Chem Soc Rev

TUTORIAL REVIEW

View Article Online
View Journal | View Issue

Check for updates

Cite this: *Chem. Soc. Rev.*, 2025, **54**, 1125

Received 11th October 2024

DOI: 10.1039/d4cs00913d

rsc.li/chem-soc-rev

From text to insight: large language models for chemical data extraction

Mara Schilling-Wilhelmi, †[a] Martiño Ríos-García, †[ab] Sherjeel Shabih, [c] María Victoria Gil, [b] Santiago Miret, [d] Christoph T. Koch, [c] José A. Márquez [c] and Kevin Maik Jablonka *[aef]

The vast majority of chemical knowledge exists in unstructured natural language, yet structured data is crucial for innovative and systematic materials design. Traditionally, the field has relied on manual curation and partial automation for data extraction for specific use cases. The advent of large language models (LLMs) represents a significant shift, potentially enabling non-experts to extract structured, actionable data from unstructured text efficiently. While applying LLMs to chemical and materials science data extraction presents unique challenges, domain knowledge offers opportunities to guide and validate LLM outputs. This tutorial review provides a comprehensive overview of LLM-based structured data extraction in chemistry, synthesizing current knowledge and outlining future directions. We address the lack of standardized guidelines and present frameworks for leveraging the synergy between LLMs and chemical expertise. This work serves as a foundational resource for researchers aiming to harness LLMs for data-driven chemical research. The insights presented here could significantly enhance how researchers across chemical disciplines access and utilize scientific information, potentially accelerating the development of novel compounds and materials for critical societal needs.

Key learning points

1. End-to-end workflow for LLM-based chemical data extraction: from data collection to structured output.
2. Advanced techniques to enhance extraction: multimodal approaches and agentic systems.
3. Quality assurance in chemical data extraction: constrained decoding and domain-specific validation.
4. Future frontiers: addressing cross-document analysis, diverse modalities, and emerging challenges in chemical LLMs.

1 Introduction

Materials and molecular design have a long history of using empirical correlations or models to guide the design and synthesis of new compounds—or to inspire new theories. For example, structured data has been used *via* Ashby plots to select materials,[1] or *via* scaling relations to design catalysts.[2] More recently, structured data has been used *via* machine learning models to predict the properties of compounds before they have been synthesized.[3–6] In some cases, models can even directly recommend materials based on desired properties or guide the design of chemical experiments.[7,8] A foundation for all these data uses is that the data must be available in a structured form, *e.g.*, a table with clearly defined columns (often representing specific chemical or physical properties).

While there have been many advances in research data management, only a minuscule fraction of all available research data is available in a usable structured form (see Fig. 1). Most data is still only reported in a compressed and highly processed form *via* text such as scientific articles. Thus, there is a large untapped potential in leveraging this unstructured text data.

The promise of leveraging data "hidden" in scientific articles—such as links between disjoint data published in separate

[a] *Laboratory of Organic and Macromolecular Chemistry (IOMC), Friedrich Schiller University Jena, Humboldtstrasse 10, 07743 Jena, Germany. E-mail: mail@kjablonka.com*
[b] *Institute of Carbon Science and Technology (INCAR), CSIC, Francisco Pintado Fe 26, 33011 Oviedo, Spain*
[c] *Department of Physics and CSMB, Humboldt-Universität zu Berlin, Berlin, Germany*
[d] *Intel Labs, Santa Clara, California, USA*
[e] *Center for Energy and Environmental Chemistry Jena (CEEC Jena), Friedrich Schiller University Jena, Philosophenweg 7a, 07743 Jena, Germany*
[f] *Helmholtz Institute for Polymers in Energy Applications Jena (HIPOLE Jena), Lessingstrasse 12-14, 07743 Jena, Germany*

† These authors contributed equally.

articles ("Swanson links")—has motivated researchers to develop dedicated extraction pipelines.[10,11] These pipelines often relied on rule-based approaches,[12–16] or smaller machine learning (ML) models trained on manually annotated corpora.[17–21] Those approaches, however, face challenges with the diversity of topics and reporting formats in chemistry and materials research as they are hand-tuned for very specific use cases.[22] In 2019, researchers still struggled to effectively utilize data from old PDFs as "currently available solutions still fail[ed] to provide high enough accuracy to reliably extract chemistry"[9] and the development of extraction workflows typically required very deep understanding and investment of multiple months of development time that had to be recommitted for every new use case. Data extraction, thus, presents a "death by 1000 cuts" problem. Automating the extraction for one particular case might not be too difficult, but the sheer scale of possible variations makes the overall problem intractable.[23,24]

With advances in large language models (LLMs) this has dramatically changed because LLMs can solve tasks for which they have not been explicitly trained.[25–28] Thus, LLMs present a powerful and scalable alternative for structured data extraction. Using these LLMs, many data extraction workflows that would have taken weeks or longer to develop can now be bootstrapped into a prototype over the course of a two-day hackathon.[29]

While there has been a growing number of reports showing the use of LLMs for data extraction in chemistry[30–33] and materials science,[21,34–40] there is still no clear framework on how to best build and evaluate such extraction pipelines for materials and molecular data. In many cases, chemical data extraction is more complicated than in other domains.[41] However, chemical expertise and physical laws also offer unique possibilities to constrain or validate the results.

This review aims to leverage the synergy between LLMs and chemistry and provide a practical guide for materials scientists and chemists looking to harness the power of LLMs for

Mara Schilling-Wilhelmi

Mara Schilling-Wilhelmi obtained her Master's degree in Chemistry earlier this year from the Friedrich-Schiller University Jena, Germany. She since has become a PhD student under the supervision of Kevin M. Jablonka. Her research focuses on developing novel benchmarks for comparing state-of-the-art models in the chemistry domain, utilizing Vision-Language Models (VLMs) to extract information from published data, and applying this data to predict chemical reactions.

Martiño Ríos-García

Martiño Ríos-García joined the Friedrich-Schiller University Jena, Germany, in the fall of this year as a PhD student under the supervision of Kevin M. Jablonka. His research focuses on unveiling the chemical capabilities of Large Language Models (LLMs) and developing foundational models for chemical discovery.

Sherjeel Shabih

Sherjeel Shabih has a double degree in Materials and Computer Science and is currently working toward his doctoral degree in physics in the Department of Physics of Humboldt Universität zu Berlin. He is also a developer of NOMAD, the central research data management software platform of the FAIRmat consortium of Germany's National Research Data Infrastructure (NFDI).

María Victoria Gil

María Victoria Gil is a Tenured Scientist at the Institute of Carbon Science and Technology of the Spanish National Research Council (INCAR-CSIC). Her research focuses on the development of sustainable energy technologies based on thermochemical processes and renewable materials, with a focus on hydrogen production and carbon dioxide emission reduction. She is using advanced artificial intelligence models and machine learning tools to optimize the use of experimental data from the chemical engineering field to accelerate innovation in sustainable chemistry and contribute to the advancement of clean energy solutions for a low-carbon future.

structured data extraction. We cover the entire workflow and real-world applications, drawing from the latest research and insights from the field. By bridging the gap between LLM research and practical application in chemistry and materials science, we aim to empower researchers to make the most of these powerful models. To facilitate this, each section will be accompanied by a section in an executable Jupyter Book[42] that elaborates on hands-on examples (**Online Material – matextract.pub**).[43]

2 Overview of the working principles of LLMs

At the most basic level, LLMs work by completing the text they receive as input with one of the (what they compute to be) most likely tokens (Fig. 2). In this context, tokens are the basic units of text on which those models operate. They are usually words but can be suffixes, prefixes, or individual characters. A model knows only a finite number of tokens, which is typically known as the vocabulary. This vocabulary must be determined prior to the training of the model and can limit the performance of models, *e.g.*, *via* suboptimal splitting of chemical formulas or numbers.[44]

2.1 Sampling outputs

The output of the most commonly used models (so-called decoder-only models) is a probability distribution over tokens, *i.e.*, a measure of how likely each token the model knows is as a completion of the input sequence (Fig. 2). While it might seem most intuitive to choose the most likely token (so-called greedy decoding), this often leads to unnatural-sounding text. Thus, in

Santiago Miret

Santiago Miret is an AI Researcher Lead at Intel Labs, specializing in materials discovery and understanding. He manages international academic collaborations across three continents, focusing on AI applications in science. As a key organizer of the AI for Accelerated Materials Discovery (AI4Mat) workshop at NeurIPS 2022, he facilitated interdisciplinary dialogue between materials science and AI experts. His commitment to the AI4Mat community continues with events at NeurIPS 2023, BOKU University Vienna (2024), and NeurIPS 2024.

Christoph T. Koch

Christoph Koch is a professor in the department of physics of Humboldt-Universität zu Berlin and co-spokesperson of the FAIRmat consortium within Germany's National Research Data Infrastructure (NFDI). The research of his group focuses on exploring the atomic and electronic structure of materials using electron microscopes, as well as the development of numerical algorithms, including ML, for retrieving detailed material information from the recorded data. Some of his projects also include the use of LLMs for making software and instrumentation available for online operation via a chat bot.

José A. Márquez

José Márquez is the scientific coordinator of the FAIRmat consortium within Germany's National Research Data Infrastructure (NFDI) at Humboldt-Universität zu Berlin. His research focuses on optoelectronic materials, high-throughput experimental techniques for energy materials, and the integration of data infrastructure into materials science. He leads FAIRmat's scientific efforts to advance NOMAD, a distributed FAIR data management platform for solid-state physics and materials science.

Kevin Maik Jablonka

Kevin Jablonka leads an independent research group at the Helmholtz Institute for Polymers in Energy Applications of the University of Jena and the Helmholtz Center Berlin. Kevin has received numerous awards, such as the Dimitris N. Chorafas Foundation award for outstanding PhD work. Kevin belongs to a new generation of scientists with a broad skill set, combining chemistry, materials science, and artificial intelligence expertise. His research addresses challenges across scales, from atomic-level simulations to pilot plants, pushing the boundaries of AI-accelerated discovery in materials science. He has been at the forefront of applying LLMs to problems in the chemical sciences.

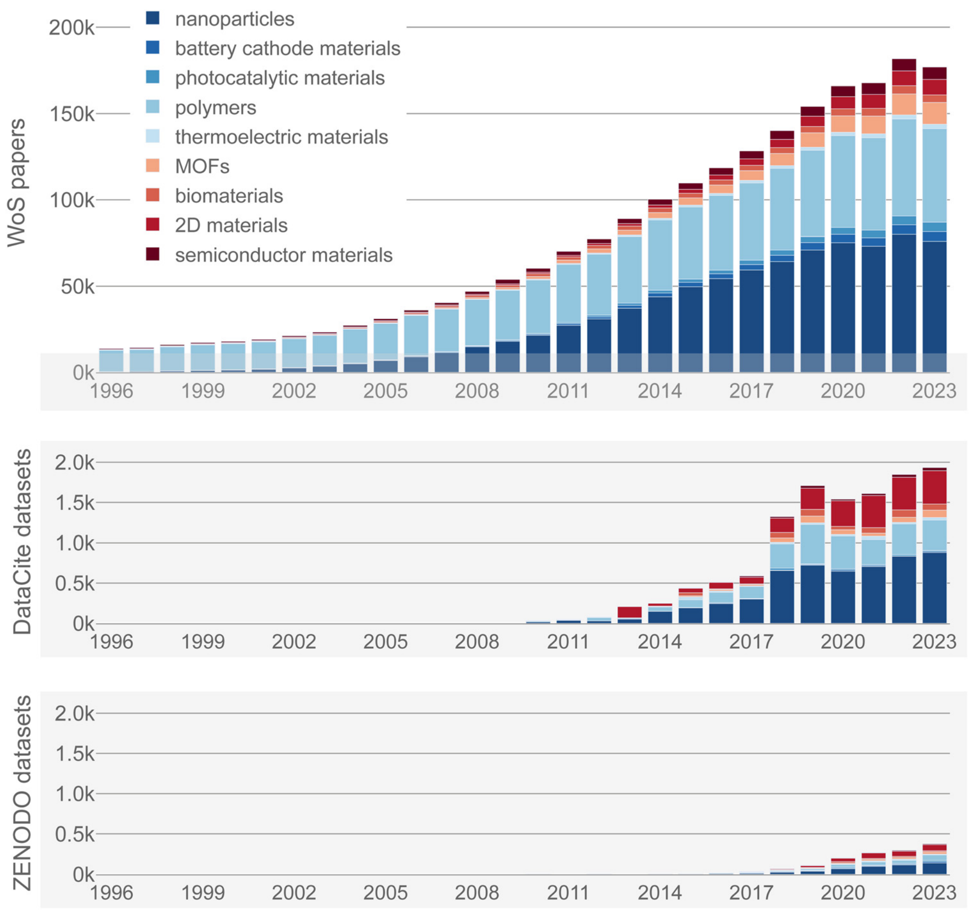


Fig. 1 Number of research papers *vs.* datasets deposited in data repositories in materials science and chemistry per year. The top graph shows the number of publications from 1996 to 2023. The number of records was obtained from the search queries "(nanoparticles)", "(battery AND cathode AND materials)", "(photocatalytic AND materials)", "(polymers)", "(thermoelectric AND materials)", "((metal–organic AND framework) OR MOF)", "(biomaterials)", "((2D AND materials) OR graphene)", and "(semiconductor AND materials)" in the Web of Science Core Collection on July 1, 2024 (search based on title, abstract and indexing, including "Article" and "Data Paper" document types) (categories based on Kononova *et al.*[9]) (see **Online Material – research articles vs. datasets in chemistry and materials science**). The two graphs below show the number of datasets in chemistry and materials science deposited in the Zenodo and DataCite repositories from 1996 to 2023. The number of records was obtained from similar queries by restricting the document type to "dataset". Note the different *y*-axis scale between the top and bottom graphs. While this figure highlights the large difference in the availability of structured datasets compared to papers, we note that a one-to-one comparison of these numbers is not always fair. This is because sometimes multiple papers can be used to create a single dataset, as in the case of curated databases, or *vice versa*, where multiple structured datasets can result from a single paper's work.

some applications, it is common to choose less likely tokens. The frequency of this happening is determined by a model parameter known as temperature. A higher value of temperature means that there are higher probabilities that less likely tokens are generated. For structured data-extraction tasks, working at a temperature value of 0 is typically the best, as this will lead to deterministic outputs with the most relevant information. It is important to keep in mind that most LLMs are so-called autoregressive models.[45–47] This means that they generate the output using their previous output tokens as input. For example, the model in Fig. 2 would generate one token based on the input, and the output token would then be added to that input. Using this longer input, the model can be queried again, producing another token. This process can be repeated a fixed number of times or until a specific end-of-sequence indicator is reached. The autoregressive nature is important because it leads to two major limitations. First, making predictions requires a full pass through the model for each output token, which can be slow. Second, errors multiply, *i.e.*, if a token is generated incorrectly, it cannot be "fixed", and the generation of subsequent tokens is affected by that error, potentially leading the output astray.[48]

2.2 Embeddings

Internally, the models do not work with the discrete tokens. Instead, they operate with high-dimensional vector representations of tokens, so-called embeddings. The power of these embeddings is that they capture the syntactic and semantic relationships between tokens. That is, synonyms or related words and phrases are mapped onto neighboring parts of this high-dimensional space. Internally, LLMs transform input embeddings by letting them interact, *e.g.*, using the attention

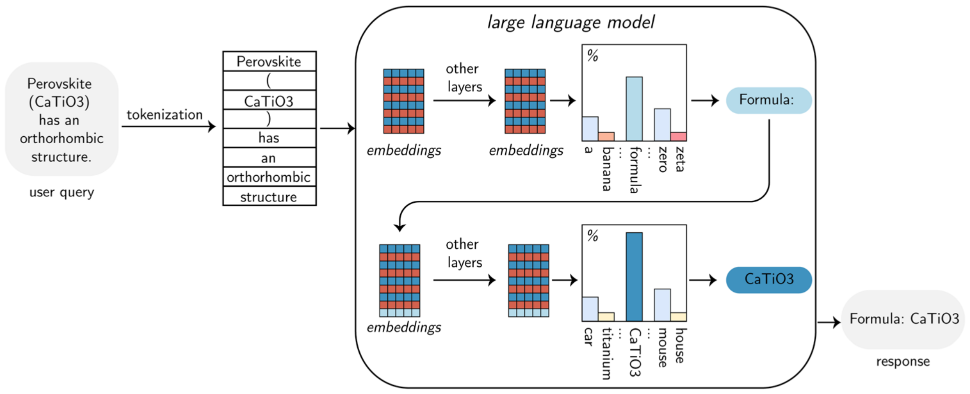


Fig. 2 High-level explanation of the working principle of an LLM. The data flow in the image corresponds to a decoder-only model, *e.g.*, a GPT or Llama model. One token is produced each time, considering all the tokens from the input and all previously produced tokens. The process starts with the tokenizer, which converts the user query into smaller constituent units, the tokens. The tokens are passed into the model, where input embeddings are computed, after which additional operations transform the embeddings. As a result, the model outputs the probabilities over possible subsequent tokens. Depending on the temperature parameter, the most probable token or a less probable one is chosen, and the sampled token is added to the input. By repeating this process, the model generates the response to the query.

mechanism[25] to give access to the entire sequence at each time step. The attention blocks are the core of recent LLMs, allowing them to capture long-range dependencies and making the training of these models highly parallelizable. However, attention does not account for changes in the order of the tokens. To still capture the order of the tokens, LLMs introduce so-called positional encoding that adds a unique signal to each word embedding, which describes the location of each token within the input.

2.3 Training and tuning of LLMs

The capacity of these models to generate coherent text comes from their training, in which they are exposed to massive amounts of natural text mined from the internet. Most recent LLMs are initially trained to predict the next token within a sentence. This process is called pre-training and provides models with general-purpose representations, an understanding of semantics and syntax, and world knowledge.[49] Since those models are trained to complete text, they are typically not ideal for answering questions. They will often attempt to complete input patterns, *e.g.*, by generating more questions. To overcome this issue, they are often specifically tuned to follow desired guidelines and to answer such as humans would do. This is known as instruction-tuning and is typically performed as a supervised training task, where the model is trained on prompts and desired completions. In the context of LLMs, prompts refer to the input sequence containing the user's query and potentially some additional examples (see Section 3.2.1). The most recent models are often also aligned to human preferences using processes such as reinforcement learning from human feedback (RLHF) in which LLMs are tuned based on preference data such as the preferred choice among two completions generated by the model.[50] Importantly, it has been observed that this process makes the output probabilities of the model less calibrated,[51] that is, they no longer correspond to expected error rates.

2.4 LLM-systems

For simplicity, we are going to mostly write LLMs even though most of the time, we are using systems. Systems are a combination of models and additional tools that might include safety filters. When a user calls a model *via* an application programming interface (API), they seldom interact with only the model but also get some inputs and outputs being processed by additional tools.

Additional resources on how LLMs work can be found in the **Online Material – overview of the working principles of LLMs**.

3 Structured data extraction workflow

The structure of this section describes a common workflow of a data extraction project. In practice, parts of the workflow must be optimized iteratively depending on the quality of the extraction result (see Fig. 3). The pipeline typically starts by collecting data (Section 3.1.1), which then must be preprocessed and often chunked before it can be sent to interact with an LLM. The LLM can be used in diverse ways, such as by prompting or fine-tuning it, or in agentic (to use tools) or multimodal (to study other modalities than text) settings. Often, the desired output follows a special structure. In those cases, one can leverage techniques to ensure that the output follows this structure. To be able to optimize the extraction, it is essential to evaluate the extraction performance. This is often not trivial but can be aided by insights from materials science and chemistry.

3.1 Preprocessing

3.1.1 Obtaining data. The first step in the structured data extraction workflow is acquiring the necessary data. Data mining can pose significant challenges due to legal restrictions and practical difficulties in accessing and processing scientific literature. To ensure the legality of data mining, it is important to consider copyright infringements that may occur when

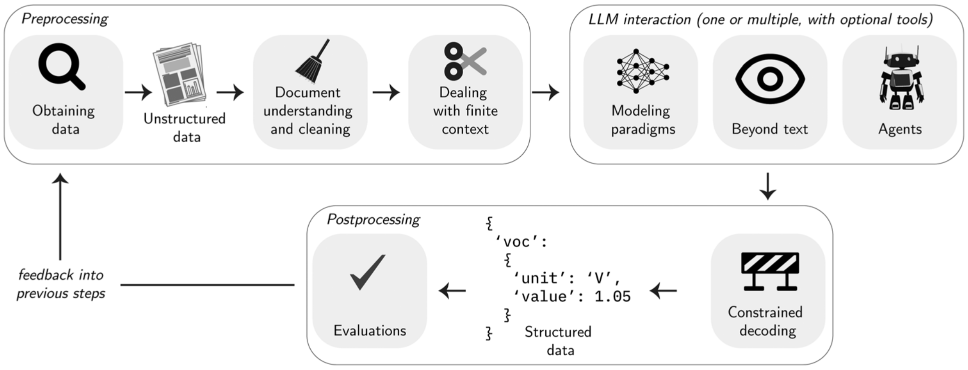

Preprocessing
Obtaining data
Unstructured data
Document understanding and cleaning
Dealing with finite context
LLM interaction (one or multiple, with optional tools)
Modeling paradigms
Beyond text
Agents
Postprocessing
Evaluations
{ 'voc': { 'unit': 'V', 'value': 1.05 } }
Structured data
Constrained decoding
feedback into previous steps

Fig. 3 Data extraction workflow. This figure illustrates the flow of data from left to right through various stages of the extraction process. The evaluation loop includes all steps in the workflow, indicating that if evaluations do not yield satisfactory results, corrections and improvements may be necessary at any stage. It is important to conduct these evaluations using a representative and labeled test set, rather than the entire unstructured data corpus. Once the evaluations demonstrate satisfactory results, the entire corpus of unstructured data can be processed.

copying and adapting the original work. There are two possibilities to avoid copyright infringements: either by getting the owners' permission for usage, or if the data mining actions fall under a copyright exception. Only a few publishers, such as Elsevier, Wiley, and Springer Nature, provide a general copyright license for text and data mining (TDM) use in addition to their usual contracts, and sometimes at an extra cost. However, most publishers lack a general TDM agreement. Thus, researchers (or the libraries that have an agreement with the publishers) must approach the publishers individually. Yet, there is a wide variety of copyright exceptions that may also cover TDM research. In the US, there is a ''fair use'' clause that represents a general exception to copyright for research and educational purposes, which has so far been interpreted by courts to allow at least some TDM applications.[52] In contrast, the UK and Japan have explicit exceptions to copyright for content mining, and EU member states have to at least permit copying for non-commercial research or private studies.[53,54]

Several resources besides the major scientific publishers are available for data mining (Table 1). First, there are repositories of open-access published articles like EuroPMC[55] or preprint servers such as ChemRxiv[56] and arXiv,[57] which contain a wealth of open-access scientific articles about different academic disciplines. Furthermore, data dumps like S2ORC[58] or the Elsevier OA Corpus[59] collect a large variety of open-access scientific publications with mostly Creative Commons Attribution licenses. Most available data sources also provide automated access through an API, simplifying data collection. However,

Table 1 Overview of some data sources relevant for structured data extraction from scientific text, including published articles in open-access archives data dumps. This table gives a non-comprehensive overview of data sources, the license provided by these publishers, the database size, the available data format of the articles, and how these articles can be accessed programmatically

Data source	Description	License	Size	Data format	Automated access
Published articles					
EuroPMC[55]	Corpus of life science literature	License terms of the respective authors or publishers	43.9m abstracts, 9.7m full text articles	XML or PDF	With RESTful API from EuroPMC
arXiv[57]	OA[a] archive for different academic disciplines	License terms of the respective authors or publishers[b]	2.4m full text articles	HTML or PDF	With arXiv API
ChemRxiv[56]	OA[a] archive for chemistry and related areas	License terms of the respective authors or publishers	150k full text articles	PDF	With ChemRxiv public API
USPTO[c]	Federal agency for granting U.S. patents and registering trademarks	Text and drawings are typically not subject to copyright restrictions	Over 50k chemical reactions only from 1976 till 2016	Images and text documents	With USPTO APIs
Data dumps					
S2ORC[58]	Academic articles from many academic disciplines	ODC-BY 1.0[d]	81.1m abstracts, 8.1m full text articles	Machine readable text	With semantic scholar APIs
Elsevier OA CC-BY[a] [59]	Corpus of OA[a] articles from across Elsevier's journals	CC-BY[e]	40k full text articles		With Elsevier's APIs
Open Reaction Database[61,62]	OA[a] corpus for organic reaction data	CC BY-SA[f]	2m chemical reactions		With ORD interface

[a] OA is 'open-access'. [b] For arXiv one has to check the copyright restrictions for each article. The default license does not specify reuse. [c] USPTO is 'The United States Patent and Trademark Office'. [d] ODC-BY 1.0 is 'Open Data Commons Attribution License v1.0'. [e] CC BY is 'Creative Commons Attribution'. [f] CC BY-SA is 'Creative Commons Attribution – ShareAlike'.

most APIs provide a limit for requests per time to protect servers from overload. Thus, API providers might block researchers' IP addresses or even cut off whole institutional access if TDM rules are violated.[53,60]

Tools for data mining. Various tools have been developed to aid in the data mining process. Thus, for the first step in this process, typically the collection of a database of potentially relevant articles for a topic, one could use different techniques like full-text or abstract search, search across different sources, so-called federated search, or search for one specific publisher. The Crossref API[63] or the Scopus API[64] are examples of federated search APIs, which can be used to collect potentially relevant works from various publishers and sources and retrieve their metadata. However, it is important to note that the selection of relevant articles for a specific research focus is still an open question. After finding relevant articles, one could use tools like SciCrawler[65] to collect the full-text articles from different corpuses, although researchers must be aware of current copyright law. Articles from open-access databases can be collected using tools such as pygetpapers.[66] One can find a demonstration of these tools in the **Online Material – obtaining data**.

Importance of structured data. However, simply mining unstructured data is insufficient for creating a data extraction pipeline. To optimize and evaluate the extraction pipeline, it is critical to programmatically evaluate the extraction performance at a later stage. This requires a set of unstructured data paired with the desired output. Tools such as Argilla[67] or Doccano[68] are emerging to aid in the creation of such datasets. Since extraction pipelines need to be optimized, such as by fine-tuning the LLMs, it is necessary to have both a test set and a validation set. The validation set is necessary because the test set should only be used to evaluate the final performance of the pipeline. If the test set is used to guide the optimization of the pipeline, information from the testing stage will influence the optimization process (data leakage), which is a major pitfall in ML.[69]

3.1.2 Curating and cleaning data.

While there are plenty of resources that provide (unstructured) data, this data is often not in a form that can, or should, be used for generative data extraction (see Fig. 4). On the one hand, this is because the data might be only provided as an image, which cannot be used as input for a text-based model. On the other hand, the unedited articles often still contain information (*e.g.*, bibliography, acknowledgments, page numbers) that might not be needed for the extraction tasks. Thus, the input for the model could be compressed and less "distracting" if such parts are removed.

Document parsing and understanding. Before any such curation can occur, the document must be in a form that curation tools and models can work with. In many cases, this involves a step of visual document understanding (VDU), which enables machines to understand documents. This might require not only optical character recognition (OCR), which is focused on extracting machine-readable text from text images, but also the ordering and sequence of text blocks and layout (*i.e.*, the reading order). Further layout understanding is needed because documents such as images or PDFs, in contrast to extensible markup language (XML) or hypertext markup language (HTML), do not have semantic annotations that describe the meaning and context of different parts of the document. In such cases, additional tools must be used to analyze the document's structure. Traditionally, this has often been performed using rule-based systems (as, *e.g.*, in PDFDataExtractor by Zhu and Cole[70]) operating on OCR output (which can be obtained using OCR tools such as Tesseract[71]). However, recently, it has been shown that high performance can be obtained using end-to-end modeling approaches such as Nougat[72] or Marker,[73] which are specialized for converting scientific papers from PDF to Markdown (see **Online Material – document parsing with OCR tools**). In those Markdown files, the meaning of different parts (*e.g.*, sections, tables, equations) is semantically annotated using the Markdown markup. Complicated structures like complex equations and large tables have a high potential for errors.[74] As an alternative to text-based models, multi-modal models can also handle other types of inputs, *e.g.*, images. These approaches are discussed in Section 3.2.4.

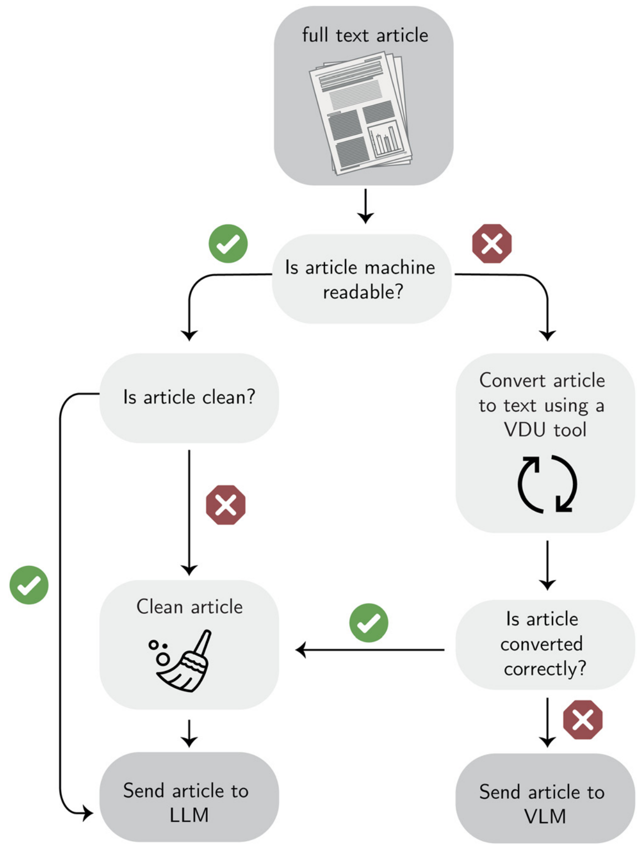

full text article
Is article machine readable?
Is article clean?
Convert article to text using a VDU tool
Clean article
Is article converted correctly?
Send article to LLM
Send article to VLM

Fig. 4 Data preprocessing workflow. The process from the mined articles to machine-readable and cleaned format, which one could send to a LLM. For articles for which the relevant information cannot readily be extracted using conventional visual document understanding (VDU) tools, vision language model (VLM) might be a suitable alternative (see Section 3.2.4).

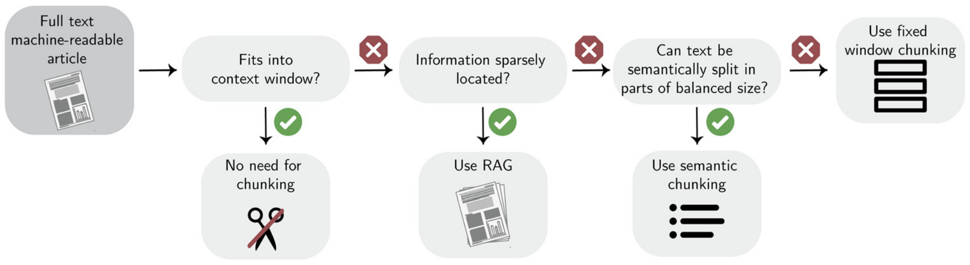

Full text machine-readable article
Fits into context window?
Information sparsely located?
Can text be semantically split in parts of balanced size?
Use fixed window chunking
No need for chunking
Use RAG
Use semantic chunking

Fig. 5 Decision tree to help decide what chunking strategy to use. If the input text is quite short, no chunking is required. In contrast, if the information is spread across a very large corpus, retrieval augmented generation (RAG) can provide cost and efficiency benefits. Chunking is typically applied with RAG. In this case, one can use semantic chunking if it provides chunks that fit into the context window. The most simple option is to chunk text using a fixed window size.

Document cleaning. For many data extraction tasks, it can be useful to preprocess the data, for example, by deleting text that does not contain the relevant information. This can make the system more robust (as noise is removed) and cheaper (as less text has to be processed). Often, sections and structures such as authors, affiliations, acknowledgments, references, page numbers, headers, and footers, as well as unneeded white spaces or linebreaks, can be removed. Some old articles include the end or the beginning of the next article in the text file, which could lead to major errors during the extraction. For chemistry and materials science-related data extraction tasks, for example, one might need only the experimental part to extract raw materials, synthesis, characterization, and products, or one need to include the Supporting Information files which usually contain detailed descriptions of synthetic routes and experimental procedures in organic chemistry.[75] Hence, the content length of the articles could be reduced to the most relevant parts. To do so, practitioners often use regular expression-based pipelines as, for example, in the ChemNLP project.[76] One can find examples for data cleaning methods in the **Online Material – document cleaning**.

3.1.3 Dealing with finite context.

The context window is the range of tokens a model can process at a time. This includes all text, such as the input given to the LLM and the response it produces. When using an LLM with long, low information-density documents, the context window size can be a hurdle. In such cases, chunking strategies can be used to fit the text in pieces ("chunks") into the context window. However, careful consideration must be given when chunking documents, as the entity of interest could be located at any position in the text.[77] This consideration is particularly critical in chemical sciences literature, where key information is often dispersed throughout the document. For instance, abbreviations and nomenclature are typically introduced at the beginning, while synthesis procedures and experimental results may be described in separate sections. Therefore, the choice of chunking strategy is crucial (see Fig. 5). Importantly, even though the context windows of LLMs are rapidly growing,[78] there is also value in chunking because models can perform better if given a higher density of relevant content.[79]

Chunking techniques. The simplest approach is to divide the text into chunks of identical size to fit in the context window. This is also known as fixed-size chunking. One downside of this approach is that a word could be split between two chunks. One can improve upon this by splitting based on special characters like periods (.) or newlines (\n). In most situations, this is similar to splitting based on sentences and, hence, will leave words intact, but it is problematic when the text, such as scientific text, has numbers with decimal points. For our use case, it is important not to split these numbers.

Even though this approach preserves sentences, chunks of text are often related to the previous chunk. To help preserve some more semantic context from neighboring chunks, one can have a certain length of overlap between them. This, in practice, has been shown to produce better results.[80] The size of this overlap plays an important role in contextual awareness and the total number of required LLM queries.

In many cases, we can also use the fact that many documents are naturally split with headings and sections. We can use this to our advantage on top of all chunking techniques. This is called semantic chunking.

Using retrieval augmented generation (RAG) and classification to save costs. If the information density in the documents is low and one has to deal with many documents, one can store these chunks in a database and retrieve relevant chunks when needed to query the LLM. Typically, a vector database is used to store embeddings of the chunks while enabling fast retrieval of semantically relevant chunks. This process of retrieving information and using it as additional context when querying an LLM is called RAG.[81] Since embedding text is cheaper than running it through a generative LLM, this can be an effective strategy when dealing with many documents. However, the performance of this technique highly depends on the embedding model and retrieval strategy used.

A classification model can also be used to determine a text's relevance to the tasks at hand to improve extraction efficiency by, for example, determining prior to extraction whether a chunk contains relevant information.[35,82] The classification models used for this task can range from logistic regression trained on the text embeddings to using the LLM directly to classify text.

Chunking techniques and RAG are demonstrated in **Online Material – strategies to tackle context window limitations**.

Extending context windows. Attention, which is a key part of most LLMs, is costly to scale because the size of the attention matrix grows quadratically with the sequence length. A number of techniques have been developed to increase the model's context window.

Architectural changes (prior to training). One approach is to modify the model architecture so that it can manage larger context windows without a corresponding increase in computational cost. This can either be done by giving the model access to the previous hidden state (but not computing the gradients for the old states)[83] or by using an additional (external) memory. This can be, for example, in the form of a small number of extra tokens that store "global state"[84] or in a global memory to which context is added during training and which can be retrieved using elements of the conventional attention mechanism. Those techniques currently still share a problem that was also faced by recurrent neural networks (RNNs): adding new information to a fixed-size memory will dilute past memory over time.

Changes after training. In addition to changes to the actual model architecture, some changes can also be performed after training. For instance, models trained with so-called rotational positional encoding (RoPE),[85] such as Llama 3, can be fine-tuned to understand (relative) position of tokens over a range larger than the context window used in the actual training.[86] While this does not increase the actual context window, it can provide the model with a better understanding of the position of a chunk within a larger document.

3.2 LLM-interaction

Once the data from different sources is correctly parsed, the next step in the extraction workflow is to select the right modeling framework to proceed. This section aims to provide insights about the different models and learning paradigms. Furthermore, it presents the advantages, challenges, and potential strategies for optimizing the utilization of these tools in extracting structured data.

3.2.1 General model with prompt engineering. In principle, as pointed out in Fig. 6, using a leading general-purpose LLM[87–90] may seem a good first option due to its simplicity since no further training is needed. This option allows for rapid deployment and does not need further labeled data beyond the data needed for the evaluation of the extraction task. Such base models have been trained on a general corpus of data that does not belong to any specific field in particular and has mostly been mined from the internet. Thus, these models may lack knowledge and understanding of field-specific vocabulary.[22,91–94] However, as a result of the amount of data they were trained on (*e.g.*, the new Llama 3 model has been trained on over 15t tokens of publicly available data[95,96]), these models are often able to yield acceptable results for any of the typical natural language processing (NLP) tasks for data extraction, such as named entity recognition (NER) or relation extraction (RE) without further tuning.[97] It is important to note, however, that it is always valuable to try multiple models as, for example, large models might be better at recall but show a lower precision as Bran *et al.*[75] found for knowledge graph extraction from total synthesis documents.

When applying LLMs to extract information, one can adapt the model to the task by changing the prompt. Besides the actual text from which structured data should be extracted, the prompt contains the instructions that are given to the model as guidance to obtain the desired results. In addition, some advanced prompting techniques such as Chain-of-Thought (CoT)[98] have been shown to improve performance on various tasks. However, simple so-called zero-shot or few-shot prompting approaches are still the most common for generative data extraction.

Zero-shot prompting. Zero-shot prompting means that the text which is provided to the model does not contain any examples or demonstrations; it will only contain the task that the LLM is supposed to do (as well as the input text). When using this zero-shot prompt, the results solely depend on the knowledge and reasoning capabilities the model acquired during its training. If the model is unfamiliar with the task and vocabulary of the field, the outcomes are unlikely to meet the expected level of satisfaction, *i.e.*, the LLMs might not recognize dimethylformamide (DMF) as a solvent in synthesis or might not correctly understand how the compositions are expressed.[22]

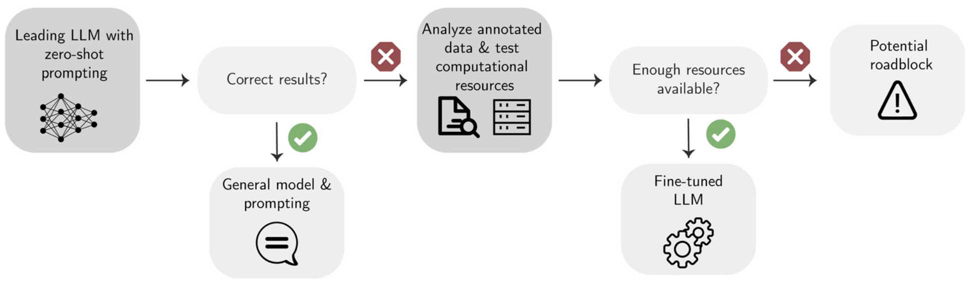

Leading LLM with zero-shot prompting
Correct results?
General model & prompting
Analyze annotated data & test computational resources
Enough resources available?
Fine-tuned LLM
Potential roadblock

Fig. 6 Decision tree on how to decide which learning paradigm is better for each case. The term "leading LLM" is intended to describe any general purpose LLM. The decision normally starts by testing a leading LLM with zero- or few-shot prompts. If the results are good enough, it is possible to continue with this method; if not, fine-tuning may be necessary. But fine-tuning requires additional labeled data and computational resources. If those are missing and simpler approaches do not work, pre-training also does not provide a solution, as it typically requires even more data (even though unlabeled) and computational resources.

Despite the simplicity, encouraging results have been obtained in this setting.[82,99] An alternative is to include some advanced prompting techniques. In this fashion, Polak and Morgan[37] prompted GPT-3.5, GPT-4, and Llama 2 models to extract information about materials such as pooling rates of metallic glasses and yield strengths of high entropy alloys. To make their zero-shot prompt more reliable, they developed a workflow with several follow-up questions. Within these follow-up questions, they included a technique called self-reflection,[100] which prompts the model to check its answers, looking for possible errors and correcting them if they exist.

Few-shot prompting. One of the most fascinating observations made in the development of GPT-3 is that LLMs can perform so-called in-context learning,[101] *i.e.*, learn a new concept without updating the model parameters but by only showing examples of desired behavior in the prompt.[102] Sometimes, even only one example (one-shot learning) is enough.[103] However, often more (k) examples (few-shot learning) lead to better performance, mainly because adding more examples helps prevent the model from becoming too focused on any single example.[104] The optimal number and order of examples will depend on each application highly and can be found by evaluating prompts with a variable number of shots.[105] Often, as in the case of Zhang *et al.*[30] for chemical data extraction, more examples improve the performance, but the number of examples one can provide is limited by the context window of the model. The examples that are provided must be chosen wisely, and it can be important to include not only positive examples but also negative ones, *e.g.*, JSON schemas with some empty optional fields.[106] To choose the examples for the prompt, one approach proposed by Liu *et al.*[107] is to use k-nearest neighbor clustering in the embedding space and build the prompt by including the k-neighbors for each test sample. Additionally, the examples can be modified to reflect the intended completion of the model better or to improve the conditions, *e.g.*, summarize the examples, so fewer tokens are consumed (see **Online Material – collecting data on the synthesis procedures of bio-based adsorbents**).

Examples of zero-shot and one-shot prompts are shown in **Online Material – choosing the learning paradigm**.

Advanced prompting techniques. Another possibility is to combine the few-shot prompt with other prompting techniques; for example, Sui *et al.*[108] combined the few-shot prompt with so-called self-augmentation. Self-augmentation consists of a two-step process in which the model is prompted twice, and the first answer of the model is used to improve the second prompt by giving insight into the data. Using this technique to prompt GPT-3.5 and GPT-4 models improved both models' results with respect to a simple one-shot prompt in better understanding and extracting data from tables of different datasets. They also observed that the performance increases significantly when going from zero-shot to one-shot prompts. One drawback that especially limits few-shot prompting is the context length of the models.[89] Nevertheless, the latest models released notably improved the context window (*e.g.*, the latest Gemini Pro model has a 2m token context window), allowing for a bigger number of shots[109] (for further discussion about the context length, see Section 3.1.3). Despite all the good results that these advanced techniques can give, it is not always clear that they are going to work (see **Online Material – collecting data on the synthesis procedures of bio-based adsorbents**).[110,111] However, due mainly to their ease of use, it is worth trying them before moving on to more complicated setups.

Many recent models differentiate between so-called system and user prompts. The system prompt provides high-level instructions or context that guides the model's behavior ("persona"). It sets the overall tone or rules for the conversation. However, the benefits of role prompting are still being debated.[93] A simple example of a system prompt for the data extraction task can be: "you are a chemistry expert assistant and your task is to extract certain information from a given text fragment. If no information is provided for some variables, return NULL". The system prompt is typically sent with the first completion and cannot be overridden by subsequent user follow-up questions. On the other hand, a user prompt represents the actual queries, statements, or prompts given by the user. They might contain the input text from which one wants to extract data, as well as a few-shot examples.

Prompt optimization. Typically, prompting techniques are implemented by hard-coding templates. This task can be made easier using frameworks like LangChain[112] or LlamaIndex,[113] which provide some of those templates. However, the models have varying preferences for prompts,[114] making actual prompting mostly empirical (or often manual and prone to data leakage) and constantly changing. This makes it impossible to create general and robust prompts. To address this, frameworks such as DSPy have been developed.[115] These systems perform the optimization of entire LLM pipelines automatically and systematically (*e.g.*, by autogenerating prompts or few-shot examples and then choosing the best ones using cross-validation). However, those approaches are still in their infancy and often require many LLM calls to optimize prompts.

Schema format. Another problem when using LLMs is the cost associated with their use. While trying to improve the performance when prompting the GPT-3.5 model for chemical data extraction, Patiny and Godin[116] reduced the number of tokens by using a YAML input schema instead of the common JSON format. Additionally, they proved that the GPT-3.5 model can perform well with YAML schemas. However, the performance of LLMs when working with different types of formats will vary among models depending on the training data[108,117] (see **Online Material – data annotation** for examples).

3.2.2 **Fine-tuning.** When the leading models fail to produce satisfactory results, fine-tuning can be used to improve a model for a particular task or domain by training it further on smaller but well-curated task-specific datasets.[118–122] It is important to point out that fine-tuning an LLM still requires a lot of expertise, which can be an important drawback (Table 2). However, Zhang *et al.*[30] found that by fine-tuning leading models (including GPT-3.5) on chemical data extraction tasks (such as reaction role labeling, metal–organic framework (MOF) synthesis information extraction, NMR data extraction) they could often outperform bespoke domain-specific models.

An example of how to fine-tune a model using an existing dataset is available at **Online Material – choosing the learning paradigm**.

Fine-tuning techniques. In principle, one can perform fine-tuning by changing all trainable weights of the model ("full fine-tuning"). This, however, results in a very computationally demanding and slow process. To address this, parameter-efficient fine-tuning (PEFT) techniques, in which only a few parameters (frequently only 1% of the total number of trainable parameters) are updated, have been developed. One popular PEFT technique is called low-rank adaptation (LoRA).[123] LoRA involves freezing the weights and then decomposing the updates' matrix (a matrix that contains the weight updates during the fine-tuning) into two lower-rank matrices that can be optimized during fine-tuning. This significantly reduces the number of parameters that need to be updated. Often, fine-tuning is done in a task-specific way, for example, focusing on NER + RE.[34] For instance, Dagdelen *et al.*[34] fine-tuned models to link dopants and host materials, and for the extraction of general information (chemical formulae, applications, and further descriptions) about metal–organic frameworks and inorganic materials. Fine-tuned LLMs have also been used for constructing a materials knowledge graph,[38] and extracting nanoparticle synthesis procedures.[124] Given enough diverse data, it is also possible to fine-tune models for general information extraction (IE) applications. For example, Sainz *et al.*[125] built a model called GoLLIE to follow task-specific guidelines, improving zero-shot results on unseen IE tasks.

One important drawback of fine-tuning is that it modifies the pre-trained model, potentially limiting its generalizability. This has been observed by Ai *et al.*,[126] who used the Llama-Adapter[127] method (another PEFT technique) to fine-tune a Llama 2–7b model for the extraction of reaction information from organic synthesis procedures. Interestingly, it has been shown that models tuned with LoRA are less prone to forget the general capabilities of the base model but are also less proficient in learning new capabilities compared to full fine-tuning.[128,129]

Human-in-the-loop annotation. One of the main limitations of fine-tuning is the need for large amounts of annotated data. To solve this problem, Dagdelen *et al.*[34] proposed a human-in-the-loop annotation process in which a model performs the initial data annotation. A human expert corrects possible errors in the annotated data, and the resulting corrected data is used to further fine-tune the model that continues to label the data. Using this annotation method, they reduced the time needed to annotate each sample by more than half for the last annotated abstracts since correction is faster than annotating from scratch.

3.2.3 Pre-training.

If fine-tuning falls short, one costly resort might be to train a model from scratch. For LLMs this usually requires prohibitory computational resources, datasets, and expertise, as described in Table 2. For example, training the new Llama 3 models needed almost 8m GPU hours using two clusters with 24k GPU each.[95,96] However, it is possible to fine-tune the Llama 3–70b model using a single GPU with 80 GB of memory.[34]

However, when the data to extract is very specific and general capabilities are not required, good results can be achieved using smaller language model (LM) such as the ones of the BERT series (so-called encoder-only LMs).[130] For those smaller models, less training data and less computational resources are required. For example, SciBERT was pre-trained on a corpus of 1.14m articles, resulting in 3.17b tokens.[131] Similarly, MatSciBERT was trained on 3.45b tokens (compared to 300b tokens for GPT-3).[132] Note that for those smaller models, it is still recommended to fine-tune on the task that the model is intended to perform. In this way, Cole *et al.* built task-specific BERT models by pre-training them on batteries,[20] optical materials,[133] or photocatalytic water

Table 2 Key factors to consider for each learning paradigm assuming that the model is already deployed, either by API or by other means. Note that we assume a first-year chemistry student is the user and consider the use cases in which the student may be interested. We assume for all cases the use of proper LLMs (>1b parameters). "Prototyping time" key factor refers to the time that it takes the user to do the first tests and see if the model can be used for that task. "Field-specificity" intent to explain the understanding that the model can deal with terminology that is particular to a certain field, such as chemical formulas. This factor is especially related to the tokenizers of the models. Tokenizers need to be set prior to training. Hence, tokens relevant to a particular topic might be missing from pre-trained tokenizers. In pre-training or fine-tuning additional tokens can be added, which can result in an improvement when extracting data from texts with complex notation

Key factors	Zero-shot prompting	Few-shot prompting	Fine-tuning	Pre-training
Expertise	Very low	Very low	Medium/high[a]	High
Prototyping time	Minutes	Hours	Days	Weeks
Field-specificity	None[b]	Low[b]	High	Very high
Task-specificity	None[b]	Low[b]	High	Medium
Data	None	10^0–10^1 examples[c]	10^2–10^3 samples	$\approx 10^{12}$ tokens

[a] Depending on how much data is available or needs to be cleaned and prepared. [b] Considering knowledge added to the general pre-training of the model during inference. [c] The number stands for the traditional use of the few-shot paradigm. Thus, with the actual context length of some current models, the number of shots used can increase a lot.

splitting[134] field-specific text extracted from scientific articles. After the pre-training, they fine-tuned them on the Q&A task using the Stanford question-answering (SQuAD) v2.0 dataset.[135] This pre-training and fine-tuning approach of smaller encoder-only models has also been applied to extract polymer property data[21] and synthesis protocols of heterogeneous catalysts.[40]

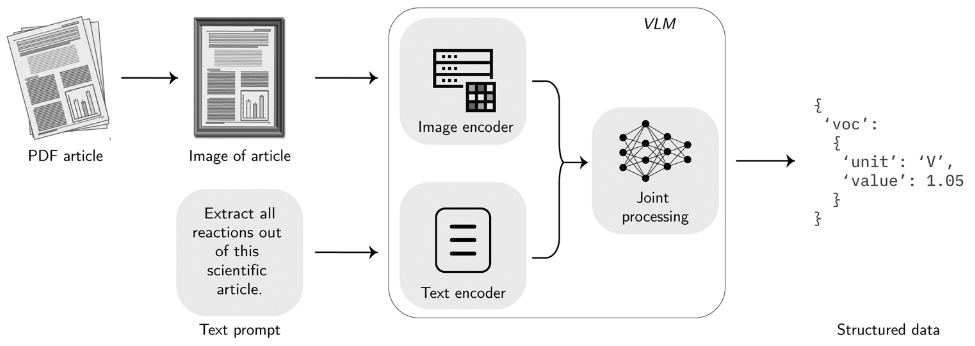
Even though training smaller LM needs less data and computation compared to training an LLM, training a small BERT from scratch might be more costly than performing fine-tuning using LoRA. As shown by Song *et al.*,[136] LoRA-based instruction fine-tuning of Llama models can outperform BERT-based models on a suite of materials science tasks relevant for data extraction.[137] Thus, before deciding to train an LLM for specialized tasks related to data extraction, we recommend exploring other options outlined in this review to find potentially feasible alternatives.

3.2.4 Beyond text

Vision models. While there are plenty of tools for converting text, *e.g.*, in PDF documents, in machine-readable markup language as discussed in Section 3.1.2, the challenge of analyzing and converting complex structures like large tables or plots can often not be easily addressed with those tools.[138] This problem is becoming even more prominent in the extraction of chemical data, where, in addition to plots and tables, crystal structures, reaction schema, complex spectra, and intricate graphical representations contain vast amounts of critical information. Besides classic text processing models, there are also models that were specifically trained to understand and analyze images alongside text, which are so-called Vision language models (VLMs). These models typically consist of three main components: an image encoder, a text encoder, and a component that fuses the information of these two.[139] The main advantages of the usage of such VLMs are the additional information gained through images and plots and the end-to-end procession of the data since the loss of information due to conversion steps mentioned in Section 3.1.2 is minimized.[140] Moreover, the end-to-end approach is preferred since models could choose and learn the best preprocessing method instead of humans choosing and hard-coding bespoke processing workflows. For instance, some of these models can leverage the specific layout of scientific articles, such as figures, tables, and captions, and, therefore, can extract the data more accurately. The downsides of using VLMs instead of LLMs are potentially higher costs for the data extraction. However, also VLMs still requires some preprocessing of the input, like converting the input into suitable images and potentially removing irrelevant sections like acknowledgments or references or resizing and scaling the images. An example of a basic extraction workflow as shown at Fig. 7 can be found in the **Online Material – beyond text**.

A question that arises is in which case one should use an OCR-LLMs extraction pipeline or a VLM for extraction instead. Since usually LLMs model calls are less expensive, starting by testing these models instead of VLMs is typically preferred for the extraction of mostly text-based articles. However, if the input data includes a lot of complex structures like tables and plots, a VLM can often be a suitable choice, as demonstrated in the extraction of copolymerization reactions[141] or electrosynthesis reactions[142]—in particular if the focus is on obtaining results without building complex preprocessing or agentic pipelines. For example, Lei *et al.*[36] demonstrated the higher efficiency of a VLM in the task of detection of the material present in micrographs than a text-only LLM.

A wide variety of open-source and commercial vision models are available. The GPT-4 vision and Claude 3 models are the largest and most widely used models. Although these are commercial, some open-source VLM such as DeepSeekVL also have achieved good results.[143] Liu *et al.*[144] used VLMs to perform different OCR tasks and introduced a benchmark to compare the performance of different VLMs. While for simple tasks like recognizing regular text, the VLMs achieved performance comparable to state-of-the-art OCR-tools, they performed poorly on unclear images, handwritten text, and adhering to task instructions, as well as in extracting knowledge graphs of organic chemistry reactions.[75] Alampara *et al.*[145] have shown that leading VLMs struggle to analyze basic spectra and extract information from reaction schema and tables. They have also exposed that the current models still fail to understand the images' content. Therefore, using specific tools developed for these tasks might still notably increase the accuracy.



```
{
 'voc':
  {
   'unit': 'V',
   'value': 1.05
  }
}
```

Fig. 7 Workflow of data extraction with VLMs. Papers, for example, in the form of PDFs, can be converted into images and then processed using an image encoder. The prompt containing the instruction is still provided in the form of text and the output of the VLMs is structured data in the form of text.

Tools for special inputs/modalities. Besides these general vision models, several tools are available for specific tasks and structures, such as tables, chemical structures, and plots. Since tables in scientific articles can be quite complex, a dedicated focus on these could help to increase accuracy.[146,147] For instance, tools like TableTransformer[148] or DISCOMAT[149] are specifically optimized for dealing with data in tables, while Liu *et al.*[150] reported a plot-to-table translation model.

In chemistry and materials science, the most important information about chemical components and reactions is often hidden in images of structural formulas and reaction schemes. Therefore, these non-machine-readable depictions have to be extracted separately from the text. For instance, the ReactionDataExtractor 2.0[151] opens the possibility of extracting a complete reaction scheme, including components and reaction conditions. OpenChemIE also extracts reaction data from text or figures.[152] Nevertheless, many of these tools encounter problems with variable end groups mostly noted as 'R-group'.[153]

Another important modality for data extraction is plots and images. The WebPlotDigitizer V4 is an open-source tool to extract data from plot images to numerical data,[154] as shown by Zaki *et al.*[155] to extract glasses information from graphs. In addition, there has been a focus on extracting data (*e.g.*, shape, size, and distribution of the pictured particles) from microscopy images.[156–158]

Due to the wealth of tools available and the various possible use cases (leading to a combinatorial explosion of possible combinations of tools), one can consider using an agentic approach described in Section 3.2.5 to automatize the usage of these.

3.2.5 Agentic approaches. LLM-based agents are novel decision-making agents powered by one or more LLM that can interact with (and modify) an external environment.[159,160] They have shown promise in tasks as varied as simulating human behavior,[161] playing Minecraft,[162] molecule design,[163–165] and autonomously performing chemical reactions.[166,167] As a result of the several reasoning cycles that the agents perform, they often outperform vanilla LLMs.[168–170] But what advantages can they present in extracting data?

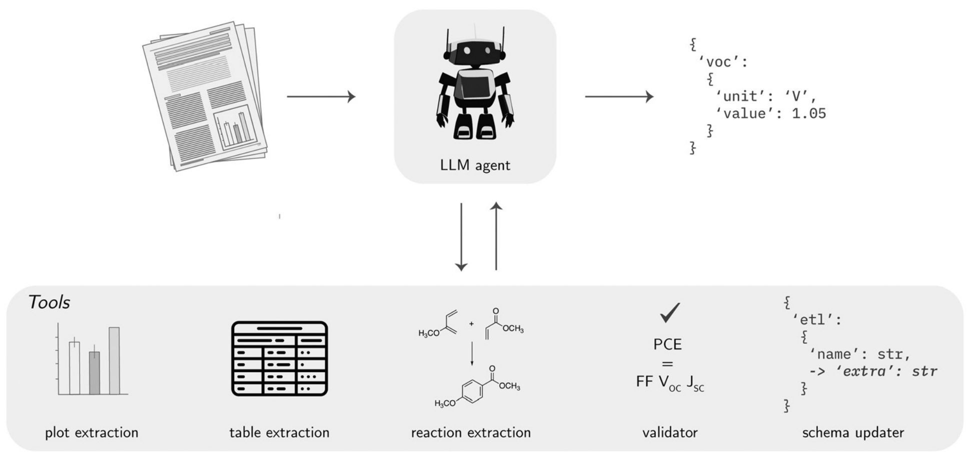
Why to use agents for structured data extraction?

Flexible dynamic workflows. The use of agents can be especially interesting for data extraction when working with multimodal data, *i.e.*, not only text but also tables and figures (Fig. 8). Although VLMs can understand figures, the results when extracting data might be not good enough,[144] especially when this data is field-specific, such as specialized scientific images.[171] Even though specialized tools exist for many of these use cases a limitation of all these specialized tools is that they must be manually used or manually chained into workflows by developing bespoke programs. However, an LLM agent with access to these tools can autonomously build dynamic workflows without manual human input, making data extraction more accessible, scalable, and flexible.[172]

Higher accuracy. Another important advantage of using agents is that they can show the ability to automatically improve and correct wrong results by using self-reflection and self-criticism of previous actions.[173] This ability has been shown to reduce hallucination considerably in some data extraction applications,[174] even achieving superhuman performance.[79]

Design patterns for LLM-powered agents. Different classifications have been proposed in the literature to define the agents.[159,169,175,176] However, we think that the most appropriate for the data extraction task is a classification close to the one proposed by Weng.[160]

Planning. Complex tasks typically involve many steps. Thus, agents need to be able to decompose tasks and plan ahead. In LLM-agents, planning is typically provided by the



```
{
 'voc':
  {
   'unit': 'V',
   'value': 1.05
  }
}
```

```
{
 'etl':
  {
   'name': str,
   -> 'extra': str
  }
}
```

Fig. 8 General workflow of an agent for the data extraction task. The unstructured data from the articles is given to the agent as text, images, equations in LaTeX form, or any other format that can be given to the models. Using its reasoning capabilities, the agent decides which one of the available tools is best to extract each data type. When all the data available are extracted, the agent provides it as structured data.

reasoning capabilities of the LLM. The simplest form of task decomposition is CoT prompting, which utilizes test-time computation to decompose the task into simpler steps by prompting the model to "think step by step". This framework has been extended in a multitude of ways, such as tree of thoughts,[177] which, similar to CoT decomposes tasks into steps but then, in a tree-like fashion, explores multiple reasoning paths at once using techniques such as breadth-first search (BFS).

Reflection. Self-reflection is a vital aspect that allows autonomous agents to improve iteratively by refining past action decisions and correcting previous mistakes. It is a common design pattern that involves the system explicitly criticizing and evaluating its output and subsequently refining it in an automated setup, often leading to notable performance gains.[100,178,179]

Memory. Another important building block for solving complex tasks is the ability to memorize information. Apart from the information provided to the agent within the context window of the LLM, it is possible to embed previous interactions of the agent in a vector database and retrieve them using RAG.[180] This enables systems to retain and use information across very extended periods.

Tool use. As alluded to above, one of the most important design patterns for LLM-powered agents is to let them use external tools such as specialized models,[181] web APIs,[182] simulation engines, or databases (*e.g.*, citation traversal[79]) to make up for the information and capabilities that might be missing. A powerful tool can also be additional LLM calls, as in the reranking and contextual summarization step in PaperQA.[79] Here, the tool summarizes different chunks of papers and then rates the relevance for answering the question (in contrast to naïve retrieval that is typically performed in RAG systems). The so-called ReAct framework is a common way to implement this.[183] It prompts the LLM to reason about the user's query and then act by choosing a tool. It is important to mention that the reasoning is done through inference, *i.e.*, the model reasons only by the completion. The actions will lead to observations (*e.g.*, responses from an API call) that might lead to further think-act-observe cycles before a final answer is returned.

A simple case about building an agent with custom tools can be found in **Online Material – agents**.

Multi-agent collaboration. The multi-agent approach has proven to be a robust variant for self-reflection and self-criticism.[184] This approach involves making more than one agent work together. By doing this, it is possible to define different roles. For example, defining one evaluator agent and one critic agent can greatly improve the overall results. Another possibility is to include an agent that provides feedback or to add self-feedback to all the agents. Feedback can help the agents better understand possible areas of improvement, leading to better system performance. In multi-agent setups, it is even possible to create "creator" agents[185] that create new agents with a specific role and goals to accomplish.

Limitations. Current agent workflows are still limited in several ways, most importantly by the error rate of the base model, which leads to a high risk of failures in longer workflows.[111]

Error amplification. A fundamental problem of autoregressive models such as current LLMs is that errors accumulate since the outputs of the models are the inputs for the subsequent generation. In the case of agents, this means that in a long reasoning path, if one tool has an error, the agent passes this error to the next tool or reasoning step. The following tools can amplify the initial errors leading to large errors in the final answer.[186,187]

Limited context-length. Similar to other applications of LLMs, agents are limited by the finite context length of the model.[188] This is particularly pronounced for agent systems as they typically "memorize" the planning, reflection, or tool use traces within their finite context window.[189] This problem might be resolved by the rapidly growing context windows of frontier models.

Endless loop. Another situation that may happen is that the agents can end up stuck in the same loop forever. This situation can arise as a result of hallucinations and chaining different thoughts.[190] This is especially likely to happen when dealing with complex problems that the agent may not be able to solve.[185]

Safety-critical interactions. Since agents are supposed to interact with their environment,[191–193] giving them access to computer systems or even laboratories requires additional safety precautions.[167,194,195]

Challenges in evaluation. Finally, these agents are very difficult to evaluate and optimize since not only the final answer must be evaluated but also the reasoning process.[196,197] The evaluation of the reasoning path can be necessary for the proper application of these systems for open-ended tasks as it allows for identifying the different sources of errors, *i.e.*, if the error comes from a bad reasoning step of the LLM or if it comes from a wrong answer by one of the tools.[162] Evaluating the reasoning path is a very difficult task because of the freedom that the agent has to reason and make decisions. In addition, most existing benchmarks rely on a ground truth; however, for many open-ended tasks, such a ground truth may not be available.

3.3 Postprocessing

3.3.1 Constrained decoding and enforcing valid outputs. The raw outputs of an LLM are probability distributions over tokens (see Section 2). The distributions indicate how likely each of these tokens (*i.e.*, text fragments) are a continuation of the given text. By default, LLMs consider all possible tokens they have seen during training, which means every token has a nonzero probability. There are different techniques ("sampling strategies'') for choosing the next token, thus text, from the probability distribution. Naïvely, one might always choose the next token with the highest probability (''greedy decoding''). It has, however, been shown that this sometimes leads to unnatural text. Thus, practitioners often sample from the distribution to obtain natural text, where a factor called temperature indicates how frequently the sampling will give less probable tokens. This is nonideal for two reasons. First, we often know *a priori* that only a certain subset of tokens might be relevant—for example, we may care only about numbers, and limiting the pool of tokens we consider to only numbers will increase our chances of generating one. Second, sampling introduces randomness, which is typically unwanted in structured data extraction. The latter point can often be avoided (without taking into account randomness due to hardware or inherent tradeoffs in model deployment[198]) by always taking the most probable token (*i.e.*, setting temperature to zero). The former point can be implemented *via* so-called constrained decoding techniques. Those techniques can ensure that only a subset of ''allowed'' tokens will be used in the sampling stage (see Fig. 9).

Constrained decoding can be implemented in different ways. One of the first widely popularized implementations has been in jsonformer.[199] The key to understanding this approach is that models generate outputs one token at a time. That is, predictions are used as inputs to make the next predictions (so-called autoregressive generation). If one aims to create structured data, one can make multiple optimizations in this process. First, some of these tokens are obvious, can be generated using code, and do not have to be generated by a model. For instance, code can generate much of the structure of a JSON file, such as the opening brace. That is, we can make the generation more efficient by only generating the content tokens using a model. Second, if we know the types of our data, we can limit the pool we sample from to only a subset of all tokens. For instance, Llama 3's vocabulary encompasses 128 256 tokens. If we want to fill a field with a boolean value, however, we only need to compare the values of the tokens for ''true'' and ''false'' (*i.e.*, two out of 128 256 tokens). In the simplest setting, one can start by already inserting a part of the desired output in the response, *e.g.*, {''query'': One can then decode until a stopping criterion is reached. This would be any token that is not a number or decimal separator (.) for simple numeric fields. From there, one can then start again by inserting all the text that is already known to be part of the desired output, *e.g.*, a closing comma (,) and the next key. These ideas can be extended to more complex constraining patterns, for example, based on formal grammars.[200] These formal languages can describe, in principle, the structure of any computable object.[201] Thus, one could, for example, constrain the model to generate syntactically valid code in a programming language of choice (as programming languages can be described using formal grammar). While such constrained decoding is now well-supported in libraries such as outlines,[202] instructor[203] (see **Online Material – collecting data for reactions procedures**), marvin,[204] ggml (which supports grammars provided in Backus–Naur form),[205] or even the OpenAI API (*via* JSON mode and function calling, or its newest feature, structured outputs, which ensures that the model's response adheres to a provided schema *via* specifically finetuned models), it is still not widely used for generative data extraction in the chemical sciences.[206] This is a promising future research direction as many relevant chemical datatypes (*e.g.*, IUPAC names) can be represented with a formal grammar. One limitation with those approaches, however, is that they cannot naïvely be used with advanced prompting techniques that nudge the model to ''think'' by generating tokens. In those cases, it might make sense to use multiple prompts.

A middle ground between prompting without type and syntax constraints and constrained decoding can be to provide type hints in the prompt. These type hints can, for instance, also be literal, meaning a list of permitted strings.

Examples of constrained decoding and enforcing valid outputs are shown in **Online Material – constrained generation to guarantee syntactic correctness**.

To aid the integration with existing knowledge bases, one can also ground the LLM output on established ontologies as, for example, Caufield *et al.*[207] have implemented in SPIRES, which demonstrated effectiveness in extracting chemical-disease relationships from biomedical literature, grounding entities to standardized identifiers (*e.g.*, mapping ''Cromakalim'' to MESH:D019806).

3.3.2 Evaluations.

To optimize a system, its performance needs to be evaluated. In the case of structured data extraction, this is not trivial as there are many different yet related data items one extracts in a potentially nested data model (Fig. 10).

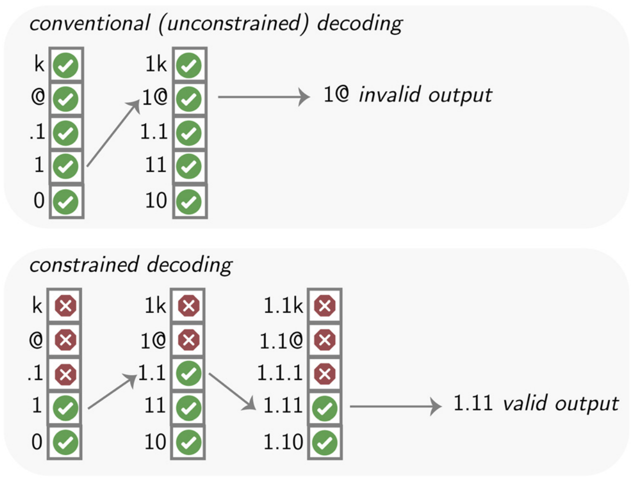


Fig. 9 Example of generating a number with conventional and constrained decoding. In conventional decoding (top), all tokens have a non-zero probability of being sampled (indicated by green checkmarks). Thus, there is a non-zero probability that outputs are being generated that are not a number. With constrained decoding (bottom), however, we can dynamically adjust the set of tokens from which the system samples to generate only valid outputs. For instance, in the first step, only the two integer tokens are allowed. In the next step, also .1 is allowed, as it might lead to a valid number. However, once .1 has been sampled, it is no longer allowed (indicated by red crosses) as a number can only contain one decimal point.

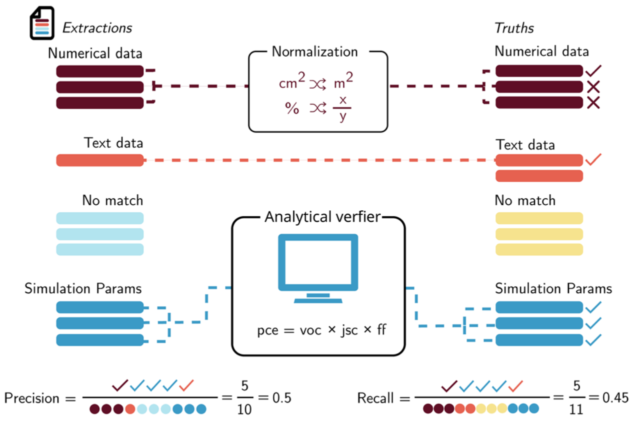


Fig. 10 Evaluation workflow. The extracted structured data is displayed on the left, while the manually labeled truth data is on the right. The colors indicate matching on both sides. Checkmarks indicate correct keys, crosses indicate incorrect ones, and circles refer to the keys in the extracted and true data. Notice how there is an unmatched blue set in the extractions and a yellow set in the truths. This affects how precision and recall are calculated. Numerical data needs to be normalized if the units are reported differently from how they are stored in the truth data. Certain fields, like the simulation parameters, can be validated using scientific analysis tools to make sure they obey all domain rules.

In addition, many entities can be reported in multiple synonymous ways. Thus, it is essential to carefully inspect the extracted data for common error sources. This manual inspection then allows an informed development of an automated evaluation pipeline that allows for systematic assessment of the extraction performance of different systems. As Sayeed *et al.*[206] indicate, the lack of a properly defined metric for structured data extraction makes systematic evaluation challenging.

Scoring the extraction of a simple entity extraction. The simplest setting can be thought of as extracting a single value. For example, if the task is collecting all chemicals in a document, the performance of a model could be scored by measuring how many chemicals we found (retrieved entries) out of all the ones reported in the documents (relevant entries) and how many of the extracted chemicals are actually correct. The first measure is recall (eqn (1)).

$$\text{Recall} = \frac{|\{\text{relevant entries}\} \cap \{\text{retrieved entries}\}|}{|\{\text{relevant entries}\}|}, \tag{1}$$

where the $\cap$ symbol denotes a set intersection between the set of entries, and |set| indicates the number of items in a set.

The second measure is known as precision and is defined in almost the same way, except for the denominator being the retrieved entries (eqn (2)).

$$\text{Precision} = \frac{|\{\text{relevant entries}\} \cap \{\text{retrieved entries}\}|}{|\{\text{retrieved entries}\}|}. \tag{2}$$

Intuitively, recall can be thought of as how many of the values we expected are extracted correctly and precision as out of just the extracted values how many are actually correct. Based on those two definitions, additional metrics can be defined, such as the F_1-score (which is the harmonic mean between precision and recall and also known as the Dice score).

Due to the infancy of evaluations of structured extractions, there is much confusion about how to use metrics meant for classification models for this task. We propose to define true positive as a correct value extracted by the LLM. False positive is when there is an extraction but the value does not match what we expect. False negative can be taken as a value that is in our ground truth but has not been extracted by the LLM. True negative is not applicable and cannot be realistically defined for our task. True negative can be thought of as a value that we did not ask the LLM and it did not provide it. Therefore, it is every concept or word available in our document or vocabulary that hasn't been reported. Table 3 provides examples of these different error types one might encounter in structured data extraction. Using these definitions, we can calculate many metrics like the F_1-score as it does not use true negative (TN). The recall

Table 3 Examples of error types. In the case of true positives, the extracted data matches the ground truth labels. In case there has been an extraction, but for an entry that does not exist in the ground truth, we call it a false positive ("hallucination"). False negatives are values the LLM missed to extract. As there are potentially infinite true negatives (not existing data the LLM did not extract), it is not meaningful to consider them

Outcome type	Content example	Extracted data	Expected data
True positive (TP)	The properties of TiO_2 include a bandgap of 3.2 eV, which is typical for materials used in photocatalysis.	Formula: TiO_2, bandgap: 3.2 eV	Formula: TiO_2, bandgap: 3.2 eV
False positive (FP)	The properties of TiO_2 include a bandgap of 3.2 eV, which is typical for materials used in photocatalysis.	Formula: CeO_2, bandgap: 3.37 eV	Formula: TiO_2, bandgap: 3.2 eV
False positive (FP)	The properties of TiO_2 include a bandgap of 3.2 eV, which is typical for materials used in photocatalysis.	Formula: TiO_2, bandgap: 3.37 eV	None
False negative (FN)	The properties of TiO_2 include a bandgap of 3.2 eV, which is typical for materials used in photocatalysis.	Formula: TiO_2,	Formula: TiO_2, bandgap: 3.2 eV
True negative (TN)[a]	The properties of TiO_2 include a bandgap of 3.2 eV, which is typical for materials used in photocatalysis.	None	None

[a] Practically cannot be calculated.

and precision based on true positive (TP), false positive (FP), and false negative (FN) looks like this:

$$\text{Precision} = \frac{|\text{TP}|}{|\text{TP}| \cap |\text{FP}|} \quad \text{recall} = \frac{|\text{TP}|}{|\text{TP}| \cap |\text{FN}|}. \tag{3}$$

Matching to ground truth. One challenge with these tailored metrics for more complex data models is the presence of multiple instances of the same object type. For instance, when extracting reactions, there may be more than one reaction in a paper. This poses a problem when scoring performance, as the ground truth and extracted output may contain different numbers of reactions. The model may have missed some reactions or falsely identified others. To accurately score a specific field, such as the yield of a reaction, the extracted entities must be matched with those in the ground truth before a comparison can be made.

To achieve this, it is recommended to first define a unique identifier for entities (such as a sorted list of the normalized reactant names in reaction) and then utilize a fuzzy matching score, such as the Levenshtein edit distance, to match each extracted entry with one element from the ground truth. This matching process can be seen as a one-to-one mapping between two lists (extracted outputs and ground truth), which is achieved by minimizing the total distance between all pairs. More formally, this is known as the linear sum assignment problem.[208]

Data normalization. For many extraction tasks, normalization is relevant before calculating metrics. For chemicals, this is relevant because there are often multiple equivalent ways of naming the same compound. While the normalization workflow will differ from use-case to use-case, tools such as PubChem[209] or the Chemical Identifier Resolver Server[210] will often be useful to derive canonical representations for chemical compounds. Additionally, when dealing with units, tools such as pint[211] or unyt[212] can be used to convert data into standard units before performing metric computations (see **Online Material – evaluations**).

Overall metrics. In addition to specific metrics, having a general metric, that provides a score on the whole extraction task, can be helpful. The Damerau–Levenshtein edit distance is one such metric. It measures the minimum number of operations needed to change one document into another, including inserting, deleting, substituting, or transposing characters. This metric must be computed on canonicalized documents, *i.e.*, documents that have been sorted and encoded similarly.

Validation using chemical knowledge and understanding. Data extraction in chemical and materials science has a significant advantage over other domains due to our understanding of rules and links between different data entries. This knowledge enables us to conduct "sanity checks" to ensure the accuracy and consistency of the extracted data. Despite this unique opportunity, it has not been widely utilized. A notable example is the work of Patiny and Godin,[116] where they extracted molecular properties from text and used cheminformatics tools to validate the consistency. For instance, they extracted nuclear magnetic resonance (NMR) spectra and used cheminformatics tools to verify if they were consistent with the given molecular formula (see **Online Material – validation case study: matching NMR spectra to composition of the molecule** and **Online Material – retrieving data from chalcogenide perovskites for an example using stability criteria for perovskites**). Similar validation can be performed on other experimental data, such as mass spectra and elemental analysis (see **Online Material – collecting data for reactions procedures** for an example of checking the extraction of the correct number of atoms on both sides of a reaction). The benefit of these consistency checks is not only improving data quality but also enabling a first evaluation loop without the need for manually labeled data.

Beyond the use of validation based on chemical knowledge, it can also be practical to use another LLM to check, for example, for factual inconsistencies (*e.g.*, if there were hallucinations during the extraction).

4 Frontiers

LLMs have greatly advanced the capabilities of data extraction given their ability to process text and other data modalities, such as figures through VLMs. These advancements open up further compelling opportunities to make data extraction more robust and accessible, which we describe along a handful of research frontiers.

4.1 Improving multimodal models

As described in Section 3.2.4, data modalities beyond text often lead to unique challenges. While VLMs can be quite helpful in extracting useful information from diverse types of data one might encounter in scientific literature, such as tables, formulas, structure files (*e.g.*, CIF files for crystals[213]), and sub-figures containing images with intricate relationships. Future work remains to make them more robust and amendable to the diversity of data found in materials science and chemistry. As described in Hira *et al.*,[22] some of the complexity can arise when diverse modalities are contained within a different data structure, such as chemical composition and related properties within a table that are linked to a figure in other parts of the document. Furthermore, the data format that current methods provide, such as LaTeX or XML code, may not be ideal for training LLMs and other AI models for desired downstream tasks. As such, further work is needed to expand the capabilities of modern tools to not only provide structured data but also diverse forms of structured data.

4.2 Cross-document linking

As reported by Miret and Krishnan,[214] current extraction methods, including LLMs and VLMs, mainly focus on extracting data contained within a single document or a sequence of documents given in a broader window, as described in Section 3.1.3. Much of scientific data, however, relies heavily on referenced work for relevant concepts, descriptions, and experimental results (that might be in other repositories).[22,215] In scientific publishing, including this work, it is very common for a scientific report to reference a procedure from another paper or report for brevity and accreditation. As such, the references and their potentially important context are usually not considered for information extraction methods today. The exclusion of such information likely limits the performance of modern LLMs and VLMs to perform relevant chemical tasks.[214] This also extends to data extraction, where understanding the relevant scientific background of a given figure, table, or data modality could help the LLMs process the data to the appropriate structured format and provide semantic meaning. One potential approach for addressing this challenge could be using multiple agents to analyze the same set of data with one agent providing relevant scientific background, such as a RAG-based chemistry LLM, and another providing extraction capabilities for the figure itself, such as VLMs.

4.3 Scientific literature bias

The scientific literature is strongly biased towards positive results and highly refined information presented in curated text, figures, and tables. While it is important to maintain the high-quality standard in scientific publishing, there are likely adverse effects of not having LLMs observe negative results that are common in a wide range of real-world use cases, such as internal reports and communications.[7,216] On top of that, many works of the scientific literature contain only incomplete information, which has fueled a reproducibility crisis in various fields, leading to concerns that advanced AI methods, such as LLMs, will make the situation worse.[217] As such, it is important to improve data reporting methods that enable better dissemination of scientific knowledge in line with the development of scientific LLMs.

4.4 Beyond data extraction from papers

Much of scientific innovation occurs when deploying new types of capabilities, such as synthesis equipment, characterization tools, and scientific simulation codes. The dynamic nature of these tools makes it possible for such new types of data structures and modalities to be continuously invented and deployed for diverse sets of applications. Ideally, LLMs should be capable of extracting relevant information from new tools and procedures. This represents a fundamentally new problem that future research work can tackle, building on the approaches described in this review. Agentic approaches described in Section 3.2.5 may be a useful framework given the flexibility of adding diverse tools for upcoming data modalities.

4.5 From query to model

Given the emergence of agentic systems that can autonomously build ML models,[218] it is not difficult to envision coupling data search agents (*e.g.*, PaperQA[79]) with data extraction agents and ML agents. The result would be a system that can take in a search query and autonomously find data to train a model to answer the question. However, those systems would face the same challenges we discussed in Section 3.2.5 such as fragility and complex evaluation.

4.6 Benchmarks and open questions

For information extraction, most existing benchmarks focused on evaluating the performance on separate tasks such as NER, or RE. More comprehensive benchmarks have been scarce, given the high cost of data labeling. This challenge, however, is unlikely to abate given the fact that new scientific discoveries continue to expand human scientific knowledge. As such, it might be useful to redirect the development of benchmarks towards adaptation of LLMs in low-data scenarios. While this has been observed in prior work, some potentially relevant benchmarks are too easy for modern LLMs[137] while others measure capabilities in adjacent tasks.[219] New benchmarks should address the challenges of LLMs and VLMs for current capabilities and enable research along the current frontiers described above. This prompts a diverse set of remaining questions that the research community can work towards, including but not limited to:

- What modalities exist in chemistry that current LLMs and VLMs cannot process?
- What methods can appropriately handle the complex relationships of knowledge contained across multiple documents and knowledge sources in the chemical sciences?
- How can we mitigate scientific literature bias[216] to build more comprehensive databases that can be digested by LLMs?
- How can we integrate the developments in LLMs for data extraction into broader efforts to build performant scientific assistants for chemistry?

5 Conclusions

Structured data is immensely important for the advancement of science. As shown in Fig. 1, the aggregate information across diverse sub-fields in chemistry and materials science continues to grow at a notable pace. While there have been prior attempts to systematically extract data from these sources, only LLMs present a scalable solution to address both the breadth and scale of scientific data.

We hope that this review enables chemists and materials scientists to profit from these developments, thereby accelerating the understanding and discovery of new compounds that further scientific knowledge and enable extraordinary technological advancement—leading from text to insights.

Glossary

Chain-of-Thought	A prompt engineering technique that involves a series of intermediate natural language reasoning steps that lead to the final output.[98] CoT encourages an LLM to explain its reasoning step by step (*e.g.*, by prompting it to ''think step by step''), and it is intended to improve the ability of LLMs to perform complex reasoning.
Chain-of-Verification	A four-step process intended to reduce hallucination in LLMs:[220] (i) the LLM generates a baseline response; (ii) generates a list of verification questions to self-analyze if there are any mistakes in the original response; (iii) answers each verification question in turn, checking the answer against the original response to check for inconsistencies or mistakes; and (iv) generates the final revised response incorporating the verification results.
Information extraction	The process of automatically extracting information from unstructured text. This process involves converting raw text into structured data by recognizing entities, relationships, and events.
Language model	A ML model that estimates the probability of a token or sequence of tokens occurring in a longer sequence of tokens. This probability is used to predict the most likely next token based on the previous sequence. Language models are trained on large datasets of text, learning the patterns and structures of language to understand, interpret, and generate natural language.
Large language model	A language model that has a large number of parameters. There is no agreed rule for the number of parameters that makes a language model large enough to be called a large language model, but this number is usually on the scale of billions.[221] For example, Llama 3, GPT-3, and GPT-4 contain 70b, 175b, and 1.76t parameters, respectively, while Claude 3 Opus is estimated to have 2 trillion parameters. Most current LLMs are based on the transformer architecture. LLMs can perform many types of language tasks, such as generating human-like text, understanding context, translation, summarization, and question-answering.
Low-rank adaptation	A PEFT technique that freezes the pre-trained model weights and decomposes the weights update matrix into two lower-rank matrices that contain a reduced number of trainable parameters to be optimized during fine-tuning.[123]
Masked language modeling	A method of self-supervised learning in which a random sample of the input tokens are masked and, during the training, the model is asked to predict the original inputs for the masked tokens based on the context provided by the unmasked tokens.
Named entity recognition	A NLP technique that identifies and classifies named entities within text into predefined categories such as names of people, organizations, locations, dates, and other entities.
Name entity recognition and relation extraction	A combined process in NLP that first identifies and categorizes entities (such as materials and properties) within a text and then determines semantic relationships between these recognized entities. This dual approach enables the extraction of structured data, revealing how entities interact within the text, and is crucial for building knowledge graphs or information retrieval systems.
Natural language processing	A subfield of computer science that uses machine learning to enable computers to understand, interpret, and generate human language. The main tasks in NLP are speech recognition, text classification, natural-language understanding, and natural-language generation.
Next sentence prediction	A task used during the pre-training of LLMs where the model is trained to predict whether a given sentence logically follows another sentence. This task helps the model learn the relationships between sentences, improving its understanding of context and coherence in text. By training on vast amounts of text data, the model develops the ability to generate coherent and contextually appropriate continuations for a given text.
Optical character recognition	A technique used to identify and convert images of printed or handwritten text into a machine-readable text format. This involves segmentation of text regions, character recognition, and post-processing to correct errors and enhance accuracy.
Parameter-efficient fine-tuning	A methodology to efficiently fine-tune large pre-trained models without modifying their original parameters. PEFT strategies involve adjusting only a small number of additional model parameters during fine-tuning on a new, smaller training dataset. This significantly reduces the computational and storage costs while achieving comparable performance to full fine-tuning.
Proximal policy optimization	A reinforcement learning algorithm used to train LLMs for alignment by optimizing their behavior through interaction with an environment, balancing exploration and exploitation. PPO achieves this by adjusting the policy parameters in a way that keeps changes within a predefined safe range to maintain stability and improve learning efficiency. This method helps ensure that the LLM aligns its outputs with desired goals, such as ethical guidelines or user intentions, by iteratively refining its responses based on feedback. PPO is often used as part of RLHF.
Reinforcement learning from human feedback	A mechanism to align LLMs with user preferences by fine-tuning with human feedback. Users are asked to rate the quality of a model's response. Based on this, a preference model is trained and then used in a reinforcement learning setup to optimize the generations of the LLM.
Relation extraction	A NLP task that identifies and categorizes the semantic relationships between entities within a text, *e.g.*, materials and properties.
Retrieval augmented generation	A technique for improving the quality of text generation by providing LLMs with access to information retrieved from external knowledge sources. In practice, this means that relevant retrieved text snippets are added to the prompt.
Rotational positional encoding	Rotational positional encoding is a method used in LLMs to represent the position of tokens within a sequence. Instead of using absolute positional values, this technique encodes positional information as rotations in a continuous vector space, which allows the model to understand both the absolute position of tokens and their relative distances.
Self-supervised learning	A machine learning technique that involves generating labels from the input data itself instead of relying on external labeled data. It has been foundational for the success of LLMs, as their pre-training task (next word prediction or filling in of masked words) is a self-supervised task.
Supervised fine-tuning	The process in which a pre-trained LLM is fine-tuned on a labeled dataset of a specific task. It involves adapting the parameters of the pre-trained model to improve its performance on the new task, leveraging the knowledge and representations learned from the initial training.

Table (*continued*)

Vision language model	A multimodal model that can learn simultaneously from images and texts, generating text outputs.
Visual document understanding	Capability of systems to automatically interpret and analyze the content of visual documents, such as images, scanned papers, or digital documents containing complex layouts and graphics. It involves employing computer vision techniques to extract textual information, recognize structural elements like tables and diagrams, and comprehend the overall content.

Acronyms

ANN	Artificial neural network
API	Application programming interface
BFS	Breadth-first search
CLI	Command-line interface
CoT	Chain-of-Thought
CoVe	Chain-of-Verification
DFS	Depth-first search
EAE	Event argument extraction
FN	False negative
FP	False positive
HTML	Hypertext markup language
IE	Information extraction
LLM	Large language model
LM	Language model
LoRA	Low-rank adaptation
MAE	Mean absolute error
ML	Machine learning
MLM	Masked language modeling
MOF	Metal–organic framework
NER	Named entity recognition
NERRE	Name entity recognition and relation extraction
NLP	Natural language processing
NMR	Nuclear magnetic resonance
NN	Neural network
NSP	Next sentence prediction
OCR	Optical character recognition
PEFT	Parameter-efficient fine-tuning
PPO	Proximal policy optimization
RAG	Retrieval augmented generation
RE	Relation extraction
RLHF	Reinforcement learning from human feedback
RNN	Recurrent neural network
RoPE	Rotational positional encoding
SFT	Supervised fine-tuning
SSL	Self-supervised learning
TDM	Text and data mining
TN	True negative
TP	True positive
VDU	Visual document understanding
VLM	Vision language model
XML	Extensible markup language

Data availability

The online book is available at **https://matextract.pub** for which the source code is available at **https://github.com/lamalab-org/matextract-book** and archived at **https://doi.org/10.5281/zenodo.14249541**.

Conflicts of interest

There are no conflicts to declare.

Acknowledgements

The research of M. S.-W. and K. M. J. was supported by the Carl-Zeiss Foundation as well as Intel and Merck *via* the AWASES programme. S. S., C. T. K., J. A. M., and K. M. J. are part of the NFDI consortium FAIRmat funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – project 460197019. M. R. G. and M. V. G. acknowledge financial support from the Spanish Agencia Estatal de Investigación (AEI) through grants TED2021-131693B-I00 and CNS2022-135474, funded by MICIU/AEI/10.13039/501100011033 and by the European Union NextGenerationEU/PRTR, and support from the Spanish National Research Council (CSIC) through Programme for internationalization i-LINK 2023 (Project ILINK23047). We thank Adrian Mirza and Maximilian Greiner for their feedback on a draft of this article.

References

1 M. F. Ashby, *Materials Selection in Mechanical Design*, Butterworth-Heinemann, Oxford, England, 2nd edn, 1999.
2 F. Abild-Pedersen, J. Greeley, F. Studt, J. Rossmeisl, T. R. Munter, P. G. Moses, E. Skulason, T. Bligaard and J. K. Nørskov, *Phys. Rev. Lett.*, 2007, **99**, 016105.
3 K. T. Butler, D. W. Davies, H. Cartwright, O. Isayev and A. Walsh, *Nature*, 2018, **559**, 547–555.
4 K. M. Jablonka, D. Ongari, S. M. Moosavi and B. Smit, *Chem. Rev.*, 2020, **120**, 8066–8129.
5 R. Ramprasad, R. Batra, G. Pilania, A. Mannodi-Kanakkithodi and C. Kim, *npj Comput. Mater.*, 2017, **3**, 54.
6 K. Choudhary, B. DeCost, C. Chen, A. Jain, F. Tavazza, R. Cohn, C. W. Park, A. Choudhary, A. Agrawal, S. J. L. Billinge, E. Holm, S. P. Ong and C. Wolverton, *npj Comput. Mater.*, 2022, **8**, 59.
7 P. Raccuglia, K. C. Elbert, P. D. Adler, C. Falk, M. B. Wenny, A. Mollo, M. Zeller, S. A. Friedler, J. Schrier and A. J. Norquist, *Nature*, 2016, **533**, 73–76.
8 B. Sanchez-Lengeling and A. Aspuru-Guzik, *Science*, 2018, **361**, 360–365.
9 O. Kononova, T. He, H. Huo, A. Trewartha, E. A. Olivetti and G. Ceder, *iScience*, 2021, **24**, 102155.
10 M. Krallinger, O. Rabal, A. Lourenço, J. Oyarzabal and A. Valencia, *Chem. Rev.*, 2017, **117**, 7673–7761.
11 J. Mavračić, C. J. Court, T. Isazawa, S. R. Elliott and J. M. Cole, *J. Chem. Inf. Model.*, 2021, **61**, 4280–4289.

12 D. M. Jessop, S. E. Adams, E. L. Willighagen, L. Hawizy and P. Murray-Rust, *J. Cheminf.*, 2011, **3**, 41.
13 D. M. Lowe and R. A. Sayle, *J. Cheminf.*, 2015, **7**, S5.
14 L. Hawizy, D. M. Jessop, N. Adams and P. Murray-Rust, *J. Cheminf.*, 2011, **3**, 17.
15 M. C. Swain and J. M. Cole, *J. Chem. Inf. Model.*, 2016, **56**, 1894–1904.
16 S. H. M. Mehr, M. Craven, A. I. Leonov, G. Keenan and L. Cronin, *Science*, 2020, **370**, 101–108.
17 J. Guo, A. S. Ibanez-Lopez, H. Gao, V. Quach, C. W. Coley, K. F. Jensen and R. Barzilay, *J. Chem. Inf. Model.*, 2021, **62**, 2035–2045.
18 T. Rocktäschel, M. Weidlich and U. Leser, *Bioinformatics*, 2012, **28**, 1633–1640.
19 O. Kononova, H. Huo, T. He, Z. Rong, T. Botari, W. Sun, V. Tshitoyan and G. Ceder, *Sci. Data*, 2019, **6**, 203.
20 S. Huang and J. M. Cole, *Chem. Sci.*, 2022, **13**, 11487–11495.
21 P. Shetty, A. C. Rajan, C. Kuenneth, S. Gupta, L. P. Panchumarti, L. Holm, C. Zhang and R. Ramprasad, *npj Comput. Mater.*, 2023, **9**, 52.
22 K. Hira, M. Zaki, D. Sheth, Mausam and N. M. A. Krishnan, *Digital Discovery*, 2024, **3**, 1021–1037.
23 C. Ré, *AI trends that I unironically love*, 2021, https://cs.stanford.edu/people/chrismre/papers/SIGMOD-Chris-Re-DataCentric-Foundation-Models-KeyNote.pdf.
24 C. L. Borgman, M. S. Goshen, A. E. Sands, J. C. Wallis, R. L. Cummings, P. T. Darch and B. M. Randles, *Int. J. Digital Curation*, 2016, **11**, 128–149.
25 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, *arXiv*, 2023, preprint, arXiv:1706.03762, DOI: 10.48550/arXiv.1706.03762.
26 G. Yenduri, M. Ramalingam, G. C. Selvi, Y. Supriya, G. Srivastava, P. K. R. Maddikunta, G. D. Raj, R. H. Jhaveri, B. Prabadevi, W. Wang, A. V. Vasilakos and T. R. Gadekallu, *arXiv*, 2023, preprint, arXiv:2305.10435, DOI: 10.48550/arXiv.2305.10435.
27 K. Song, X. Tan, T. Qin, J. Lu and T.-Y. Liu, *arXiv*, 2019, preprint, arXiv:1905.02450, DOI: 10.48550/arXiv.1905.02450.
28 T. Wu, L. Luo, Y.-F. Li, S. Pan, T.-T. Vu and G. Haffari, *arXiv*, 2024, preprint, arXiv:2402.01364, DOI: 10.48550/arXiv.2402.01364.
29 K. M. Jablonka, Q. Ai, A. Al-Feghali, S. Badhwar, J. D. Bocarsly, A. M. Bran, S. Bringuier, L. C. Brinson, K. Choudhary, D. Circi, S. Cox, W. A. de Jong, M. L. Evans, N. Gastellu, J. Genzling, M. V. Gil, A. K. Gupta, Z. Hong, A. Imran, S. Kruschwitz, A. Labarre, J. Lála, T. Liu, S. Ma, S. Majumdar, G. W. Merz, N. Moitessier, E. Moubarak, B. Mouriño, B. Pelkie, M. Pieler, M. C. Ramos, B. Ranković, S. G. Rodriques, J. N. Sanders, P. Schwaller, M. Schwarting, J. Shi, B. Smit, B. E. Smith, J. Van Herck, C. Völker, L. Ward, S. Warren, B. Weiser, S. Zhang, X. Zhang, G. A. Zia, A. Scourtas, K. J. Schmidt, I. Foster, A. D. White and B. Blaiszik, *Digital Discovery*, 2023, **2**, 1233–1250.
30 W. Zhang, Q. Wang, X. Kong, J. Xiong, S. Ni, D. Cao, B. Niu, M. Chen, Y. Li, R. Zhang, Y. Wang, L. Zhang, X. Li, Z. Xiong, Q. Shi, Z. Huang, Z. Fu and M. Zheng, *Chem. Sci.*, 2024, **15**, 10600–10611.
31 N. Smith, X. Yuan, C. Melissinos and G. Moghe, *bioRxiv*, 2024, preprint, DOI: 10.1101/2024.07.22.604620.
32 J. Li, D. Zhang, X. Wang, Z. Hao, J. Lei, Q. Tan, C. Zhou, W. Liu, Y. Yang, X. Xiong, W. Wang, Z. Chen, W. Wang, W. Li, S. Zhang, M. Su, W. Ouyang, Y. Li and D. Zhou, *arXiv*, 2024, preprint, arXiv:2408.07246, DOI: 10.48550/arXiv.2408.07246.
33 K. Chen, H. Cao, J. Li, Y. Du, M. Guo, X. Zeng, L. Li, J. Qiu, P. A. Heng and G. Chen, *arXiv*, 2024, preprint, arXiv:2402.12993, DOI: 10.48550/arxiv.2402.12993.
34 J. Dagdelen, A. Dunn, S. Lee, N. Walker, A. S. Rosen, G. Ceder, K. A. Persson and A. Jain, *Nat. Commun.*, 2024, **15**, 1418.
35 J. Choi and B. Lee, *Commun. Mater.*, 2024, **5**, 13.
36 G. Lei, R. Docherty and S. J. Cooper, *Digital Discovery*, 2024, 1257–1272.
37 M. P. Polak and D. Morgan, *Nat. Commun.*, 2024, **15**, 1569.
38 Y. Ye, J. Ren, S. Wang, Y. Wan, H. Wang, I. Razzak, T. Xie and W. Zhang, *arXiv*, 2024, preprint, arXiv:2404.03080, DOI: 10.48550/arXiv.2404.03080.
39 M. P. Polak, S. Modi, A. Latosinska, J. Zhang, C.-W. Wang, S. Wang, A. D. Hazra and D. Morgan, *Digital Discovery*, 2024, **3**, 1221–1235.
40 M. Suvarna, A. C. Vaucher, S. Mitchell, T. Laino and J. Pérez-Ramírez, *Nat. Commun.*, 2023, **14**, 7964.
41 G. Khalighinejad, D. Circi, L. C. Brinson and B. Dhingra, *arXiv*, 2024, preprint, arXiv:2403.00260, DOI: 10.48550/arXiv.2403.00260.
42 E. B. Community, *Jupyter Book*, 2020, https://zenodo.org/record/4539666.
43 M. Schilling-Wilhelmi, M. Ríos-García, S. Shabih, M. V. Gil, S. Miret, C. T. Koch, J. A. Márquez and K. M. Jablonka, *matextract*, 2014, https://github.com/lamalab-org/matextract-book.
44 A. K. Singh and D. Strouse, *arXiv*, 2024, preprint, arXiv:2402.14903, DOI: 10.48550/arXiv.2402.14903.
45 A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, *OpenAI blog*, 2018.
46 A. Radford, J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, *et al.*, *OpenAI blog*, 2019, **1**, 9.
47 H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave and G. Lample, *arXiv*, 2023, preprint, arXiv:2302.13971, DOI: 10.48550/arXiv.2302.13971.
48 N. Dziri, X. Lu, M. Sclar, X. L. Li, L. Jiang, B. Y. Lin, P. West, C. Bhagavatula, R. L. Bras, J. D. Hwang, S. Sanyal, S. Welleck, X. Ren, A. Ettinger, Z. Harchaoui and Y. Choi, *arXiv*, 2023, preprint, arXiv:2305.18654, DOI: 10.48550/arXiv.2305.18654.
49 C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. Yu, S. Zhang, G. Ghosh, M. Lewis, L. Zettlemoyer and O. Levy, *arXiv*, 2023, preprint, arXiv:2305.11206, DOI: 10.48550/arXiv.2305.11206.

50 L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike and R. Lowe, *arXiv*, 2022, preprint, arXiv:2203.02155, DOI: **10.48550/arXiv.2203.02155**.

51 Q. Lyu, K. Shridhar, C. Malaviya, L. Zhang, Y. Elazar, N. Tandon, M. Apidianaki, M. Sachan and C. Callison-Burch, *arXiv*, 2024, preprint, arXiv:2402.13904, DOI: **10.48550/arXiv.2402.13904**.

52 *Authors Guild, Inc. v. HathiTrust*, 902 F. Supp. 2d 445, United States District Court for the Southern District of New York, 2012.

53 J. Molloy, M. Haeussler, P. Murray-Rust and C. Oppenheim, *Working with Text*, Elsevier, 2016, pp. 89–109.

54 S. M. Fiil-Flynn, B. Butler, M. Carroll, O. Cohen-Sasson, C. Craig, L. Guibault, P. Jaszi, B. J. Jütte, A. Katz, J. P. Quintais, T. Margoni, A. R. de Souza, M. Sag, R. Samberg, L. Schirru, M. Senftleben, O. Tur-Sinai and J. L. Contreras, *Science*, 2022, **378**, 951–953.

55 EMBL's European Bioinformatics Institute, *About Europe PMC*, 2024, **https://europepmc.org/About**.

56 American Chemical Society (ACS), Chinese Chemical Society (CCS), Chemical Society of Japan (CSJ), German Chemical Society (GDCh) and Royal Society of Chemistry (RSC), *ChemRxiv*, 2024, **https://chemrxiv.org/engage/chemrxiv/public-dashboard**.

57 Cornell University, *arXiv*, 2024, **https://arxiv.org/**.

58 Allen Institute for AI, *The Semantic Scholar Open Research Corpus (S2ORC)*, 2019, **https://allenai.org/data/s2orc**.

59 Elsevier, *Elsevier OA CC-BY Corpus*, 2020, **https://researchcollaborations.elsevier.com/en/datasets/elsevier-oa-cc-by-corpus**.

60 P. Baldi, *J. Chem. Inf. Model.*, 2011, **51**, 3029.

61 Open Reaction Database Project Authors, *Open Reaction Database*, **https://open-reaction-database.org**, 2021.

62 S. M. Kearnes, M. R. Maser, M. Wleklinski, A. Kast, A. G. Doyle, S. D. Dreher, J. M. Hawkins, K. F. Jensen and C. W. Coley, *J. Am. Chem. Soc.*, 2021, **143**, 18820–18826.

63 R. Lammey, *Sci. Ed.*, 2015, **2**, 22–27.

64 Elsevier B.V., *Elsevier Developer Portal*, 2024, **https://dev.elsevier.com**.

65 MasterAI-EAM, SciCrawler GitHub Repository, 2023, **https://github.com/MasterAI-EAM/SciCrawler**.

66 P. Murray, *Pygetpapers GitHub Repository*, 2023, **https://github.com/petermr/pygetpapers**.

67 A. Team, *Argilla*, 2022, Software available from **https://github.com/argilla-io/argilla**.

68 H. Nakayama, T. Kubo, J. Kamura, Y. Taniguchi and X. Liang, *doccano: Text Annotation Tool for Human*, 2018, Software available from **https://github.com/doccano/doccano**.

69 S. Kapoor and A. Narayanan, *Patterns*, 2023, **4**, 100804.

70 M. Zhu and J. M. Cole, *J. Chem. Inf. Model.*, 2022, **62**, 1633–1643.

71 R. Smith, Ninth international conference on document analysis and recognition (ICDAR 2007), 2007, pp. 629–633.

72 L. Blecher, G. Cucurull, T. Scialom and R. Stojnic, *arXiv*, 2023, preprint, arXiv:2308.13418, DOI: **10.48550/arXiv.2308.13418**.

73 V. Paruchuri, *Marker: Open Source Machine Learning Model for Data Annotation*, 2023, **https://github.com/VikParuchuri/marker**.

74 N. Meuschke, A. Jagdale, T. Spinde, J. Mitrović and B. Gipp, *Information for a Better World: Normality, Virtuality, Physicality, Inclusivity*, Springer Nature, Switzerland, 2023, pp. 383–405.

75 A. M. Bran, Z. Jončev and P. Schwaller, Proceedings of the 1st Workshop on Language + Molecules (L+M 2024), Association for Computational Linguistics, 2024, pp. 74–84.

76 U.S. National Institute of Standards and Technology (NIST), *ChemNLP: Chemical Natural Language Processing Toolkit*, 2023, **https://github.com/usnistgov/chemnlp**.

77 M. J. Buehler, *arXiv*, 2024, preprint, arXiv:2403.11996, DOI: **10.48550/arXiv.2403.11996**.

78 H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes and A. Mian, *arXiv*, 2024, preprint, arXiv:2307.06435, DOI: **10.48550/arXiv.2307.06435**.

79 M. D. Skarlinski, S. Cox, J. M. Laurent, J. D. Braza, M. Hinks, M. J. Hammerling, M. Ponnapati, S. G. Rodriques and A. D. White, *arXiv*, 2024, preprint, arXiv:2409.13740, DOI: **10.48550/arXiv.2409.13740**.

80 S. Carta, A. Giuliani, L. Piano, A. S. Podda, L. Pompianu and S. G. Tiddia, *arXiv*, 2023, preprint, arXiv:2307.01128, DOI: **10.48550/arXiv.2307.01128**.

81 P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-T. Yih, T. Rocktäschel, S. Riedel and D. Kiela, *arXiv*, 2021, preprint, arXiv:2005.11401, DOI: **10.48550/arXiv.2005.11401**.

82 Z. Zheng, O. Zhang, C. Borgs, J. T. Chayes and O. M. Yaghi, *J. Am. Chem. Soc.*, 2023, **145**, 18048–18062.

83 Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le and R. Salakhutdinov, *arXiv*, 2019, preprint, arXiv:1901.02860, DOI: **10.48550/arXiv.1901.02860**.

84 I. Beltagy, M. E. Peters and A. Cohan, *arXiv*, 2020, preprint, arXiv:2004.05150, DOI: **10.48550/arXiv.2004.05150**.

85 J. Su, Y. Lu, S. Pan, A. Murtadha, B. Wen and Y. Liu, *arXiv*, 2023, preprint, arXiv:2104.09864, DOI: **10.48550/arXiv.2104.09864**.

86 S. Chen, S. Wong, L. Chen and Y. Tian, *arXiv*, 2023, preprint, arXiv:2306.15595, DOI: **10.48550/arXiv.2306.15595**.

87 W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, H. Zhang, B. Zhu, M. Jordan, J. E. Gonzalez and I. Stoica, *arXiv*, 2024, preprint, arXiv:2403.04132, DOI: **10.48550/arXiv.2403.04132**.

88 A. Mirza, N. Alampara, S. Kunchapu, B. Emoekabu, A. Krishnan, M. Wilhelmi, M. Okereke, J. Eberhardt, A. M. Elahi, M. Greiner, C. T. Holick, T. Gupta, M. Asgari, C. Glaubitz, L. C. Klepsch, Y. Köster, J. Meyer, S. Miret, T. Hoffmann, F. A. Kreth, M. Ringleb, N. Roesner, U. S. Schubert, L. M. Stafast, D. Wonanke, M. Pieler, P. Schwaller and K. M. Jablonka, *arXiv*, 2024, preprint, arXiv:2404.01475, DOI: **10.48550/arXiv.2404.01475**.

89 T. Xie, Y. Wan, Y. Zhou, W. Huang, Y. Liu, Q. Linghu, S. Wang, C. Kit, C. Grazian, W. Zhang and B. Hoex, *Patterns*, 2024, **5**, 100955.

90 J. M. Laurent, J. D. Janizek, M. Ruzo, M. M. Hinks, M. J. Hammerling, S. Narayanan, M. Ponnapati, A. D. White and S. G. Rodriques, *arXiv*, 2024, preprint, arXiv:2407.10362, DOI: 10.48550/arXiv.2407.10362.

91 V. Udandarao, A. Prabhu, A. Ghosh, Y. Sharma, P. H. S. Torr, A. Bibi, S. Albanie and M. Bethge, *arXiv*, 2024, preprint, arXiv:2404.04125, DOI: 10.48550/arXiv.2404.04125.

92 C. M. Castro Nascimento and A. S. Pimentel, *J. Chem. Inf. Model.*, 2023, **63**, 1649–1655.

93 A. D. White, G. M. Hocky, H. A. Gandhi, M. Ansari, S. Cox, G. P. Wellawatte, S. Sasmal, Z. Yang, K. Liu, Y. Singh and W. J. Peña Ccoa, *Digital Discovery*, 2023, **2**, 368–376.

94 T. Xie, Y. Wan, W. Huang, Z. Yin, Y. Liu, S. Wang, Q. Linghu, C. Kit, C. Grazian, W. Zhang, I. Razzak and B. Hoex, *arXiv*, 2023, preprint, arXiv:2308.13565, DOI: 10.48550/arXiv.2308.13565.

95 Meta, *Introducing Meta Llama 3: The most capable openly available LLM to date*, https://ai.meta.com/blog/meta-llama-3/.

96 Meta Llama Team, *Meta AI blog*, 2024, https://ai.meta.com/research/publications/the-llama-3-herd-of-models/.

97 Y. Hu, Q. Chen, J. Du, X. Peng, V. K. Keloth, X. Zuo, Y. Zhou, Z. Li, X. Jiang, Z. Lu, K. Roberts and H. Xu, *arXiv*, 2024, preprint, arXiv:2303.16416, DOI: 10.48550/arXiv.2303.16416.

98 J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le and D. Zhou, *arXiv*, 2023, preprint, arXiv:2201.11903, DOI: 10.48550/arXiv.2201.11903.

99 V. K. Kommineni, B. König-Ries and S. Samuel, *arXiv*, 2024, preprint, arXiv:2403.08345, DOI: 10.48550/arXiv.2403.08345.

100 N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan and S. Yao, *arXiv*, 2023, preprint, arXiv:2303.11366, DOI: 10.48550/arXiv.2303.11366.

101 T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, *arXiv*, 2020, preprint, arXiv:2005.14165, DOI: 10.48550/arXiv.2005.14165.

102 J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu and D. Amodei, *arXiv*, 2020, preprint, arXiv:2001.08361, DOI: 10.48550/arXiv.2001.08361.

103 A. Goel, A. Gueta, O. Gilon, C. Liu, S. Erell, L. H. Nguyen, X. Hao, B. Jaber, S. Reddy, R. Kartha, J. Steiner, I. Laish and A. Feder, in *Proceedings of the 3rd Machine Learning for Health Symposium*, ed. S. Hegselmann, A. Parziale, D. Shanmugam, S. Tang, M. N. Asiedu, S. Chang, T. Hartvigsen and H. Singh, PMLR, 2023, vol. 225, pp. 82–100.

104 X. Xu, Y. Zhu, X. Wang and N. Zhang, *arXiv*, 2023, preprint, arXiv:2305.01555, DOI: 10.48550/arXiv.2305.01555.

105 Y. Lu, M. Bartolo, A. Moore, S. Riedel and P. Stenetorp, *arXiv*, 2022, preprint, arXiv:2104.08786, DOI: 10.48550/arXiv.2104.08786.

106 M. Agrawal, S. Hegselmann, H. Lang, Y. Kim and D. Sontag, *arXiv*, 2022, preprint, arXiv:2205.12689, DOI: 10.48550/arXiv.2205.12689.

107 J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin and W. Chen, *arXiv*, 2021, preprint, arXiv:2101.06804, DOI: 10.48550/arXiv.2101.06804.

108 Y. Sui, M. Zhou, M. Zhou, S. Han and D. Zhang, *arXiv*, 2024, preprint, arXiv:2305.13062, DOI: 10.48550/arXiv.2305.13062.

109 R. Agarwal, A. Singh, L. M. Zhang, B. Bohnet, L. Rosias, S. Chan, B. Zhang, A. Anand, Z. Abbas, A. Nova, J. D. Co-Reyes, E. Chu, F. Behbahani, A. Faust and H. Larochelle, *arXiv*, 2024, preprint, arXiv:2404.11018, DOI: 10.48550/arXiv.2404.11018.

110 K. Stechly, K. Valmeekam and S. Kambhampati, *arXiv*, 2024, preprint, arXiv:2405.04776, DOI: 10.48550/arXiv.2405.04776.

111 T. Ridnik, D. Kredo and I. Friedman, *arXiv*, 2024, preprint, arXiv:2401.08500, DOI: 10.48550/arXiv.2401.08500.

112 H. Chase, *LangChain*, 2022, https://github.com/langchain-ai/langchain.

113 J. Liu, *LlamaIndex*, 2022, https://github.com/jerryjliu/llama_index.

114 M. Sclar, Y. Choi, Y. Tsvetkov and A. Suhr, *arXiv*, 2023, preprint, arXiv:2310.11324, DOI: 10.48550/arXiv.2310.11324.

115 O. Khattab, A. Singhvi, P. Maheshwari, Z. Zhang, K. Santhanam, S. Vardhamanan, S. Haq, A. Sharma, T. T. Joshi, H. Moazam, H. Miller, M. Zaharia and C. Potts, *arXiv*, 2023, preprint, arXiv:2310.03714, DOI: 10.48550/arXiv.2310.03714.

116 L. Patiny and G. Godin, *ChemRxiv*, 2023, preprint, DOI: 10.26434/chemrxiv-2023-05v1b-v2.

117 C. Xia, C. Xing, J. Du, X. Yang, Y. Feng, R. Xu, W. Yin and C. Xiong, *arXiv*, 2024, preprint, arXiv:2402.18667, DOI: 10.48550/arXiv.2402.18667.

118 M. Shamsabadi, J. D'Souza and S. Auer, *arXiv*, 2024, preprint, arXiv:2401.10040, DOI: 10.48550/arXiv.2401.10040.

119 R. K. Luu and M. J. Buehler, *Adv. Sci.*, 2023, **11**(10), DOI: 10.1002/advs.202306724.

120 J. van Herck, M. V. Gil, K. M. Jablonka, A. Abrudan, A. Anker, M. Asgari, B. Blaiszik, L. Choudhury, C. Corminboeuf and H. Daglar, *Chem. Sci.*, 2025, DOI: 10.1039/D4SC04401K.

121 S. Kim, Y. Jung and J. Schrier, *J. Am. Chem. Soc.*, 2024, **146**, 19654–19659.

122 K. M. Jablonka, P. Schwaller, A. Ortega-Guerrero and B. Smit, *Nat. Mach. Intell.*, 2024, **6**, 161–169.

123 E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang and W. Chen, *arXiv*, 2021, preprint, arXiv:2106.09685, DOI: 10.48550/arXiv.2106.09685.

124 S. Lee, K. Cruse, S. P. Gleason, A. P. Alivisatos, G. Ceder and A. Jain, *ChemRxiv*, 2024, preprint, DOI: 10.26434/chemrxiv-2024-ncjlp.

125 O. Sainz, I. Garcí-Ferrero, R. Agerri, O. L. de Lacalle, G. Rigau and E. Agirre, *arXiv*, 2024, preprint, arXiv:2310.03668, DOI: **10.48550/arXiv.2310.03668**.
126 Q. Ai, F. Meng, J. Shi, B. Pelkie and C. W. Coley, *ChemRxiv*, 2024, preprint, DOI: **10.26434/chemrxiv-2024-979fz**.
127 R. Zhang, J. Han, C. Liu, P. Gao, A. Zhou, X. Hu, S. Yan, P. Lu, H. Li and Y. Qiao, *arXiv*, 2023, preprint, arXiv:2303.16199, DOI: **10.48550/arXiv.2303.16199**.
128 H. Ivison, Y. Wang, V. Pyatkin, N. Lambert, M. Peters, P. Dasigi, J. Jang, D. Wadden, N. A. Smith, I. Beltagy and H. Hajishirzi, *arXiv*, 2023, preprint, arXiv:2311.10702, DOI: **10.48550/arXiv.2311.10702**.
129 D. Biderman, J. G. Ortiz, J. Portes, M. Paul, P. Greengard, C. Jennings, D. King, S. Havens, V. Chiley, J. Frankle, C. Blakeney and J. P. Cunningham, *arXiv*, 2024, preprint, arXiv:2405.09673, DOI: **10.48550/arXiv.2405.09673**.
130 Y. Zhang, C. Wang, M. Soukaseum, D. G. Vlachos and H. Fang, *J. Chem. Inf. Model.*, 2022, **62**, 3316–3330.
131 I. Beltagy, K. Lo and A. Cohan, *arXiv*, 2019, preprint, arXiv:1903.10676, DOI: **10.48550/arXiv.1903.10676**.
132 T. Gupta, M. Zaki, N. M. A. Krishnan and Mausam, *npj Comput. Mater.*, 2022, **8**, 102.
133 J. Zhao, S. Huang and J. M. Cole, *J. Chem. Inf. Model.*, 2023, **63**, 1961–1981.
134 T. Isazawa and J. M. Cole, *J. Chem. Inf. Model.*, 2024, **64**, 3205–3212.
135 P. Rajpurkar, R. Jia and P. Liang, *arXiv*, 2018, preprint, arXiv:1806.03822, DOI: **10.48550/arXiv.1806.03822**.
136 Y. Song, S. Miret, H. Zhang and B. Liu, *arXiv*, 2023, preprint, arXiv:2310.08511, DOI: **10.48550/arXiv.2310.08511**.
137 Y. Song, S. Miret and B. Liu, *arXiv*, 2023, preprint, arXiv:2305.08264, DOI: **10.48550/arXiv.2305.08264**.
138 D. Circi, G. Khalighinejad, B. Dhingra and L. C. Brinson, Proceedings of the SouthNLP 2024 Conference, USA, 2024.
139 L. Weng, *Generalized Visual Language Models*, 2022, **https://lilianweng.github.io/posts/2022-06-09-vlm/**.
140 Z. Zheng, Z. He, O. Khattab, N. Rampal, M. A. Zaharia, C. Borgs, J. T. Chayes and O. M. Yaghi, *Digital Discovery*, 2024, **3**, 491–501.
141 M. Schilling-Wilhelmi and K. M. Jablonka, *arXiv*, AI4Mat-Vienna-2024, 2024, **https://openreview.net/pdf?id=zlutCyZ12H**.
142 S. X. Leong, S. Pablo-García, Z. Zhang and A. Aspuru-Guzik, *Chem. Sci.*, 2024, **15**, 17881–17891.
143 H. Lu, W. Liu, B. Zhang, B. Wang, K. Dong, B. Liu, J. Sun, T. Ren, Z. Li, H. Yang, Y. Sun, C. Deng, H. Xu, Z. Xie and C. Ruan, *arXiv*, 2024, preprint, arXiv:2403.05525, DOI: **10.48550/arXiv.2403.05525**.
144 Y. Liu, Z. Li, B. Yang, C. Li, X. Yin, C. L. Liu, L. Jin and X. Bai, *arXiv*, 2024, preprint, arXiv:2305.07895, DOI: **10.48550/arXiv.2305.07895**.
145 N. Alampara, M. Schilling-Wilhelmi, M. Ríos-García, I. Mandal, P. Khetarpal, H. S. Grover, N. M. A. Krishnan and K. M. Jablonka, *arXiv*, 2024, preprint, arXiv:2411.16955, DOI: **10.48550/arXiv.2411.16955**.
146 LlamaIndex, *Multi-modal PDF Tables Example*, 2023, **https://docs.llamaindex.ai/en/v0.10.17/examples/multi_modal/multi_modal_pdf_tables.html**.
147 S. Lee, S. Heinen, D. Khan and O. Anatole von Lilienfeld, *Mach. Learn.: Sci. Technol.*, 2024, **5**, 015052.
148 B. Smock and R. Pesala, *Table Transformer*, version 1.0.0, 2021, **https://github.com/microsoft/table-transformer**.
149 T. Gupta, M. Zaki, D. Khatsuriya, K. Hira, N. M. A. Krishnan and Mausam, *arXiv*, 2024, preprint, arXiv:2207.01079, DOI: **10.48550/arXiv.2207.01079**.
150 F. Liu, J. M. Eisenschlos, F. Piccinno, S. Krichene, C. Pang, K. Lee, M. Joshi, W. Chen, N. Collier and Y. Altun, *arXiv*, 2023, preprint, arXiv:2212.10505, DOI: **10.48550/arXiv.2212.10505**.
151 D. M. Wilary and J. M. Cole, *J. Chem. Inf. Model.*, 2023, **63**, 6053–6067.
152 V. Fan, Y. Qian, A. Wang, A. Wang, C. W. Coley and R. Barzilay, *arXiv*, 2024, preprint, arXiv:2404.01462, DOI: **10.48550/ARXIV.2404.01462**.
153 K. Rajan, H. O. Brinkhaus, A. Zielesny and C. Steinbeck, *J. Cheminf.*, 2020, **12**, 60.
154 automeris-io, *WebPlotDigitizer: A Web-Based Tool to Extract Data from Plots, Images, and Maps*, 2023, **https://github.com/automeris-io/WebPlotDigitizer**.
155 M. Zaki, Jayadeva and N. A. Krishnan, *Chem. Eng. Process.*, 2022, **180**, 108607.
156 K. T. Mukaddem, E. J. Beard, B. Yildirim and J. M. Cole, *J. Chem. Inf. Model.*, 2019, **60**, 2492–2509.
157 L. von Chamier, R. F. Laine, J. Jukkala, C. Spahn, D. Krentzel, E. Nehme, M. Lerche, S. Hernández-Pérez, P. K. Mattila, E. Karinou, S. Holden, A. C. Solak, A. Krull, T.-O. Buchholz, M. L. Jones, L. A. Royer, C. Leterrier, Y. Shechtman, F. Jug, M. Heilemann, G. Jacquemet and R. Henriques, *Nat. Commun.*, 2021, **12**, 2276.
158 J. Stuckner, B. Harder and T. M. Smith, *npj Comput. Mater.*, 2022, **8**, 200.
159 S. Gao, A. Fang, Y. Huang, V. Giunchiglia, A. Noori, J. R. Schwarz, Y. Ektefaie, J. Kondic and M. Zitnik, *arXiv*, 2024, preprint, arXiv:2404.02831, DOI: **10.48550/arXiv.2404.02831**.
160 L. Weng, *LLM-powered Autonomous Agents*, 2023, **https://lilianweng.github.io/posts/2023-06-23-agent/**.
161 J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang and M. S. Bernstein, Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology, 2023, pp. 1–22.
162 G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. Fan and A. Anandkumar, *Intrinsically-Motivated and Open-Ended Learning Workshop*, NeurIPS, 2023.
163 H. Liu, Y. Li, J. Jian, Y. Cheng, J. Lu, S. Guo, J. Zhu, M. Zhang, M. Zhang and H. Wang, *arXiv*, 2024, preprint, arXiv:2402.12391, DOI: **10.48550/arXiv.2402.12391**.
164 A. Ghafarollahi and M. J. Buehler, *arXiv*, 2024, preprint, arXiv:2402.04268, DOI: **10.48550/arXiv.2402.04268**.
165 A. Bou, M. Thomas, S. Dittert, C. N. Ramírez, M. Majewski, Y. Wang, S. Patel, G. Tresadern, M. Ahmad, V. Moens, W. Sherman, S. Sciabola and G. D. Fabritiis,

arXiv, 2024, preprint, arXiv:2405.04657, DOI: 10.48550/arXiv.2405.04657.

166 A. M. Bran, S. Cox, O. Schilter, C. Baldassari, A. D. White and P. Schwaller, *Nat. Mach. Intell.*, 2024, **6**, 525–535.

167 D. A. Boiko, R. MacKnight, B. Kline and G. Gomes, *Nature*, 2023, **624**, 570–578.

168 T. Masterman, S. Besen, M. Sawtell and A. Chao, *arXiv*, 2024, preprint, arXiv:2404.11584, DOI: 10.48550/arXiv.2404.11584.

169 Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou, R. Zheng, X. Fan, X. Wang, L. Xiong, Y. Zhou, W. Wang, C. Jiang, Y. Zou, X. Liu, Z. Yin, S. Dou, R. Weng, W. Cheng, Q. Zhang, W. Qin, Y. Zheng, X. Qiu, X. Huang and T. Gui, *arXiv*, 2023, preprint, arXiv:2309.07864, DOI: 10.48550/arXiv.2309.07864.

170 M. Caldas Ramos, C. Collison and A. D. White, *Chem. Sci.*, 2024, DOI: 10.1039/D4SC03921A.

171 H. Liu, C. Li, Q. Wu and Y. J. Lee, *arXiv*, 2023, preprint, arXiv:2304.08485, DOI: 10.48550/arXiv.2304.08485.

172 M. Ansari and S. M. Moosavi, *Digital Discovery*, 2024, **3**, 2607–2617.

173 Y. Du, S. Li, A. Torralba, J. B. Tenenbaum and I. Mordatch, *arXiv*, 2023, preprint, arXiv:2305.14325, DOI: 10.48550/arXiv.2305.14325.

174 J. Lála, O. O'Donoghue, A. Shtedritski, S. Cox, S. G. Rodriques and A. D. White, *arXiv*, 2023, preprint, arXiv:2312.07559, DOI: 10.48550/arXiv.2312.07559.

175 L. Wang, C. Ma, X. Feng, Z. Zhang, H. Yang, J. Zhang, Z. Chen, J. Tang, X. Chen, Y. Lin, W. X. Zhao, Z. Wei and J. Wen, *Front. Comput. Sci.*, 2024, **18**, 186345.

176 T. R. Sumers, S. Yao, K. Narasimhan and T. L. Griffiths, *arXiv*, 2024, preprint, arXiv:2309.02427, DOI: 10.48550/arXiv.2309.02427.

177 S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao and K. Narasimhan, *arXiv*, 2023, preprint, arXiv:2305.10601, DOI: 10.48550/arXiv.2305.10601.

178 A. Madaan, N. Tandon, P. Gupta, S. Hallinan, L. Gao, S. Wiegreffe, U. Alon, N. Dziri, S. Prabhumoye, Y. Yang, S. Gupta, B. P. Majumder, K. Hermann, S. Welleck, A. Yazdanbakhsh and P. Clark, *arXiv*, 2023, preprint, arXiv:2303.17651, DOI: 10.48550/arXiv.2303.17651.

179 Z. Gou, Z. Shao, Y. Gong, Y. Shen, Y. Yang, N. Duan and W. Chen, *arXiv*, 2024, preprint, arXiv:2305.11738, DOI: 10.48550/arXiv.2305.11738.

180 W. Wang, L. Dong, H. Cheng, X. Liu, X. Yan, J. Gao and F. Wei, *arXiv*, 2023, preprint, arXiv:2306.07174, DOI: 10.48550/arXiv.2306.07174.

181 Y. Shen, K. Song, X. Tan, D. Li, W. Lu and Y. Zhuang, *arXiv*, 2023, preprint, arXiv:2303.17580, DOI: 10.48550/arXiv.2303.17580.

182 S. G. Patil, T. Zhang, X. Wang and J. E. Gonzalez, *arXiv*, 2023, preprint, arXiv:2305.15334, DOI: 10.48550/arXiv.2305.15334.

183 S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan and Y. Cao, *arXiv*, 2023, preprint, arXiv:2210.03629, DOI: 10.48550/arXiv.2210.03629.

184 C. Qian, W. Liu, H. Liu, N. Chen, Y. Dang, J. Li, C. Yang, W. Chen, Y. Su, X. Cong, J. Xu, D. Li, Z. Liu and M. Sun, *arXiv*, 2024, preprint, arXiv:2307.07924, DOI: 10.48550/arXiv.2307.07924.

185 Y. Talebirad and A. Nadiri, *arXiv*, 2023, preprint, arXiv:2306.03314, DOI: 10.48550/arXiv.2306.03314.

186 Y. Song, W. Xiong, D. Zhu, W. Wu, H. Qian, M. Song, H. Huang, C. Li, K. Wang, R. Yao, Y. Tian and S. Li, *arXiv*, 2023, preprint, arXiv:2306.06624, DOI: 10.48550/arXiv.2306.06624.

187 Y. Zhuang, Y. Yu, K. Wang, H. Sun and C. Zhang, *arXiv*, 2023, preprint, arXiv:2306.13304, DOI: 10.48550/arXiv.2306.13304.

188 J. Andreas, *arXiv*, 2022, preprint, arXiv:2212.01681, DOI: 10.48550/arXiv.2212.01681.

189 F. Shi, X. Chen, K. Misra, N. Scales, D. Dohan, E. Chi, N. Schärli and D. Zhou, *arXiv*, 2023, preprint, arXiv:2302.00093, DOI: 10.48550/arXiv.2302.00093.

190 X. Huang, W. Liu, X. Chen, X. Wang, H. Wang, D. Lian, Y. Wang, R. Tang and E. Chen, *arXiv*, 2024, preprint, arXiv:2402.02716, DOI: 10.48550/arXiv.2402.02716.

191 T. Cai, X. Wang, T. Ma, X. Chen and D. Zhou, *arXiv*, 2024, preprint, arXiv:2305.17126, DOI: 10.48550/arXiv.2305.17126.

192 C. Qian, C. Han, Y. R. Fung, Y. Qin, Z. Liu and H. Ji, *arXiv*, 2024, preprint, arXiv:2305.14318, DOI: 10.48550/arXiv.2305.14318.

193 L. Yuan, Y. Chen, X. Wang, Y. R. Fung, H. Peng and H. Ji, *arXiv*, 2024, preprint, arXiv:2309.17428, DOI: 10.48550/arXiv.2309.17428.

194 Y. Ruan, H. Dong, A. Wang, S. Pitis, Y. Zhou, J. Ba, Y. Dubois, C. J. Maddison and T. Hashimoto, *arXiv*, 2024, preprint, arXiv:2309.15817, DOI: 10.48550/arXiv.2309.15817.

195 X. Tang, Q. Jin, K. Zhu, T. Yuan, Y. Zhang, W. Zhou, M. Qu, Y. Zhao, J. Tang, Z. Zhang, A. Cohan, Z. Lu and M. Gerstein, *arXiv*, 2024, preprint, arXiv:2402.04247, DOI: 10.48550/arXiv.2402.04247.

196 S. Kapoor, B. Stroebl, Z. S. Siegel, N. Nadgir and A. Narayanan, *arXiv*, 2024, preprint, arXiv:2407.01502, DOI: 10.48550/arXiv.2407.01502.

197 W. Huang, P. Abbeel, D. Pathak and I. Mordatch, *arXiv*, 2022, preprint, arXiv:2201.07207, DOI: 10.48550/arXiv.2201.07207.

198 J. Puigcerver, C. Riquelme, B. Mustafa and N. Houlsby, *arXiv*, 2023, preprint, arXiv:2308.00951, DOI: 10.48550/arXiv.2308.00951.

199 R. Sengottuvelu, *jsonformer*, https://github.com/1rgs/jsonformer.

200 S. Geng, M. Josifoski, M. Peyrard and R. West, *arXiv*, 2024, preprint, arXiv:2305.13971, DOI: 10.48550/arXiv.2305.13971.

201 D. Deutsch, S. Upadhyay and D. Roth, Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL), 2019.

202 B. T. Willard and R. Louf, *arXiv*, 2023, preprint, arXiv:2307.09702, DOI: 10.48550/arXiv.2307.09702.

203 J. Liu, *jxnl/instructor: structured outputs for llms*, https://github.com/jxnl/instructor/.

204 Prefect, *marvin*, https://github.com/prefecthq/marvin.

205 G. Gerganov, *ggml*, https://github.com/ggerganov/ggml.

206 H. M. Sayeed, T. Mohanty and T. D. Sparks, *Integr. Mater. Manuf. Innov.*, 2024, **13**, 445–452.

207 J. H. Caufield, H. Hegde, V. Emonet, N. L. Harris, M. P. Joachimiak, N. Matentzoglu, H. Kim, S. Moxon, J. T. Reese, M. A. Haendel, P. N. Robinson and C. J. Mungall, *Bioinformatics*, 2024, **40**(3), btae104.

208 R. E. Burkard and U. Derigs, *Assignment and Matching Problems: Solution Methods with FORTRAN-Programs*, Springer Berlin Heidelberg, Berlin, Heidelberg, 1980, pp. 1–15.
209 S. Kim, P. A. Thiessen, E. E. Bolton, J. Chen, G. Fu, A. Gindulyte, L. Han, J. He, S. He, B. A. Shoemaker, J. Wang, B. Yu, J. Zhang and S. H. Bryant, *Nucleic Acids Res.*, 2015, **44**, D1202–D1213.
210 M. Sitzmann, I. Filippov and M. Nicklaus, *SAR QSAR Environ. Res.*, 2008, **19**, 1–9.
211 H. E. Grecco, *Pint: a Python Units Library*, 2014, **https://github.com/hgrecco/pint**.
212 N. J. Goldbaum, J. A. ZuHone, M. J. Turk, K. Kowalik and A. L. Rosen, *J. Open Source Software*, 2018, **3**, 809.
213 N. Alampara, S. Miret and K. M. Jablonka, *arXiv*, 2024, preprint, arXiv:2406.17295, DOI: **10.48550/arXiv.2406.17295**.
214 S. Miret and N. M. A. Krishnan, *arXiv*, 2024, preprint, arXiv:2402.05200, DOI: **10.48550/arXiv.2402.05200**.
215 D. Ongari, L. Talirz, K. M. Jablonka, D. W. Siderius and B. Smit, *J. Chem. Eng. Data*, 2022, **67**, 1743–1756.
216 X. Jia, A. Lynch, Y. Huang, M. Danielson, I. Lang'at, A. Milder, A. E. Ruby, H. Wang, S. A. Friedler and A. J. Norquist, *et al.*, *Nature*, 2019, **573**, 251–255.
217 P. Ball, *Nature*, 2023, **624**, 22–25.
218 Q. Huang, J. Vora, P. Liang and J. Leskovec, *arXiv*, 2024, preprint, arXiv:2310.03302, DOI: **10.48550/arXiv.2310.03302**.
219 X. Wang, Z. Hu, P. Lu, Y. Zhu, J. Zhang, S. Subramaniam, A. R. Loomba, S. Zhang, Y. Sun and W. Wang, Forty-first International Conference on Machine Learning, 2024.
220 S. Dhuliawala, M. Komeili, J. Xu, R. Raileanu, X. Li, A. Celikyilmaz and J. Weston, *arXiv*, 2023, preprint, arXiv:2309.11495, DOI: **10.48550/arXiv.2309.11495**.
221 S. Minaee, T. Mikolov, N. Nikzad, M. Chenaghlu, R. Socher, X. Amatriain and J. Gao, *arXiv*, 2024, preprint, arXiv:2402.06196, DOI: **10.48550/arXiv.2402.06196**.