SUPPLEMENTARY NOTES

Note 1. Schemas for doping tasks

The English sequence completion schema for the `Doping-English` and `DopingExtra-English` models are shown below. Each paradigm represents a single line in the output completion; aside from the "No information" paradigm, there may be one or more of each of the paradigms in the output sequence (*e.g.,* there may be multiple separate one-to-many host-dopant relationships, a single dopant with no host, multiple results, and multiple modifiers extracted from the same sentence). The placeholders `<HOST>`, `<DOPANT*>`, `<result>`, and `<modifier*>` are used in place of actual string literal entities. The apostrophe "`'`" character was used to more easily delimit entity captures. Output sequences not matching one of these patterns were considered not parsable.

- **One host to one-or-more dopants:** `The host '<HOST>' was doped with '<DOPANT1>', '<DOPANT2>', ... and '<DOPANTN>'.`
- **Single dopant, no host**: `'<DOPANT>' is a dopant.`
- **Single host, no dopant**: `The host '<HOST>' was doped.`
- **Single result**: `'<result>' is a possible doped result formula.`
- **One or more modifiers**: `Modifiers of the doping are '<modifier1>', '<modifier2>'... '<modifierN>'.`
- **No doping-related information:** `There is no doping information.`

These sequences returned by the LLM are decoded programmatically with a simple script that can be found in the code repository accompanying this publication.

Note 2. GPT-3 fine-tuning and inference parameters

All doping models were trained with 7 epochs. Intermediate models shown in the learning curve in the main text were trained with a number of epochs depending on the number of training samples t: 2 epochs for $2^0 \leq t < 2^6$, 4 epochs for $2^6 \leq t \leq 2^7$, and 7 epochs for $t = 2^8$. The remaining models were trained with a batch size of 1 for 4 epochs with start sequence `"\n\n\n###\n\n\n"` and stop sequence `"\n\n\nEND\n\n\n"`. All doping models used GPT-3 inference parameters of 512 max tokens, 0 temperature, `"\n\n\n###\n\n\n"` start token, and `"\n\n\nEND\n\n\n` end token. All other models used 1024 max tokens with remaining parameters identical to the doping models. All models were trained using the python version of OpenAI's API version 0.27.7.

Note 3. Llama-2 fine-tuning and inference parameters

Llama-2 [1] fine-tunes were performed using a modified version of the Facebook Llama-2 recipes repository as of August 11, 2023 (`commit 03faba6`); the modified repository can be found at `https://github.com/lbnlp/nerre-llama`. Llama-2 fine-tunes were performed using the 70 billion parameter version of Llama-2 (`llama-2-70b-hf`) with quantization enabled (8 bit precision) with the following training configuration parameters:

```
batch_size_training: 1
num_epochs: 4 or 7
num_workers_dataloader: 1
lr: 1e-4
weight_decay: 0.0
gamma: 0.85
seed: 42
use_fp16: False
mixed_precision: True
micro_batch_size: 1
peft_method: str = "lora"
quantization: True
```

The number of epochs was set to 7 for doping tasks and 4 for the MOF/general tasks.

Llama-2 fine tunes use parameter efficient fine-tuning (PEFT) using low rank adaptation (LoRa) [2] with the following parameter set:

```
r: 8
lora_alpha: 32
lora_dropout: 0.05
inference_mode: False
```

Decoding was performed without sampling using greedy decoding to be consistent with GPT-3 decoding setting of temperature=0, with max tokens=512 for doping task and 1024 for general and MOF task. More details on the fine tuning and inference parameters are available in the modified repository. All fine-tuning and inference was performed on a single A100 (Ampere) tensor core GPU with 80GB VRAM.

Note 4. seq2rel parameters

Seq2rel models were trained using 267 doping sentences that had dopant-basemat links, with 4 different training:validation splits (90:10, 80:20, 70:30, 95:5) and the model with the highest validation micro-F1 score was selected. Training configurations for seq2rel used parameters for training gene-disease association (GDA) used in Giorgi et al.[3], while modifying the entity tokens to `"@DOPANT@","@BASEMAT@"` and relation token to `"@DBR@"`. To be specific:

```
model_name = microsoft/BiomedNLP-PubMedBERT-
```

```
    base-uncased-abstract-fulltext
max_length=512
max_steps=96
num_epochs=30
batch_size=1
grac_acc_steps=1
decoder_lr=5e-4
encoder_lr=2e-5
encoder_wd=0.01
reinit_layers=1
weight_dropout=0.5
beam_size=4
length_penalty=0.8
```

Note 5. MatBERT-NER + Proximity method

The MatBERT-NER doping model[4], a BERT-based model fine-tuned on 455 abstracts manually annotated by human annotators, produces labels for three entities: `DOPANT` (identical definition to "dopant" entity in the main text), `BASEMAT` ("base material", identical definition to "host" entities in the main text), and `DOPMODQ` (similar definition to "modifier" entities in the main text). We discard the `DOPMODQ` entity, as it differs definitionally from the "modifiers" entity in this work.

Using the labels provided by the MatBERT model for `DOPANT` and `BASEMAT`, we create links only between tagged host and and dopant entities if they occur in the same sentence. This is done with an "all-to-all" rule, meaning all hosts are connected with all dopants *within the proximity of a single sentence*. The resulting set of entities and links between entities can then be evaluated in identical fashion (via word-basis metrics) to the LLM-NERRE models explained in the main text.

Note 6. Class support for materials extraction tasks

Supplementary Table 1: Class support for doping tasks among 77 test (gold) set sentences. "Entites" denotes the number of multi-word entities while "words" denotes the number of total words.

Class/Relation	Support (entities)	Support (words)
`host`	60	111
`dopant`	76	110
`host` → `dopant` (link)	72	114
`results`	12	103
`modifiers`	7	77

Supplementary Table 2: Class support for the general materials joint named entity recognition and relation extraction (NERRE) task from 320 total test set abstracts, including the core entities and the subset of potential links shown in results.

Class/Relation	Support (entities)	Support (words)
`acronym`	41	42
`applications`	305	513
`name`	175	269
`formula`	353	480
`structure/phase`	276	421
`description`	233	314
`formula` → `acronym` (link)	8	8
`formula` → `application` (link)	341	577
`formula` → `name` (link)	60	108
`formula` → `structure/phase` (link)	296	485
`formula` → `description` (link)	202	281

Supplementary Table 3: Class support for the metal organic framework (MOF) task from 255 total test set abstracts including the core entities and the subset of potential links shown in results.

Class/Relation	Support (entities)	Support (words)
`application`	559	1089
`guest_species`	105	110
`description`	143	451
`name`	315	346
`mof_formula`	67	238
`name` → `application` (link)	637	1317
`name` → `guest_species` (link)	152	154
`name` → `description` (link)	87	291
`name` → `mof_formula` (link)	19	26

Note 7. Metrics for Sequence Reconstruction

Exact Match Accuracy. The exact sequence match accuracy is defined based on the an exact match between a predicted completion sequence $\hat{c}_i$ and true completion sequence c_i, averaged over n all samples to be evaluated:

$$\text{Exact match accuracy} = \frac{\sum_i^n \delta_{\hat{c}_i,c_i}}{n} \tag{1}$$

Where δ is the Kronecker delta:

$$\delta_{\hat{c}_i,c_i} = \left\{ \begin{array}{ll} 0, & \text{if } \hat{c}_i \neq c_i \\ 1, & \text{if } \hat{c}_i = c_i \end{array} \right\} \tag{2}$$

Thus, the exact match accuracy is a stringent metric, as any character addition (*e.g.,* addition of extra whitespace) or missing character will not result in an exact match. Equivalent permutations in the orders of entities are considered equivalent; if the models predict verbatim correct JSON for `material 1` and `material 2` but they are reported in the reverse order from the annotation (*e.g.,* `material 1, material 2` vs. `material 2, material 1`), the match is considered exact. This condition ensures JSON documents containing identical information in different orders are considered equivalent. Exact match sequence accuracy is a lower bound on the information capture of the models.

Jaro-Winkler Similarity. For a more granular analysis, we use the Jaro-Winkler similarity, Φ, as a string comparator metric. As in the exact sequence match, we average over all $i = 1, 2, 3, ..., n$ evaluation sequences where the predicted string completion is labelled $\hat{c}_i$ and the corresponding true string completion is c_i. With weights of the first string, second string, and transposition all set equal, the similarity Φ_i between a predicted completion string $\hat{c}_i$ and a true completion string c_i is defined as:

$$\Phi_i = \begin{cases} 0, & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|\hat{c}_i|} + \frac{m}{|c_i|} + \frac{m-t}{m}\right), & \text{otherwise} \end{cases} \quad (3)$$

Where m is the number of matching characters between $\hat{c}_i$ and c_i, t is the number of transpositions, and $|\hat{c}_i|$ and $|c_i|$ are the lengths of the predicted and true completions, respectively. The final average Jaro-Winkler similarity $\bar{\Phi}$ is the arithmetic mean averaged over n samples, $\bar{\Phi} = \sum_i^n \Phi_i / n$.

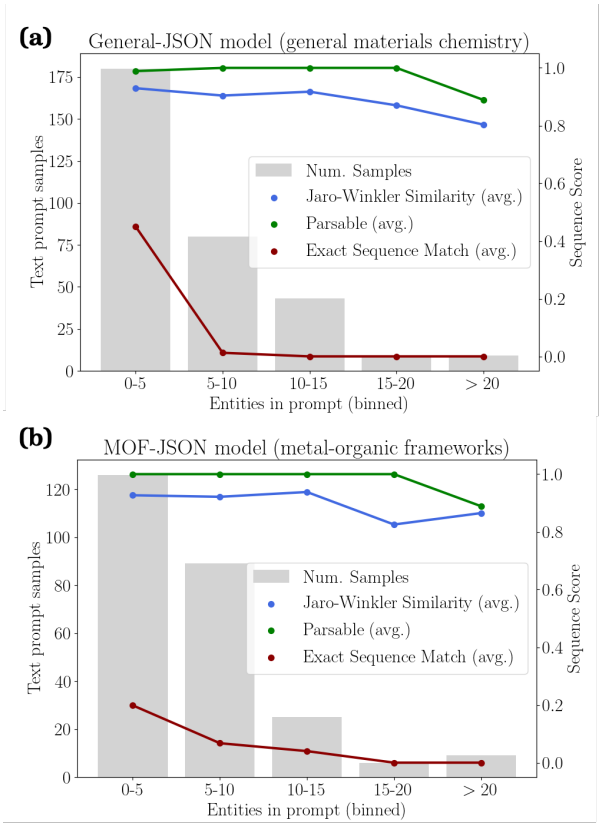
Parsability. As a final sequence reconstruction metric, we report the average percentage of samples which can be parsed from string literal into object form. For the `*-JSON` models, this indicates that the output sequence is well-formatted JSON; for `*-ENG` models, this indicates the output sequence adheres to the natural-language like schema on which the model was trained. In either case, the parsability of the output sequences simply indicates whether the sequence can be easily transformed into a relational object. If a sequence can be parsed using the same function to encode training samples, we return a parsability of one; otherwise, parsability is zero. We average the parsability over all n samples of the evaluation dataset to calculate a final average parsability percentage.

Supplementary Discussion

Discussion 1. Sequence-level results

Supplementary Table 4 shows results for sequence-level matching. We find a wide variability between models for reconstructing test set output sequences exactly, with the simpler doping models containing exact matches on 58.4 – 71.4% of test sequences. The more complex tasks, `MOF-JSON` and `General-JSON`, have exact match accuracies ranging from 12.2 to 30.6% of test sequences. Exact matches are inherently less probable with longer output sequences, as even a single error in a very long output sequence results in an exact match failure. This trend is easier to see with the test samples' sequence reconstruction scores binned into groups based on the number of entities each test set true sample contains. As shown in Supplementary Figure 1 for `MOF-JSON` and `General-JSON`, exact match sequence reconstruction is the highest for the simplest entries (*i.e.,* those with the lowest number of entities).

All models have average Jaro-Winkler similarities $\geq$ 91.2% and parsabilities $\geq$ 98.7%. In contrast to the exact match score, Jaro-Winkler similarity and parsability do not degrade as rapidly with increasing complexity (for which we use true number of entities as a proxy) of the prompt. This is encouraging, as the model is able to retain consistent, parsable formatting even in very long output sequences with many (20+) interrelated entities. All GPT-3 doping models could not produce a parsable response for a single example containing one host material and eight connected dopants; this is likely due to the doping model's inference parameter token limit. If the token limit were increased, it is likely all LLMs could produce parsable output for all texts.



Supplementary Figure 1: Sequence reconstruction metrics segmented by number of entities in the prompt for **(a)** The `General-JSON` model and **(b)** the `MOF-JSON` model. The x axis defines the bins by the number of true entities in the test-set sample; as we progress from left to right, the bins represent more complex samples. The left y axis (corresponding to the grey bars) shows the number of samples in each bin. The right y axis (corresponding to the colored lines centered on each bin) shows the average scores of each bin by the metrics of exact string matching (red), Jaro-Winkler similarity (blue), and parsability (green). While exact string matches are very unlikely responses for the models for highly complex samples (20+ entities), the models are able to retain high sequence similarity and parsability. Source data are provided as a Source Data file.

Supplementary Table 4: Sequence-level error metrics for completions for all tasks, evaluated and averaged over the test set or cross-validation for each task. Completions are only considered correct in the exact match if the full output sequence c is recovered exactly. Average Jaro-Winkler similarities are also shown to measure string similarity continuously on $[0, 1]$, where 1 indicates a perfect match and 0 indicates no match. Parsability, the ability to format c in the same manner as the training schema, is recorded for each sample as 0 (not parsable) or 1 (parsable). The average exact correctness, Jaro-Winkler similarity, and parsability are converted to the range $0 - 100\%$. Best scores for each of the tasks according to each metric are shown in **bold**.

Task	Base Model	Schema	Exact Match Sequence Accuracy (%)	Avg. Jaro-Winkler Similarity (%)	Parsable and Decodable (%)
Doping	GPT-3 davinci-175B	`Doping-English`	59.7	94.3	98.7
	GPT-3 davinci-175B	`Doping-JSON`	59.7	94.5	98.7
	GPT-3 davinci-175B	`DopingExtra-English`	58.4	93.9	98.7
	Llama-2-70B-8bit	`Doping-English`	64.9	94.6	**100**
	Llama-2-70B-8bit	`Doping-JSON`	**71.4**	**97.5**	**100**
	Llama-2-70B-8bit	`DopingExtra-English`	55.8	91.2	**100**
General materials	GPT-3 davinci-175B	`General-JSON`	**30.6**	**94.2**	**100**
	Llama-2-70B-8bit	`General-JSON`	28.3	91.7	**100**
MOFs	GPT-3 davinci-175B	`MOF-JSON`	**15.7**	**94.7**	**100**
	Llama-2-70B-8bit	`MOF-JSON`	12.2	94.2	**100**

Discussion 2. Named entity recognition

We show in Supplementary Table 5 the named entity recognition (NER) scores for recall, precision, and F_1 score for each of the models. These NER scores do not reflect the ability of models to link entities together. The highest scores per common entity category (F_1, precision, recall) are shown in bold, as all tasks are evaluated with multiple models. Entities are evaluated on an exact per-word match basis rather than the basis of exact matches between entire multi-word entities, as it is unclear even to annotators exactly where to denote the end of some complex multi-word entities (e.g., "ZnO nanoparticle crystals" vs. "ZnO nanoparticle" vs. "ZnO"). However, since compositions are the core part of the desired data for all tasks, all words of any formula entity (and hence all its links) are marked incorrect if the entire composition is not captured exactly. Support for each class may be found in Supplementary Section Note 6.

Doping task. In the doping task, we observe roughly comparable performance between the Llama-2 and GPT-3 models across all schemas. In particular, the GPT-3 model fine-tuned using the `Doping-ExtraEnglish` schema has the highest host recall (0.921, margin of 1%) and F_1 (0.901, margin of 1%), while the Llama-2-`Doping-JSON` model has the highest dopant recall (0.892, margin of 4%), dopant precision (0.872, margin of 2%), dopant F_1 (0.882, margin of 4%). Overall, all GPT-3 and Llama-2 models across all schemas are roughly similar, with the Llama-2 `Doping-JSON` model arguably performing the best across entity categories. Broadly, the Llama-2 models tend to perform better at NER tasks when done in JSON schema while GPT-3 models tend to perform better when using English schemas. If we include the `results` and `modifiers` categories, the GPT-3-`DopingExtra-English` performs on average the best across entity categories. This is a notable result, as the `DopingExtra-English` models must additionally extract `results` and `modifiers` entities in the output sequence. We speculate that this discrepancy is due to a combination of model variability and extra specification of non-dopant chemical species within the `DopingExtra-English` model's training set. For example, when the `DopingExtra-English` model is trained with "high-doping" as a labelled `modifier`, it may learn "high" is not a valid dopant entry. This may be useful in the design of more complex completion schema, as `DopingExtra-English` is an example of a *more* complex completion schema extracting all of its entities more accurately.

General and MOF tasks. Both of the more complex General and MOF entity recognition tasks are dominated by the GPT-3 models. Only in the categories of formula recall, MOF name recall, and MOF name F_1 do the Llama-2 models have the highest scores (0.681, 0.779, 0.742 with margins of 1%, 12%, and 2%, respectively).

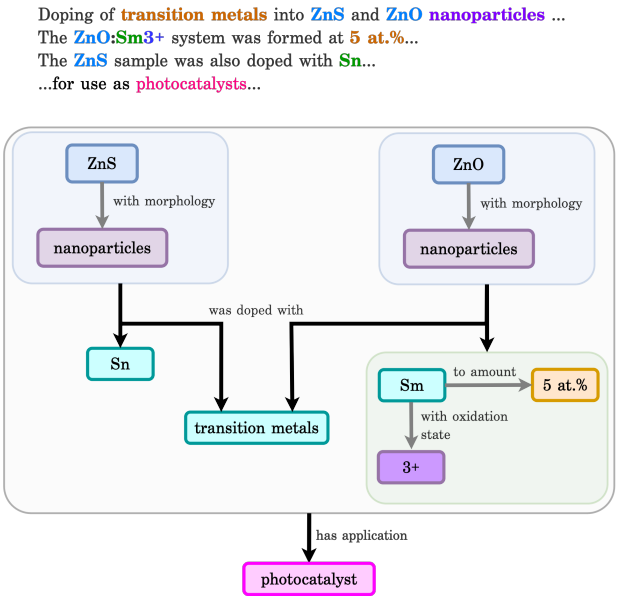
Supplementary Table 5: Exact match (E.M.) named entity recognition scores for the three materials engineering tasks. Scores are computed on the exact-match word basis as described in the main text *Methods*. Highest scores according to each scoring metric (precision, recall, F_1) are shown in **bold** for each individual entity. Results for the General Materials and MOF tasks are shown as an averages from over five-fold cross validation. For these two tasks, the cross-validation averaging means the average F_1 will not necessarily equal the harmonic mean of the average precision and average recall, since F_1 scores are computed per fold and then averaged.

Task	Model	Schema	Entity	E.M. Recall	E.M. Precision	E.M. F_1
Doping	GPT-3 davinci-175B	Doping-English	host	0.899	0.889	0.893
			dopant	0.774	0.844	0.807
		Doping-JSON	host	0.888	0.888	0.888
			dopant	0.714	0.779	0.745
		DopingExtra-English	host	**0.921**	0.881	**0.901**
			dopant	0.845	0.855	0.850
			results	**0.556**	0.909	**0.689**
			modifiers	**0.429**	0.136	0.207
	Llama-2-70B-8bit	Doping-English	host	0.838	**0.932**	0.882
			dopant	0.857	0.837	0.847
		Doping-JSON	host	0.910	0.871	0.890
			dopant	**0.892**	**0.872**	**0.882**
		DopingExtra-English	host	0.899	0.816	0.856
			dopant	0.809	0.701	0.751
			results	0.389	**1.00**	0.560
			modifiers	0.428	**0.167**	**0.240**
	MatBERT-Doping		host	0.551	0.620	0.583
			dopant	0.500	0.583	0.519
	Seq2rel-Doping		host	0.843	0.431	0.570
			dopant	0.652	0.542	0.581
General Materials	GPT-3 davinci 175B	General-JSON	acronym	**0.557**	**0.699**	**0.613**
			applications	**0.696**	**0.731**	**0.712**
			name	**0.585**	**0.777**	**0.666**
			formula	0.672	**0.733**	**0.696**
			structure/phase	**0.581**	**0.676**	**0.622**
			description	**0.484**	**0.510**	**0.495**
	Llama-2-70B-8bit	General-JSON	acronym	0.516	0.687	0.512
			applications	0.677	0.722	0.696
			name	0.537	0.619	0.544
			formula	**0.681**	0.698	0.685
			structure/phase	0.563	0.654	0.600
			description	0.480	0.488	0.476
MOFs	GPT-3 davinci 175B	MOF-JSON	mof_name	0.679	**0.795**	0.728
			mof_formula	**0.730**	**0.816**	**0.767**
			applications	**0.734**	**0.760**	**0.746**
			guest_species	**0.675**	**0.796**	**0.696**
			mof_description	**0.525**	**0.548**	**0.519**
	Llama-2-70B-8bit	MOF-JSON	mof_name	**0.779**	0.711	**0.742**
			mof_formula	0.670	0.811	0.732
			applications	0.696	0.683	0.689
			guest_species	0.552	0.722	0.605
			mof_description	0.420	0.502	0.449

Discussion 3. Doping task graph example

In Supplementary Figure 2 an example end result from a LLM-NERRE model is shown in graph format. We aim here to recognize not just relationships between individual entities, but hierarchical relationships with relationship types which need not be explicitly and comprehensively enumerated beforehand. The LLM-NERRE models presented in the

main text provide a step towards this kind of final product.



Supplementary Figure 2: An example complex graph resolved from the outputs of a LLM-NERRE model. The distinction of this graph from typical entity relationship graphs is the hierarchical format. This hierarchical graph, in contrast to a flat graph, may denote that ZnO *nanoparticles* - as opposed to bulk wurtzite ZnO - were doped by representing an entity with its own subgraph. Similarly, the samarium dopant is represented as a subgraph specifying its oxidation state and amount. Finally, all dopant relationships for ZnO and ZnS are linked to an application "photocatalyst". This hierarchical graph specifies a much more precise series of relationships extracted from text; for example, *"ZnO (as nanoparticles) was doped with Sm (having an oxidation state of +3, and to an amount of 5 atomic percent) resulting in a photocatalyst."* is more precise than a flat-graph relationship such as *"ZnO is a photocatalyst"*.

Discussion 4. Manual scoring examples

Here we provide two examples of manual scoring according to a domain expert using outputs directly from GPT-3 fine-tuned on the general materials information extraction task. False positive errors are shown in the model annotations in orange; false negatives (missed by the model) are shown in the human annotations in red. True positives are shown in the model outputs in blue. Examples are explained below the model outputs, and certain entities of interest (used in explanation) are shown in teal. We show two examples here. Example 1 is simple, while Example 2 is more complex. Note unicode characters are shown in responses prefixed by "\u".

Example 1: *"Self-healing thermoplastic elastomer brush copolymers having a glassy polymethylmethacrylate backbone and rubbery polyacrylate-amide brushes. We synthesized a series of brush copolymers having glassy polymethylmethacrylate (PMMA) backbone and flexible polyacrylate-amide (PA-amide) brushes that exhibit thermoplastic elastomer properties. Importantly, the dynamic hydrogen bonds in the soft PA-amide matrix enables the material to self-heal after mechanical damage at room temperature without the need of any external stimulus."*

```
Human Annotation
[{
    "acronym": "PMMA",
    "applications": ["thermoplastic elastomer"],
    "description": [""],
    "formula": "",
    "name": "polymethylmethacrylate",
    "structure_or_phase": [""]
},
{
    "acronym": "PA-amide",
    "applications": [
        "thermoplastic elastomer",
        "self-healing"
    ],
    "description": [""],
    "formula": "",
    "name": "polyacrylate-amide",
    "structure_or_phase": [""]
}]
-----------------------------------
Model output
[{
    "acronym": "PMMA",
    "applications": ["thermoplastic elastomer"],
    "description": ["backbone"],
    "formula": "",
    "name": "polymethylmethacrylate",
    "structure_or_phase": [""]
},
{
    "acronym": "PA-amide",
    "applications": [
        "thermoplastic elastomer",
    ],
    "description": [""],
    "formula": "",
    "name": "polyacrylate-amide",
    "structure_or_phase": [""]
}]
```

Explanation for example 1: The model correctly determines there are two materials here, PMMA and PA-amide. The model misses the critical self-healing aspect of the material, meaning the "self-healing" is marked as a false negative. The description "backbone" is marked as correct because, while it was not explicitly accounted for in the human

annotation, it is a valid description of PMMA given the context of the passage.

Example 2: *"Synthesis and vapor transport growth of ternary group III chalcogenides The CVT method, using iodine as the transporting gas, has been used to prepare well defined, large (up to 3 cm long) single crystals of AB(III)C(VI)2 dichalcogenides where A = Mn, Zn; B = Al, Ga; C = S, Se, Te. The predominant crystal habit is flat planar growth except Zn1.00Ga1.01Te2.13 which grows as long hexagonal base needles. Mn1.01Al1.04S1.84 (transparent yellow) is monoclinic with a = 11.112 Å, b = 10.490 Å, c = 23.224 Å, β = 87.17°, space group P21/m, Dm = 2.79 g/cm3, Dx = 2.84 g/cm3 and Z = 32; Mn1.04Al1.00Se1.77 (dark metallic grey) has the cubic zincblende (zb) structure with a = 6.273 Å, space group Fm3m, Dm = 3,10 g/cm3, Dx = 3.01 g/cm3 and Z = 2; Mn1.00Al1.02Te2.18 (metallic grey) is monoclinic with a = 8.349 Å, b = 8.260 Å,c = 10.808 Å, β = 84.47°, space group P2/m, Dm = 1.63 g/cm3, Dx = 1.61 g/cm3 and Z = 2; Zn1.00Ga1.02Se1.89 (orange red) is cubic (zb) with a = 5.624 Å, space group Fm3m, Dm = 5.46 g/cm3, Dx = 5.33 g/cm3 and Z = 2; Zn1.00Ga1.01Te2.13 (dark metallic grey) is tetragonal (chalcopyrite) with a = 15.2325 åA, c = 9.9287 Å, space group P 421/m, Dx = 1.18 g/cm3 and z = 4."*

Human Annotation


```
[{
    "acronym": "",
    "applications": [""],
    "description": [""],
    "formula": "Mn1.01Al1.04S1.84",
    "name": "",
    "structure_or_phase": [
        "monoclinic",
        "a = 11.112 A \u030a,
            b = 10.490 A \u030a,
            c = 23.224 A \u030a,
            \u03b2 = 87.17\u00b0"
            \u03b2 = 87.17",
        "P21/m"
    ]
},
{
    "acronym": "",
    "applications": [""],
    "description": [""],
    "formula": "Mn1.04Al1.00Se1.77",
    "name": "",
     "structure_or_phase": [
        "cubic",
        "zincblende",
        "a = 6.273 A \u030a",
        "Fm3m"
    ]
},
{
    "acronym": "",
    "applications": [""],
    "description": [""],
    "formula": "Mn1.00Al1.02Te2.18",
    "name": "",
    "structure_or_phase": [
        "monoclinic",
        "a = 8.349 A \u030a,
            b = 8.260 A \u030a,
            c = 10.808 A \u030a,
            \u03b2 = 84.47\u00b0",
        "P2/m"
    ]
},
{
    "acronym": "",
    "applications": [""],
    "description": [""],
    "formula": "Zn1.00Ga1.02Se1.89",
    "name": "",
    "structure_or_phase": [
        "cubic",
        "zincblende",
        "a = 6.273 A \u030a",
        "Fm3m"
    ]
},
{
    "acronym": "",
    "applications": [""],
    "description": [""],
    "formula": "Zn1.00Ga1.01Te2.13",
    "name": "",
    "structure_or_phase": [
        "tetragonal",
        "chalcopyrite",
        "a = 15.2325 a \u030aA,
            c = 9.9287 A \u030a
            P 421/m"
    ]
}]
```

Model output


```
[{
    "acronym": "",
    "applications": [""],
    "description": [""],
    "formula": "Mn1.01Al1.04S1.84",
    "name": "",
```

```
    "structure_or_phase": [
        "monoclinic",
        "P21/m",
        "a = 11.112 A \u030a,
            b = 10.490 A \u030a,
            c = 23.224 A \u030a,
            \u03b7 = 87.17\u00b0"
        ]
},
{
    "acronym": "",
    "applications": [""],
    "description": [""],
    "formula": "Mn1.04Al1.00Se1.77",
    "name": "",
    "structure_or_phase": [
      "cubic zincblende",
      "Fm3m",
      "a = 6.273 A \u030a"
},
{
    "acronym": "",
    "applications": [""],
    "description": [""],
    "formula": "Mn1.00Al1.02Te2.18",
    "name": "",
    structure_or_phase": [
      "monoclinic",
      "P2/m",
      "a = 8.349 A \u030a"
    ]
},
{
    "acronym": "",
    "applications": [""],
    "description": [""],
    "formula": "Zn1.00Ga1.02Se1.89",
    "name": "",
    "structure_or_phase": [
        "cubic",
        "Fm3m",
        "a = 5.624 A \u030a"
    ]
},
{
    "acronym": "",
    "applications": [""],
    "description": [""],
    "formula": "Zn1.00Ga1.01Te2.13",
    "name": "",
    "structure_or_phase": [
        "tetragonal",
        "chalcopyrite",
            "a = 15.2325 a \u030aA"
    ]
}]
```


Explanation for example 2: In this more complicated example, there are five materials to be extracted, each with several structural descriptions. The model is able to resolve all five formulae correctly, but struggles on a few of the structures and lattice parameters. In particular, the model gives incomplete entities for the lattice parameters of $Mn_{1.00}Al_{1.02}Te_{2.18}$ and $Zn_{1.00}Ga_{1.01}Se_{2.13}$; the entire human annotation for these entities provides the lattice parameters a, b, c, and β, while the model annotation provides only a, resulting in one false positive entity and one false negative entry for each of these compositions. Among the other structural entities, the model resolves all spacegroups and crystal prototypes correctly except for $Zn_{1.00}Ga_{1.02}Se_{1.89}$ where "cubic" and "Fm3m" are extracted, but "zincblende" is missed by the model. Interestingly, while an error in the human annotation for $Zn_{1.00}Ga_{1.02}Se_{1.89}$ reports the wrong lattice parameter ($a = 6.273$Å), the model extracts the correct lattice parameter ($a = 5.624$Å) as it appears in the passage; this is counted as a true positive as the model's extraction is correct. For $Mn_{1.00}Al_{1.02}Se_{1.77}$, the entity "cubic zincblende" is marked as correct because the human annotation's separate "cubic" and "zincblende" entities are split arbitrarily, and the model's output is equivalent.

Discussion 5. Ambiguity of annotations

In practice, annotation can be a complex task even for trained researchers. We have found qualitatively that the performance of the LLM-NERRE models is limited by how consistently and comprehensively the desired schema and entities can be defined. As opposed to canonical NLP examples, where distinctions between entities such as "person" and "place" are relatively clear, scientific texts often contain entities which could plausibly be considered as one or more kinds of entity depending on definition. Reducing ambiguity in the annotation schema is therefore a source of potential improvement for LLM-NERRE methods. We examine several tasks below in regards to ambiguity.

Doping task ambiguity. Definitional ambiguity is particularly apparent in cases where long-range dopant-host relationships may be referenced with no clear "correct" answer to either the annotator or model. However, most sentence-level dopant-host relationships are very clear to annotators. As mentioned in the main text, this is a primary motivator for using a sentence-level annotation scheme rather than an abstract-level annotation scheme. However, even within a sentence, there is definitional ambiguity as to what exactly defines a dopant and host. For example, in a single-doped host crystal (*e.g.,* MgO-doped $LiNbO_3$) with a second dopant of interest (*e.g.,* Cr_3^+), the entities may either be considered a

host doped by one dopant (DOPANT Cr_3^+ → HOST MgO-doped $LiNbO_3$) or a pure crystal codoped with two species (DOPANT Cr_3^+ and DOPANT MgO → HOST LiNbO3) depending on the context of the investigation. Similarly, dopants may not be referenced as such when added in large amounts (*e.g.,* to a ceramic glass).

General task schema ambiguity. In the General-JSON schema, entity class definitions are not entirely consistent throughout the training set. Class definitions drifted somewhat as annotators found more edge cases during the annotation process, and it was still not possible to completely capture all relevant information contained in a passage with this schema. Keeping these details in mind, annotation was done on a "best effort" basis, which seems to still be sufficient to train models capable of complex information extraction tasks. Additionally, this schema is not currently optimized for composite materials (*e.g.,* "F-TiO2/NO3-layered double hydroxide composite"). We annotated examples such that components of the composite were sometimes listed as separate materials and sometimes listed as a single material. While this partially contributes to lower string-match F_1 scores, we find the inconsistencies in the annotations does not significantly impair the model's ability to deal with composites.

Discussion 6. Zero-shot performance

A natural question arising from the LLM-NERRE method is whether similar results can be achieved with zero-shot or prompt engineering approaches in place of fine-tuning. Zero-shot inference in place of fine-tuning would enable researchers to use LLM-NERRE with much less technical investment, since they could use online APIs (such as ChatGPT's API) without having to tune their own bespoke models. However, zero- and few- shot LLM tasks typically magnify the errors found using fine-tuning, even for relatively straightforward examples in chemistry and materials science. In general Q/A tasks, the GPT-4 Technical Report [5] finds zero-shot versions of GPT-4 and GPT-3 have roughly 50% the performance of their RLHF fine-tuned counterparts. In the domain of chemistry, Castro Nascimento and Pimentel [6] report ChatGPT without prompt engineering is capable of converting only 27% of SMILES strings to/from common molecules such as ethylene.

However, it is worth discerning between zero-shot models' ability to act as *knowledge engines* based on their training corpus - *e.g.,* recalling point groups of common molecules such as methane [6] - and their ability to extract information from novel text outside of the training corpus. The findings cited above [5, 6] primarily evaluate the former. It's worthwhile to hypothesize that zero-shot models might be similar to fine-tuned models when evaluated on the latter task.

To briefly test this hypothesis, we evaluate zero-shot GPT-4 on simple examples from the tasks shown in the main text. We illustrate two examples here. We show the test prompt and then with the human (correct) output and model output. Incorrect model outputs are shown in red.

General Materials Information Extraction Example 1

A GPT-4 model retrieved on Nov. 11, 2023 was provided instructions to extract general materials science information from an abstract. The model was also provided prompt-completion (abstract-JSON) pairs of the 10 most relevant abstracts to the test abstract as context. Relevance was determined using GPT-Ada embeddings (text-embedding-ada-002).

Instruction: *"Your current task is to extract data from materials science research paper abstracts. Here is the JSON schema you MUST use. Only output the extracted data in this schema. Do not fill in any information that is not explicitly in the abstract. If you don't know something from the context, just leave that spot blank (don't guess!) Make a list of JSON objects. One for each individual material in the abstract. What is a material? A material is a chemical compound such as 'titania', 'SiO2', or 'graphene'. A material is NOT a device (e.g. 'valve-regulated lead/acid battery'. That would be an application). For composite materials, make one entry for each part of the composite and put the fact it's a composite (and what composite) as one element in the description."*

```
SCHEMA:
[
{
"name": (string) The material's English
    name or other identifier (e.g.
    copper oxide, poly-propylene, BB-1,
    etc). NOT THE FORMULA OR ACRONYM!,
"formula": (string) The chemical formula
     for the material (e.g. BiSTiO3).
    NOT THE NAME OR ACRONYM!,
"acronym": (string) Any acronym used to
    refer to a material (e.g. PEO for
    polyethylene oxide). NOT THE NAME OR
     FORMULA!,
"description": (list of string)
    Descriptions of the material sample.
     For example, if it is a thin film,
    nanoparticles, doped with an
    impurity, etc.
"structure_or_phase": (list of string)
    Any information that indicates the
    materials crystal structure or phase
     (e.g. 'cubic', 'rocksalt', 'HCP', '
    alpha-phase', etc), Also extract
    lattice constants and angles here,
    if present. (DO NOT put things like
    "nanoporous" that describe the
```

```
    physical structure. Those are
    descriptions.)
"applications": (list of string)
    Applications the material is being
    studied/used for (e.g. 'Li-ion
    batteries', 'cathode', 'catalyst',
    etc) If the material is a catalyst,
    include the reaction it catalyzes.
},
...
]
```

Prompt: *'Organic–inorganic single-crystalline perovskites have attracted significant attentions due to their exceptional progress in intrinsic properties' investigation and applications in photovoltaics and optoelectronics. In this study, the large perovskite CH3NH3PbI3 single crystal with the largest length of 80mm was prepared through the method of inverse-temperature crystallization. Meanwhile, the mass production of integrate photodetectors have been fabricated on the single-crystalline wafer and the photoresponse performances were investigated. The results show that the single-crystalline photodetectors have broad spectrum response to 900nm, rapid response speed (40 µs) and excellent stability. These findings are of great importance for future promising perovskite single crystalline for integrated photoelectronic application.'*

Output:

```
Human annotation
[
  {
    "acronym": "",
    "applications": [
      "photovoltaics",
      "optoelectronics",
      "photodetectors",
      "integrated photoelectronic"
    ],
    "name": "",
    "formula": "CH3NH3PbI3",
    "structure_or_phase": [
      "perovskite"
    ],
    "description": [
      "single-crystalline"
    ]
  }
]

Model output
-----------------------------------

 [
  {
    "acronym": "",
    "applications": [
      "photovoltaics",
      "optoelectronics",
      "photodetectors"
    ],
    "name": "CH3NH3PbI3",
    "formula": "",
    "structure_or_phase": [
      "perovskite"
    ],
    "description": [
      "single crystal",
      "photodetectors"
    ]
  }
]
```

Explanation: The model correctly retrieves all applications and provides an acceptable additional application "integrated photoelectronics" (as determined by a human annotator). However, the model transposes the name and formula fields despite being specifically warned against it in the instructions. Additionally, it incorrectly places an application ("photodetectors") into the material description.

General Materials Information Extraction Example 2

Instruction: An identical instruction as the previous example was provided to the model, with a different abstract.
Prompt: *"Electrostatically and electrochemically induced superconducting state realized in electrochemical cells. We here report the result of in situ magnetization measurements of electrochemical cells at low temperatures. Upon applying voltages between the electrodes of the electrochemical cells, we observed shielding diamagnetic signals from several materials, indicating superconducting transitions. The superconducting states can be induced both electrochemically and electrostatically with appropriate combination of counter electrode materials and electrolytes. The present technique may become a powerful method for searching novel superconductors."*
Output:

```
Human annotation
[]

Model output
-----------------------------------
[
  {
    "name": "electrode materials",
    "formula": "",
    "acronym": "",
```

```
    "description": [],
    "structure_or_phase": [],
    "applications":
      ["used in electrochemical cells"]
  },
  {
    "name": "electrolytes",
    "formula": "",
    "acronym": "",
    "description": [],
    "structure_or_phase": [],
    "applications":
      ["used in electrochemical cells"]
  }
]
```


Explanation: We only aim to extract specific materials, hence the empty human response. No specific materials, by name or by formula, are presented in the abstract. Yet, the model returns two materials, "electrode materials" and "electrolytes".

Conclusions on zero-shot learning

The examples shown here are representative of the errors zero-shot LLMs make for scientific relation extraction. These include basic mistakes such as identifying an application as a material description. Zero-shot models often experience these failure modes on extremely straightforward texts despite being warned explicitly against them in the instructions and being provided the context of similar correct prompt-completion pairs. Reasonable variations on the model instructions/prompt yield similar underwhelming results. The LLM and prompt approach we test here are thus unsuitable for scientific NERRE tasks.

However, better LLMs and/or improved prompt engineering will likely improve these results. As White *et al.* [7] and Zheng *et al.* [8] showed, prompt engineering is a powerful lever for tuning the responses of LLMs. More, the rapid advancement of zero-shot LLM performance over the past several years is encouraging for future models to perform zero-shot NERRE.

SUPPLEMENTARY REFERENCES

[1] Touvron, H. *et al.* Llama 2: Open foundation and fine-tuned chat models (2023). URL https://arxiv.org/abs/2307.09288.

[2] Hu, E. J. *et al.* Lora: Low-rank adaptation of large language models (2021). URL https://arxiv.org/abs/2106.09685.

[3] Giorgi, J., Bader, G. & Wang, B. A sequence-to-sequence approach for document-level relation extraction. In *Proceedings of the 21st Workshop on Biomedical Language Processing*, 10–25 (Association for Computational Linguistics, Dublin, Ireland, 2022). URL https://aclanthology.org/2022.bionlp-1.2.

[4] Trewartha, A. *et al.* Quantifying the advantage of domain-specific pre-training on named entity recognition tasks in materials science. *Patterns* **3**, 100488 (2022). URL https://doi.org/10.1016/j.patter.2022.100488.

[5] OpenAI. Gpt-4 technical report (2023). arXiv:2303.08774.

[6] Castro Nascimento, C. M. & Pimentel, A. S. Do large language models understand chemistry? a conversation with chatgpt. *Journal of Chemical Information and Modeling* **63**, 1649–1655 (2023). URL http://dx.doi.org/10.1021/acs.jcim.3c00285.

[7] White, A. D. *et al.* Assessment of chemistry knowledge in large language models that generate code. *Digital Discovery* **2**, 368–376 (2023). URL http://dx.doi.org/10.1039/D2DD00087C.

[8] Zheng, Z., Zhang, O., Borgs, C., Chayes, J. T. & Yaghi, O. M. ChatGPT chemistry assistant for text mining and the prediction of MOF synthesis. *Journal of the American Chemical Society* **145**, 18048–18062 (2023). URL https://doi.org/10.1021/jacs.3c05819.