [%]	R [%]	F_1 [%]	P [%]	R [%]	F_1 [%]
WizardLM	JSON	6.9	11.3	8.6	0.1	0.8	0.1
	YAML	9.7	35.6	15.3$_{\Delta+6.7}$	0.1	1.5	0.1$_{\Delta+0.0}$
Vicuna$_{4k}$	JSON	15.1	9.3	11.5	0.7	3.8	1.2
	YAML	17.3	31.5	22.3$_{\Delta+10.8}$	0.0	0.8	0.1$_{\Delta-1.1}$
Falcon	JSON	**37.1**	5.9	10.2	0.0	0.0	0.0
	YAML	32.7	14.2	19.8$_{\Delta+9.6}$	0.0	0.0	0.0$_{\Delta+0.0}$
GALACTICA	JSON	25.9	15.7	19.5	0.1	2.3	0.3
	YAML	23.1	19.5	21.1$_{\Delta+1.6}$	0.0	0.8	0.1$_{\Delta-0.2}$
GPT-3.5	JSON	27.9	**42.8**	33.8	5.4	10.7	7.2
	YAML	34.0	41.7	**37.4**$_{\Delta+3.6}$	**5.8**	**12.2**	**7.8**$_{\Delta+0.6}$
5-shot		Entity Recognition			Relation Extraction		
Vicuna$_{16k}$	JSON	34.4	46.7	39.6	0.8	4.6	1.3
	YAML	**43.9**	**44.1**	**44.0**$_{\Delta+0.4}$	**4.5**	**9.9**	**6.1**$_{\Delta+4.8}$

to its size, Falcon [2] because of its alleged performance, and GALACTICA [28] because it was trained on scientific text. Vicuna$_{16k}$ is a model extended using Position Interpolation [8] based on Rotary Positional Embeddings [27], which makes it the only model in our experiments with a sufficient context size for a few-shot evaluation.

The models are run as follows. GPT-3.5 is accessed through its official API. All open models are run on a high performance compute cluster. Vicuna$_{4k}$ and WizardLM are run on nodes with 4×NVIDIA Tesla V100. GALACTICA, Falcon, and Vicuna$_{16k}$ are run on nodes with 4×NVIDIA A100.

As a baseline, we use a JSON variant for each model, where the `[format]` and `[examples]` compontents of prompts use JSON, and compare it to the respective YAML version. All models are used with greedy decoding (temperature = 0) for the sake of reproducibility.

Results In Table 3, show the prediction performance of all models and prompt variants. Overall, LLM performance does not reach the level of our pre-trained models. For zero-shot, we observe the best performance with both GPT-3.5

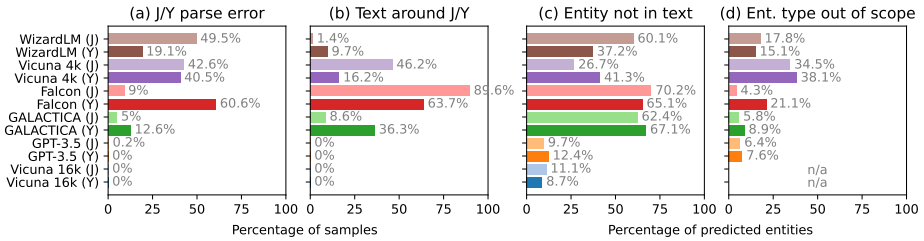


Fig. 6: Parsing success, format adherence, hallucinations, and scope adherence of LLM generated JSON (J) and YAML (Y).

variants, where YAML outperforms JSON (+3.6% ER and +0.6% RE in F_1 score). The second highest ER F_1 score by model is achieved by Vicuna$_{4k}$ (22.3), despite its size being less than a 10th that of GPT-3.5. For RE, however, even the best model only reaches 7.8%. With our few-shot approach, we are able to considerably improve performance between Vicuna models (+27% ER and +6% RE in F_1), surpassing the zero-shot performance of GPT-3.5 in ER. Lastly, we see that using YAML leads to better ER results accross all six models, with ER performance being comparable or improved as well.

Analysis In Figure 6 we show an analysis of the steps leading up to model prediction. Focussing first on the zero-shot models (upper five) we observe the following across the four plots from left to right. (a) For three of five models, prompting for YAML leads to fewer parsing errors. (b) Unwanted text around the extracted data is generated more/less by two models each. (c) Hallucinated entities and (d) out of scope entities appear overall slightly more often for in YAML compared to JSON. For our few shot approach (bottom model), we see that the use of a grammar (a, b) prevents all output format issues. Furthermore (c) hallucinated entities are reduced. (d) Out of scope entities can not be evaluated, because our in-context examples lead to frequent omission of type information in the output.

Through manual analysis, we find that "entities not in the text" can arise from unsolicited LaTeX parsing by the LLM (e.g. "`\lambda`" in text → "λ" in YAML). Prompting for *verbatim* parameter/value strings did not mitigate this.

6 Discussion

Our overall results, with a top performance of 79% F_1 for ER and 39% for RE, show that HyperPIE can be accomplished to a degree that yields sound results, but challenges still remain. Our novel data set enables further development of approaches from hereon. Our IE results on large-scale unannotated data give an indication of possible downstream analyses and applications. Here we see large potential for reproducibility research, faceted search, and recommendation.

Our LLM evaluation shows that for IE tasks dealing with complex information, the choice of text based data serialization format can have a considerable impact on performance, even when using grammar based generation constrains. Additionally, we can see that in-context learning enabled by larger context sizes, as well as grammars are an effective method to improve IE performance.

Limitations: Our work considers HyperPIE from text. This is sensible for a focussed approach, but downstream applications could furthermore benefit from composite pipelines also targeting extraction from tables, source code, etc. Our work does not test transferability of methods to domains outside of ML related fields. It would require domain expertise to find useful definitions for hyperparameters in each respective domain. Lastly, our data and experiments unfortunately are limited to English text only and do not cover other languages.

7 Conclusion

We formalize the novel ER+RE task HyperPIE and develop approaches for it, thereby expanding IE from scientific text to hyperparameter information. To this end, we create a manually labeled data set spanning various ML fields. In a supervised learning setting, we propose a BERT-based model that achieves an improvement of 29% F_1 in RE compared to a state-of-the-art baseline. Using the model, we perform IE on a large amount of unannotated papers, and analyze patterns of hyperparameter reporting across ML disciplines. In a zero-/few-shot setting, we propose an LLM based approach using YAML for complex IE, achieving an average improvement of 5.5% F_1 in ER over using JSON. We furthermore achieve large performance gains for LLMs using grammar based generation constrains and in-context learning. In future work, we plan to investigate fine-tuning LLMs, as well as additional practical use cases for data extracted from large publication corpora, such as knowledge graph construction.

Author Contributions

Tarek Saier: Conceptualization, Data curation, Formal analysis, Methodology, Software, Visualization, Writing – original draft, Writing – review & editing. Mayumi Ohta: Conceptualization (LLM few-shot), Formal analysis (LLM few-shot), Methodology (LLM few-shot), Software (LLM few-shot), Writing – original draft (support). Takuto Asakura: Conceptualization, Writing – review & editing. Michael Färber: Writing – review & editing.

Acknowledgements

This work was partially supported by the German Federal Ministry of Education and Research (BMBF) via [KOM,BI], a Software Campus project (01IS17042). The authors acknowledge support by the state of Baden-Württemberg through bwHPC. We thank Nicholas Popovic for extensive feedback on the experiment

design and prompt engineering. We thank Tarek Gaddour for feedback during the annotation scheme development, and Xiao Ning for input during early model development.

References

1. Agrawal, M., Hegselmann, S., Lang, H., Kim, Y., Sontag, D.: Large language models are few-shot clinical information extractors. In: Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing. pp. 1998–2022 (Dec 2022)
2. Almazrouei, E., Alobeidli, H., Alshamsi, A., Cappelli, A., Cojocaru, R., Debbah, M., Goffinet, E., Heslow, D., Launay, J., Malartic, Q., Noune, B., Pannier, B., Penedo, G.: Falcon-40B: an open large language model with state-of-the-art performance (2023)
3. Auer, S., Oelen, A., Haris, M., Stocker, M., D'Souza, J., Farfar, K.E., Vogt, L., Prinz, M., Wiens, V., Jaradeh, M.Y.: Improving access to scientific literature with knowledge graphs. Bibliothek Forschung und Praxis **44**(3), 516–529 (2020). https://doi.org/10.1515/bfp-2020-2042
4. Baudart, G., Kirchner, P.D., Hirzel, M., Kate, K.: Mining documentation to extract hyperparameter schemas. In: Proceedings of the 7th ICML Workshop on Automated Machine Learning (AutoML 2020) (2020)
5. Beltagy, I., Lo, K., Cohan, A.: SciBERT: A pretrained language model for scientific text. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 3615–3620. Association for Computational Linguistics (Nov 2019). https://doi.org/10.18653/v1/D19-1371
6. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)
7. Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS'20 (2020)
8. Chen, S., Wong, S., Chen, L., Tian, Y.: Extending context window of large language models via positional interpolation. arXiv preprint arXiv:2306.15595 (2023)
9. Chiang, W.L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J.E., Stoica, I., Xing, E.P.: Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality (March 2023), https://lmsys.org/blog/2023-03-30-vicuna/
10. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: 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). pp. 4171–4186. Association for Computational Linguistics (Jun 2019). https://doi.org/10.18653/v1/N19-1423

11. Dunn, A., Dagdelen, J., Walker, N., Lee, S., Rosen, A.S., Ceder, G., Persson, K., Jain, A.: Structured information extraction from complex scientific text with fine-tuned large language models (Dec 2022). https://doi.org/10.48550/arXiv.2212.05238
12. Harper, C., Cox, J., Kohler, C., Scerri, A., Daniel Jr., R., Groth, P.: SemEval-2021 task 8: MeasEval – extracting counts and measurements and their related contexts. In: Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021). pp. 306–316 (Aug 2021). https://doi.org/10.18653/v1/2021.semeval-1.38
13. Jain, S., van Zuylen, M., Hajishirzi, H., Beltagy, I.: SciREX: A Challenge Dataset for Document-Level Information Extraction. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. pp. 7506–7516. Association for Computational Linguistics (Jul 2020). https://doi.org/10.18653/v1/2020.acl-main.670
14. Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large Language Models are Zero-Shot Reasoners. Advances in Neural Information Processing Systems **35**, 22199–22213 (Dec 2022)
15. Kuhn, T.: A survey and classification of controlled natural languages. Comput. Linguist. **40**(1), 121–170 (mar 2014). https://doi.org/10.1162/COLI_a_00168
16. Lai, V., Pouran Ben Veyseh, A., Dernoncourt, F., Nguyen, T.: SemEval 2022 task 12: Symlink - linking mathematical symbols to their descriptions. In: Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022). pp. 1671–1678 (Jul 2022). https://doi.org/10.18653/v1/2022.semeval-1.230
17. Liesenfeld, A., Lopez, A., Dingemanse, M.: Opening up chatgpt: Tracking openness, transparency, and accountability in instruction-tuned text generators. In: Proceedings of the 5th International Conference on Conversational User Interfaces. CUI '23, New York, NY, USA (2023). https://doi.org/10.1145/3571884.3604316
18. Lin, J., Yu, Y., Song, J., Shi, X.: Detecting and analyzing missing citations to published scientific entities. Scientometrics **127**(5), 2395–2412 (May 2022). https://doi.org/10.1007/s11192-022-04334-5
19. Luan, Y., He, L., Ostendorf, M., Hajishirzi, H.: Multi-task identification of entities, relations, and coreferencefor scientific knowledge graph construction. In: Proc. Conf. Empirical Methods Natural Language Process. (EMNLP) (2018)
20. Polak, M.P., Morgan, D.: Extracting Accurate Materials Data from Research Papers with Conversational Language Models and Prompt Engineering – Example of ChatGPT (Mar 2023). https://doi.org/10.48550/arXiv.2303.05352
21. QasemiZadeh, B., Schumann, A.K.: The ACL RD-TEC 2.0: A language resource for evaluating term extraction and entity recognition methods. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16). pp. 1862–1868. European Language Resources Association (ELRA) (May 2016)
22. Raff, E.: A step toward quantifying independently reproducible machine learning research. In: Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019)
23. Rak-Amnouykit, I., Milanova, A., Baudart, G., Hirzel, M., Dolby, J.: Extracting Hyperparameter Constraints from Code. In: ICLR Workshop on Security and Safety in Machine Learning Systems (May 2021), https://hal.science/hal-03401683
24. Saier, T., Krause, J., Färber, M.: unarXive 2022: All arXiv Publications Pre-Processed for NLP, Including Structured Full-Text and Citation Network. In: Proceedings of the 23rd ACM/IEEE Joint Conference on Digital Libraries. JCDL '23 (2023)

25. Sethi, A., Sankaran, A., Panwar, N., Khare, S., Mani, S.: Dlpaper2code: Auto-generation of code from deep learning research papers. Proceedings of the AAAI Conference on Artificial Intelligence **32**(1) (Apr 2018). https://doi.org/10.1609/aaai.v32i1.12326
26. Stocker, M., Oelen, A., Jaradeh, M.Y., Haris, M., Oghli, O.A., Heidari, G., Hussein, H., Lorenz, A.L., Kabenamualu, S., Farfar, K.E., Prinz, M., Karras, O., D'Souza, J., Vogt, L., Auer, S.: Fair scientific information with the open research knowledge graph. FAIR Connect **1**(1), 19–21 (2023). https://doi.org/10.3233/FC-221513
27. Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., Liu, Y.: Roformer: Enhanced transformer with rotary position embedding. arXiv preprint arXiv:2104.09864 (2021)
28. Taylor, R., Kardas, M., Cucurull, G., Scialom, T., Hartshorn, A., Saravia, E., Poulton, A., Kerkez, V., Stojnic, R.: GALACTICA: A Large Language Model for Science (2022)
29. Wang, S., Sun, X., Li, X., Ouyang, R., Wu, F., Zhang, T., Li, J., Wang, G.: GPT-NER: Named Entity Recognition via Large Language Models (May 2023). https://doi.org/10.48550/arXiv.2304.10428
30. Xie, T., Wan, Y., Huang, W., Zhou, Y., Liu, Y., Linghu, Q., Wang, S., Kit, C., Grazian, C., Zhang, W., Hoex, B.: Large Language Models as Master Key: Unlocking the Secrets of Materials Science with GPT (Apr 2023). https://doi.org/10.48550/arXiv.2304.02213
31. Xu, C., Sun, Q., Zheng, K., Geng, X., Zhao, P., Feng, J., Tao, C., Jiang, D.: Wizardlm: Empowering large language models to follow complex instructions (2023)
32. Yang, J., Jin, H., Tang, R., Han, X., Feng, Q., Jiang, H., Yin, B., Hu, X.: Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond (Apr 2023). https://doi.org/10.48550/arXiv.2304.13712
33. Ye, D., Lin, Y., Li, P., Sun, M.: Packed Levitated Marker for Entity and Relation Extraction. In: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 4904–4917. Association for Computational Linguistics (May 2022). https://doi.org/10.18653/v1/2022.acl-long.337