Digital Discovery

ROYAL SOCIETY OF CHEMISTRY

PAPER

View Article Online
View Journal | View Issue



Cite this: *Digital Discovery*, 2025, **4**, 3137

# MARCUS: molecular annotation and recognition for curating unravelled structures

Kohulan Rajan, [a] Viktor Weißenborn,[a] Laurin Lederer,[a] Achim Zielesny [b] and Christoph Steinbeck *[a]

The exponential growth of chemical literature necessitates the development of automated tools for extracting and curating molecular information from unstructured scientific publications into open-access chemical databases. Current optical chemical structure recognition (OCSR) and named entity recognition solutions operate in isolation, which limits their scalability for comprehensive literature curation. Here we present MARCUS (Molecular Annotation and Recognition for Curating Unravelled Structures), a tool designed for natural product literature curation that integrates COCONUT-aware schema mapping, CIP-based stereochemical validation, and human-in-the-loop structure refinement. This integrated web-based platform combines automated text annotation, multi-engine OCSR, and direct submission capabilities to the COCONUT database. MARCUS employs a fine-tuned GPT-4 model to extract chemical entities and utilises a Human-in-the-loop ensemble approach integrating DECIMER, MolNexTR, and MolScribe for structure recognition. The platform aims to streamline the data extraction workflow from PDF upload to database submission, significantly reducing curation time. MARCUS bridges the gap between unstructured chemical literature and machine-actionable databases, enabling FAIR data principles and facilitating AI-driven chemical discovery. Through open-source code, accessible models, and comprehensive documentation, the web application enhances accessibility and promotes community-driven development. This approach facilitates unrestricted use and encourages the collaborative advancement of automated chemical literature curation tools.

Received 15th July 2025
Accepted 21st September 2025

DOI: 10.1039/d5dd00313j

rsc.li/digitaldiscovery



## Introduction

Chemical literature contains large amounts of molecular information that often remains inaccessible due to being embedded in unstructured formats within printed or digital documents.[1] The scientific literature continues to expand at a high rate, with cross-database analysis reporting an average annual growth rate of 4.10%, suggesting that global publications double approximately every 17 years. Chemistry contributes significantly to this surge: the CAS REGISTRY® exceeded 279 million unique substances in early 2025, up from just over 105 million a decade earlier, with CAS scientists adding 13.5 million structures in 2015 alone.[1] Most of this information remains encapsulated in PDFs whose text and figures are not intrinsically machine-readable, creating a critical bottleneck for large-scale data utilisation.[2]

With the growing demand for machine learning-based tools and machine learning itself being recognised as the 'fourth pillar' of chemistry,[3] high-quality, machine-readable data have become increasingly important, particularly as graph neural networks now lead molecular property prediction benchmarks when trained on large, reliable structure datasets.[4] The FAIR principles likewise emphasise machine readability as a prerequisite for data reuse.[5] Together, these trends create a pressing need to convert unstructured literature into structured, query-ready datasets capable of facilitating AI-driven discovery pipelines.

Text extraction has reached relative maturity, with transformer models achieving F1 scores of 86.7% on full-text PubMed articles for chemical entity recognition.[6] For example, models trained on the NLM-Chem corpus have achieved F1 scores of 86.7% on full-text PubMed articles.[7] However, Optical Chemical Structure Recognition (OCSR) – the conversion of graphical depictions to a machine-readable format remains challenging. Current tools like DECIMER.ai struggle with heterogeneous drawing styles.[8] On curated benchmarks, MolScribe reaches a high accuracy, but on noisy images, it degrades.[9] A 2020 review catalogued typical failure modes, from bond-length variation to mislabelled atoms.[10] In a 2024 study, eight open-access OCSR tools were compared, revealing F1 scores ranging from 34 to 93%, with no single tool consistently outperforming others across all categories of chemical structure images.[11]

[a]*Institute for Inorganic and Analytical Chemistry, Friedrich Schiller University Jena, Lessingstr. 8, 07743 Jena, Germany. E-mail: christoph.steinbeck@uni-jena.de*
[b]*Institute for Bioinformatics and Chemoinformatics, Westphalian University of Applied Sciences, August-Schmidt-Ring 10, 45665 Recklinghausen, Germany*

These performance disparities have motivated ensemble approaches, where image-aware routing strategies direct different structure types to optimal recognisers. Such arbitration already surpasses the exact-match accuracy of the best individual tool.[11] Nevertheless, end-to-end extraction demands more than recognition: diagrams must be linked back to captions and body text, reaction components reconciled, and chemical validity enforced. Recent advances in multimodal language models demonstrate F1 values of around 88–96% for complex structure parsing,[12] yet their raw outputs often violate valence rules or yield redundant tautomers.

Recent advances demonstrate the importance of contextual integration: ReactionDataExtractor 2.0 achieves ~90% F1 scores by co-analysing image labels with surrounding text.[13] Cross-media linkage methods enable alignment between visual panels and textual ref. 14, while multimodal language models like RxnIM and MarkushGrapher achieve F1 scores of 88% and 96%, respectively, for parsing reaction images and Markush structures.[15],[16] These tools, however, don't yield the expected accuracy in real-world applications since raw recognition outputs often violate valence rules or produce redundant tautomers.

The work by Wang *et al.* introduced BioChemInsight (Wang *et al.* 2025), an automated pipeline for extracting chemical structures and bioactivity data from pharmaceutical literature and patents. While BioChemInsight focuses on Structure Activity Relationship (SAR) data extraction for drug discovery, MARCUS addresses distinct challenges in natural product structure curation. Unlike BioChemInsight's pharmaceutical focus, MARCUS is specifically designed for natural products research, featuring COCONUT-specific schema mapping, taxonomic metadata linking (organism and geographical information), and stereochemical validation with CIP annotations. Additionally, MARCUS provides human-in-the-loop validation through integrated Ketcher editing capabilities, ensuring curation quality for complex natural product structures that often contain intricate stereochemistry and diverse structural motifs.

To address these comprehensive workflow challenges and integrate the fragmented landscape of chemical information extraction tools, we present MARCUS (Molecular Annotation and Recognition for Curating Unravelled Structures), a containerised web platform that transforms unstructured text and images in raw PDFs of articles about natural products into machine-readable molecular records and allows for submission to the open-access database COCONUT, as illustrated in Fig. 1. MARCUS employs Docling for document conversion and a fine-tuned GPT-4 model to extract chemical entities from text while routing chemical structures through an ensemble of three OCSR engines – DECIMER, MolNexTR, and MolScribe – for optimal recognition accuracy. The platform features a Vue.js frontend with a FastAPI backend, supporting concurrent users through queue-based session management and Docker-based deployment across operating systems.

By bridging the gap between the dynamic growth of literature and the structured data requirements of modern AI-driven discovery, the platform enables automated chemical literature curation while maintaining human oversight for quality assurance. The platform is accessible through a user-friendly web

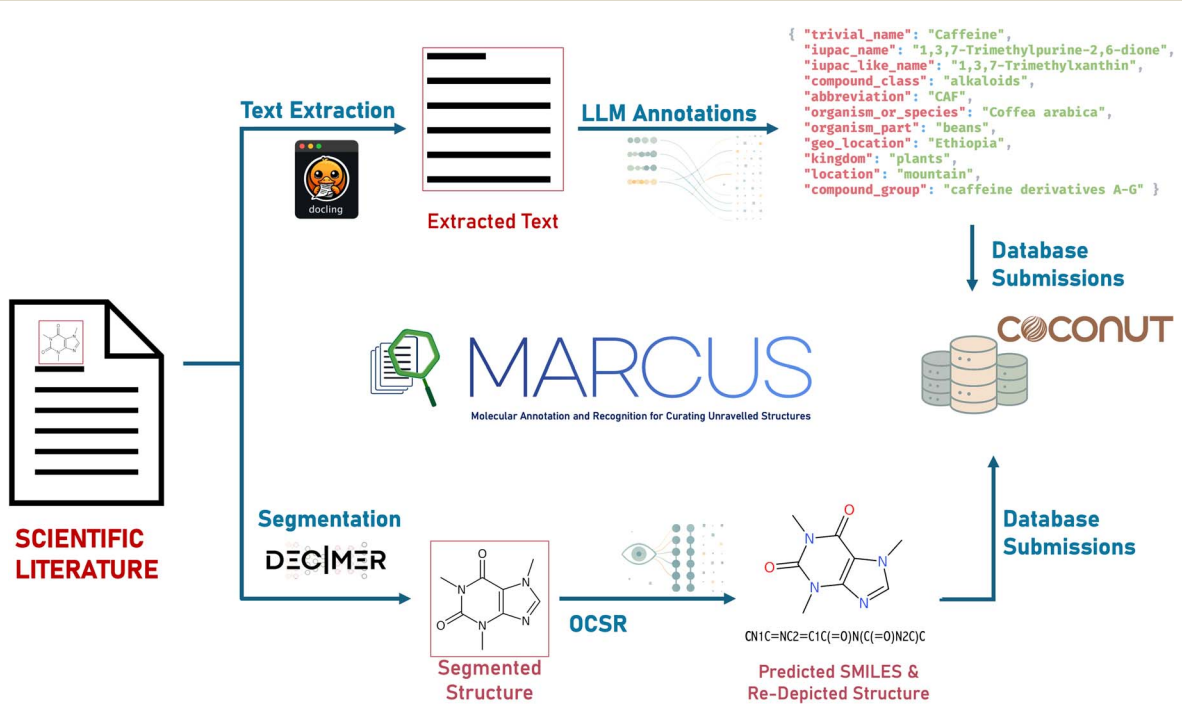


Fig. 1 MARCUS workflow overview for automated chemical literature curation. Scientific literature undergoes parallel text extraction and annotation (top) and structure segmentation with OCSR (bottom). Both pathways converge in MARCUS for validation and direct submission to the COCONUT database.

interface at **https://marcus.decimer.ai**, designed for researchers with varying technical expertise. Through open-source code availability at **https://github.com/Kohulan/MARCUS** and comprehensive documentation, MARCUS promotes community-driven development and encourages collaborative advancement of automated chemical literature curation tools. While MARCUS currently only supports the submission to the COCONUT database, it can be easily adapted for other OA databases.

## Methods

MARCUS comprises three primary components; A PDF processing module that enables users to upload PDF files and extract text in a structured manner, an annotation module that utilises a pre-trained deep learning model to annotate the extracted text and thirdly an Optical Chemical Structure Recognition (OCSR) module, which detects, identifies, and extracts chemical structure depictions from PDFs and converts them into machine-readable formats using OCSR models. In addition to these core components, several auxiliary functions have been developed to enhance user experience and usability, along with a submission interface. The platform is a web application with a Vue.js frontend and a Python-based FastAPI backend, designed to provide easy access to complex computational tasks.

### Text extraction and annotation

To create the annotation dataset, 100 articles were manually selected from the *Journal of Natural Products*. Each publication had to meet two criteria: it was published in the 2000s and reported at least one newly discovered or synthesised molecule from a living organism.

Articles were retrieved as PDF files from the American Chemical Society website and processed using the Docling[17] optical character recognition library. Docling was selected for its lightweight nature, open-source availability, high performance, and local execution capability, ensuring a reproducible preprocessing pipeline. The Docling-generated JSON output was parsed to extract the title, Abstract, and Introduction sections and combined as a single paragraph. These sections were chosen as they typically contain the highest density of chemical entity mentions and novel compound information.

To train an entity classifier for extracting molecules and related information, the text files required manual annotation. The text annotation was done using Doccano,[18] an open-source annotation platform selected for its collaborative features and ease of use. The 100 publications were divided into five groups of 20 papers to expedite the manual labelling process through parallel annotation by multiple team members.

Ten entity types were defined based on chemical literature requirements and alignment with the COCONUT[19] database schema. These are listed in Table 1.

After completing the annotation, all documents, including the labelled entity spans and types, were exported in JSON format. These JSON files were then used to fine-tune a large language model (LLM) for automated data extraction.

### Training of the GPT model and automated annotation integration

The annotation model was fine-tuned using OpenAI's GPT-4 architecture. A larger language model was selected due to the limited size of the available fine-tuning dataset, as such models have demonstrated strong performance even with relatively small training sets. Initial experiments used GPT-3.5-turbo-1106 with automatic batch size and learning rate optimisation. Following model deprecation, we transitioned to GPT-4o-2024-08-06 with similar settings and 2 epochs, achieving improved overall accuracy. Fine-tuning was performed entirely through the OpenAI API.

The fine-tuning used 2 epochs, batch size 1, and learning rate multiplier 2, totalling 210 194 tokens from our 100 annotated papers. This final parameter configuration was determined through multiple fine-tuning iterations with manual validation of entity extraction accuracy, optimising the balance between effective domain adaptation and overfitting prevention while minimising hallucination in chemical entity recognition.

While commercial models offer strong performance, their closed-source nature poses limitations. Therefore, an open-source model represents the most suitable alternative in future.

The fine-tuned model was integrated with Doccano's auto-annotation feature, enabling automated pre-labelling of text *via* a single click. This significantly accelerates the annotation

Table 1 Explanation and examples of entity types used for labelling

| Label | Explanation | Example |
|---|---|---|
| trivial_name | Common molecular names without IUPAC nomenclature | Caffeine |
| iupac_name | Systematic IUPAC chemical nomenclature | 1,3,7-Trimethylpurine-2,6-dione |
| iupac_like_name | Hybrid nomenclature combining IUPAC and trivial naming | 1,3,7-Trimethylxanthin |
| compound_class | Formal classification of compounds based on structure or function | Alkaloids |
| abbreviation | Shortened form of the molecular name | CAF |
| organism_or_species | Complete organisms or biological species | *Coffea arabica* |
| organism_part | The anatomical or functional component of an organism | Beans |
| geo_location | Specific geographical locations | Ethiopia |
| kingdom | High-level biological classifications | Plants |
| location | General physical or abstract places | Mountain |
| compound_group | Enumeration of molecules grouped into one word | Caffeine derivatives A-G |

process for additional documents, supporting iterative model improvement and rapid performance assessment.

The auto-annotation system operates through a Flask-based[20] API server, which handles REST requests initiated by Doccano. When the 'auto-annotate' feature is triggered, the target text is sent *via* a REST call to the Flask server, which forwards it to the fine-tuned language model for inference. The model identifies and extracts relevant annotations, which are then mapped back to the original text to determine precise character offsets. These span data are transmitted back to Doccano for visual rendering on the user interface. A visual representation of the complete process is illustrated in Fig. 2.

Text annotation processing requires 10–20 seconds per document, with costs approximately under $0.04 per publication through the OpenAI API. The fine-tuned model was specifically optimised for natural products literature terminology, including organism names, bioactive compound classes, and natural product-specific nomenclature, which may limit performance on synthetic medicinal chemistry or other chemical domains.

### Structure detection, segmentation and extraction

Before a molecule can be interpreted by an Optical Chemical Structure Recognition (OCSR) engine, the PDF page must be "cut out" so that each drawing is isolated. This cutting step—called segmentation—treats the page like a collage and surrounds every chemical diagram with its bounding box, creating a small image snippet for each structure. MARCUS performs this task with DECIMER-Segmentation v1.5.0,[2] an open-source model that both (i) spots chemical graphics reliably and (ii) returns their exact page number and $x$–$y$ coordinates. The resulting snippets are written to a folder named after the source PDF; inside, every file is labelled with its page and segment index, giving curators an unambiguous link back to the original document.

### Optical chemical structure recognition of the extracted structures

MARCUS enables users to convert extracted chemical structure depictions into machine-readable formats using Optical Chemical Structure Recognition (OCSR) tools. Here, three open-source OCSR models were integrated to provide ensemble validation and improved accuracy. The following three models are included. DECIMER,[8,21] an EfficientNetV2+ transformer-based model developed in-house, provides direct control over development and future improvements. DECIMER offers both standard and hand-drawn structure recognition capabilities. MolNexTR,[22] a generalised deep learning model which combines a ConvNext and a Vision Transformer in its architecture, along with a transformer decoder that provides both SMILES strings and molfiles with atomic coordinates. The coordinate information enables precise molecular visualisation, matching the original structure orientation. MolScribe[23] is an image-to-graph generation model that explicitly predicts atoms and bonds with geometric layouts. Molscribe includes a Swin Transformer encoder and a Transformer decoder. MolScribe incorporates symbolic chemistry constraints to enhance chirality recognition and facilitate abbreviated structure expansion. MolScribe also generates molfiles with atomic coordinates.

MARCUS employs a human-in-the-loop multi-model ensemble approach that enables the cross-validation of extracted structures, where users manually select the best prediction after reviewing all model outputs. Unlike automated ensemble methods that combine predictions algorithmically, this approach prioritises transparency and expert oversight. Users can select individual models or run all three for comparison. A Tanimoto similarity calculation matrix using ECFP fingerprints and Maximum Common Substructure (MCS) highlighting using RDKit's find maximum common substructure algorithm facilitates analysis of prediction consistency across models. These

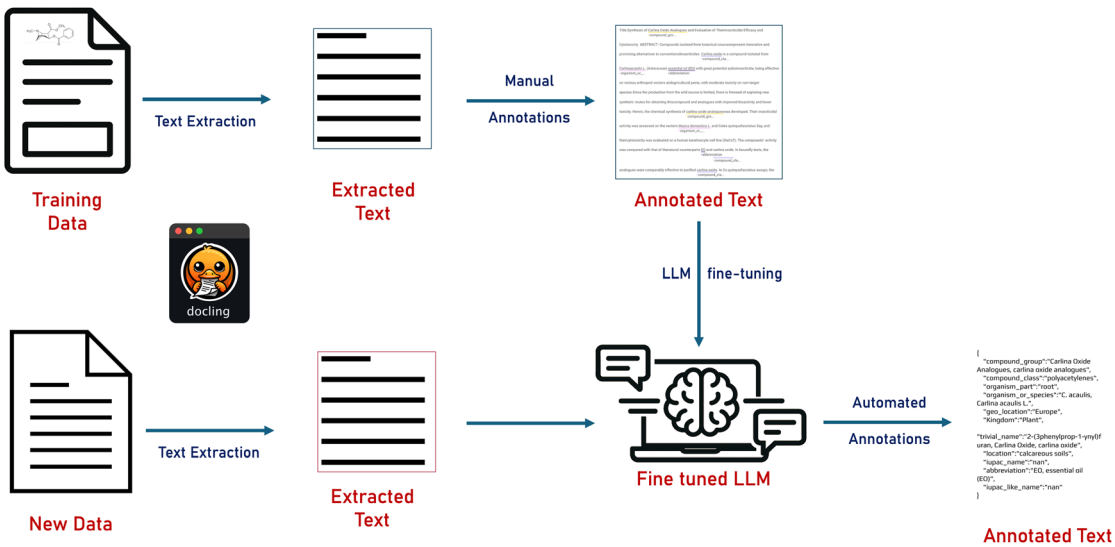


Fig. 2 Summary of the training and testing of text extraction and entity annotation.

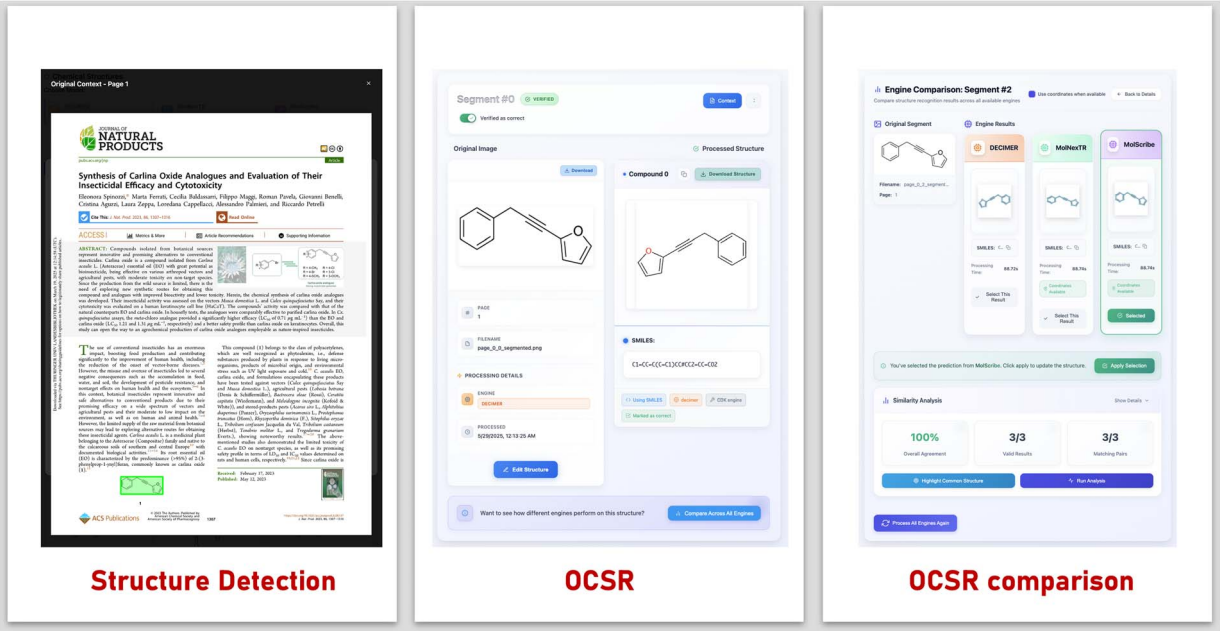


Fig. 3 Summary of structure detection and extraction, followed by OCSR and OCSR comparison.[24]

analyses help users assess model agreement and select the most accurate result. Fig. 3 illustrates the complete process of structure detection and extraction, followed by OCSR and OCSR results comparison in MARCUS.

An initial evaluation was conducted on 15 randomly selected articles from the *Journal of Natural Products* to assess model performance. A detailed analysis is provided in the Results section.

### Chemical structure depiction and editing

The predicted SMILES and MolFiles are stored locally and can be downloaded directly by the user. Users can visualise chemical structure predictions using the Chemical Structure Depiction feature, implemented with the Chemistry Development Kit (CDK).[25,26] CDK was chosen for its open-source nature, support for customisation of depiction, and the ability to render stereochemical information with Cahn–Ingold–Prelog (CIP) annotations.[27] CDK provides an integrated diagram layout generator and template-based depictions, enabling the generation of high-quality, publication-grade chemical structure visualisations.

In some cases, the predicted chemical structures may require manual adjustments before being downloaded or submitted to a database. To facilitate this, structure editing capabilities are provided through the integrated Ketcher 3.0 structure editor,[28] enabling manual refinement of predicted structures before final submission. Ketcher's open-source, standalone design allows local operation without external dependencies.

### Data download and submission

All extracted information (text annotations, segmented images, and predicted structures) is compiled into a single JSON file using JavaScript for download or database submission. For direct COCONUT submission, only single molecules can be submitted at a time to maintain proper tracking and provenance. The COCONUT submission form is automatically populated with extracted data, streamlining the database curation process.

### Web application development

The MARCUS platform enables the extraction and annotation of predefined information snippets found within a given piece of literature using LLM-based natural language processing. It also includes detecting chemical structure depictions, extracting them along with their location in the PDF, converting these structures into machine-readable formats using optical chemical structure recognition, and enabling comparison among three different models in cases where the user suspects an incorrect conversion. Finally, the system also supports data download and facilitates workflows for database submission. While this workflow is streamlined, integrating all components in a logical sequence and receiving immediate feedback remains a programmatically challenging task. To make this tool accessible to users with limited or no programming expertise, a user-friendly graphical interface is essential. Given that all models are based on deep learning and require GPU support, a web-based application featuring an intuitive graphical user interface (GUI) and a backend capable of dynamically invoking the necessary models and tools would provide the most effective implementation. To ensure this, a comprehensive full-stack web application with modern frontend interfaces and backend infrastructure was developed.

### Backend implementation

The backend was written using FastAPI,[29] a modern Python web framework for generating RESTful APIs. This was chosen for its

high performance, automatic API documentation, and type validation through Pydantic models.[30] The architecture follows a modular design with specialised routers handling distinct functionalities. This enables the creation of robust and scalable APIs. The main application establishes a versioned API using fastapi-versioning,[31] supporting both versioned endpoints (/v1/) and latest version shortcuts (/latest/) for backwards compatibility. API documentation is automatically generated through integrated Swagger UI[32] and ReDoc interfaces. Environment configuration utilises python-dotenv[33] for flexible deployment across development and production environments.

The backend consists of six routers, including those for text extraction, LLM-based text annotation, DECIMER-based chemical structure detection and segmentation, optical chemical structure recognition (OCSR), chemical structure depiction, and similarity calculation. Detailed information about each component is explained below.

**Docling conversion router.** Manages PDF text extraction using Docling.[17] Processes uploaded PDFs and returns a structured JSON containing extracted text, metadata, and document structure. Supports configurable page processing limits and automatic PDF validation. Users can also extract the text directly in plain text format.

**OpenAI annotation router.** Handles fine-tuned GPT-4 model interactions for molecular entity recognition. Processes extracted text to identify chemical entities by model inference and returns character-level span information with regular expressions. Implements result caching through local JSON storage to prevent duplicate processing.

**DECIMER segmentation router.** Manages chemical structure detection and extraction from PDFs. Identifies and extracts chemical structure depictions with metadata, including bounding box coordinates, page numbers, and segment indices. Implements hierarchical directory organisation for extracted segments. This router also allows users to extract the doi of the uploaded document directly using the Python package PDF2DOI.

**OCSR engine router.** Coordinates three OCSR models (DECIMER,[8] MolNexTR,[22] MolScribe[23]). Converts structure images to SMILES strings[34,35] or molfiles with atomic coordinates if available. Supports ensemble processing for cross-validation and accuracy. Also allows direct generation of the depiction of the predicted SMILES or MolFiles using CDK. The wrapper handles model initialisation, input preprocessing, and output standardisation.

**Depiction router.** Generates chemical structure depictions from SMILES or molfile input. Creates molecular depictions using the CDK[25] with configurable dimensions and output formats (SVG, PNG, base64). Includes support for stereochemistry annotations and substructure highlighting.

**Similarity router.** Provides molecular comparison and analysis. Calculates Tanimoto similarity matrices using ECFP fingerprints using RDKit[36] and determines the maximum common substructure across multiple molecules.[37] Returns the SMARTS patterns for substructure highlighting.

Additionally, to ensure fair access and limit the number of simultaneous users, a session management system has been implemented. The Session Router handles user sessions and concurrency control *via* WebSocket and HTTP endpoints, maintaining real-time session status and queue position information for up to three concurrent users. The system uses Python's asyncio framework[38] with deque data structures for efficient queue management. Each session receives a unique UUID[39] and maintains metadata including creation time, last activity, and current status. The number of users is currently restricted to three due to hardware limitations, but it can be easily increased if there is sufficient hardware capacity to meet.

Real-time communication utilises WebSocket connections[40] for immediate status updates, queue position notifications, and session state changes. The system includes automatic fallback to HTTP polling when WebSocket connections are unavailable. Background cleanup tasks are executed every 30 seconds to remove expired sessions and promote queued users to active status.

The backend also includes a hierarchical file storage system within the uploads directory for file storage and data management, organising files into subdirectories for original PDF uploads (PDFs), extracted chemical structure images (segments), temporary processing images (chem_images), and cached annotation results (openai_results).

Each PDF upload generates a unique identifier, ensuring isolation between concurrent users. Extracted segments include complete metadata serialised to JSON files for complete tracking.

### Frontend implementation

The frontend is implemented using Vue.js 3 ref. 41 with the Composition API, selected for its reactive data binding, component-based architecture, and development efficiency. The application also utilises Vue CLI 5 ref. 42 for build tooling and development server management.

The frontend includes centralised state management that utilises Vuex with five specialised modules and a PDF module, which manages PDF file handling, upload status, and document metadata with reactive updating for PDF viewer synchronisation. A text module that handles extracted text content, processing status, and text manipulation operations with extraction history support. An annotations module that stores entity recognition results, annotation statistics, and highlighted text spans with category management and confidence scoring. A structure module that coordinates chemical structure segments, OCSR results, and processing configurations, which also implements caching mechanisms for processed structures and supports batch processing operations with segment selection management. A theme module that controls visual themes with automatic system preference detection and persistent user settings supporting smooth transitions between light and dark modes.

### User interface design, real-time communication, and component architecture

The interface is built on a custom design system implemented with SCSS,[43] utilising CSS Grid and Flexbox[44] for responsive

layouts. The design system features custom colour palettes supporting both light and dark themes, a comprehensive feedback mechanism including loading states and progress indicators, and a flexible three-column layout with drag-and-drop resizing capabilities.

WebSocket integration ensures persistent connections for real-time session status updates and processing progress notifications. The system includes automatic reconnection with exponential backoff and connection health monitoring to maintain performance. In cases where WebSocket connections are unavailable, HTTP polling provides a reliable fallback mechanism.

The application features a three-column responsive layout managed by the home page, comprising three main component groups. First, the PDF Components include a drag-and-drop PDF upload component with file validation and progress indicators, as well as a PDF viewer that supports embedded visualisation with zoom and page navigation controls. Second, the text processing components consist of a text panel for displaying extracted text with syntax highlighting, an annotated text viewer for rendering highlighted entities, and an annotation viewer that presents tabular annotation data with filtering options. Finally, the structure processing components include a segments panel offering grid-based visualisation with selection tools, a segment viewer for displaying individual segments alongside OCSR model outputs, a structure editor based on the Ketcher structure editor for modifying predicted structures, and a comparison component that enables side-by-side analysis of model predictions with similarity scoring.

The service layer abstracts API communication through specialised modules. The core API service uses Axios[45] with automatic versioning, request/response interceptors, and error handling. Session-aware API client integration provides automatic session ID injection and session-specific error handling.

### Workflow integration, performance and error handling

When a PDF is uploaded, it triggers the automatic creation of a session and text extraction. Extracted text passes through fine-tuned language models with span validation. Chemical structures undergo segmentation followed by ensemble OCSR processing. Results display with interactive editing capabilities, downloading of results and direct database submission to COCONUT. The frontend optimisations include lazy loading for large component trees, virtual scrolling for extensive segment lists, and caching of processed results. Backend optimisations utilise connection pooling for database interactions, asynchronous processing with FastAPI's native async/await support, and memory-efficient file streaming for large PDF processing. The system implements comprehensive error handling with automatic retry mechanisms for transient failures and graceful degradation when individual services are unavailable. Security features include session-based authentication with secure token generation, CORS policy enforcement, request validation and sanitisation, and file upload security with content-type verification.

### Hardware and computational requirements

MARCUS requires GPU-accelerated hardware for optimal performance. The development environment utilised NVIDIA V100 GPUs (approximately 20 GB VRAM) with 64 GB RAM and 16-core processors. For production deployment, the minimum requirements include an NVIDIA GPU (approximately 20 GB VRAM) and an 8-core CPU; however, 32 GB RAM is recommended for improved performance. The current implementation supports up to three concurrent users due to GPU memory constraints, scalable with additional hardware resources. Docker Engine 20.10+ with NVIDIA Container Toolkit is required for deployment, along with stable internet connectivity for OpenAI API integration. Approximately 50 GB of storage is needed for container images and temporary processing files.

## Results and discussion

### OCSR model performance evaluation

To validate the human-in-the-loop ensemble approach and demonstrate the complementary strengths of different OCSR models, a systematic evaluation of the three integrated engines was conducted. This evaluation provided the necessary information for implementing multiple OCSR models rather than relying on a single tool.

A total of 20 publications were randomly selected and downloaded from four major journals in natural products chemistry to ensure diversity in publication styles, chemical structure types, and drawing conventions. The selection included five articles each from *Journal of Natural Products* (ACS Publications), *Phytochemistry* (Elsevier), *Molecules* (MDPI), and *Phytochemical Analysis* (Wiley). Chemical structure segmentation was performed using DECIMER segmentation v1.5.0 on all 20 publications.

Each extracted structure image was processed through all three OCSR engines to generate SMILES predictions. Model performance was assessed through manual validation by comparing the regenerated molecular structures (from SMILES) with the original segmented images. A prediction was considered accurate only if the generated structure matched the original depiction exactly (1 : 1 match); otherwise, it was classified as incorrect and excluded from the count. Fig. 4 summarises the overall accuracy of the tools across all the publications. Structures deemed irrelevant for natural product research (such as 2D-NMR correlation diagrams or X-ray crystallographic representations) were excluded from accuracy calculations.

The evaluation revealed similar performance levels among the three OCSR models, with each demonstrating distinct strengths across different publication types. DECIMER achieved the highest average accuracy of 75%, followed by MolScribe at 73%, and MolNexTR at 69%. See Fig. 4 for more details. Performance varies significantly by journal type, with all models showing the lowest accuracy on JNP articles (51–63%) and the highest on Phytochemical Analysis articles (73–93%). Despite similar overall averages, each model demonstrates distinct strengths across different publication styles, supporting the Human-in-the-loop ensemble approach implemented in MARCUS.

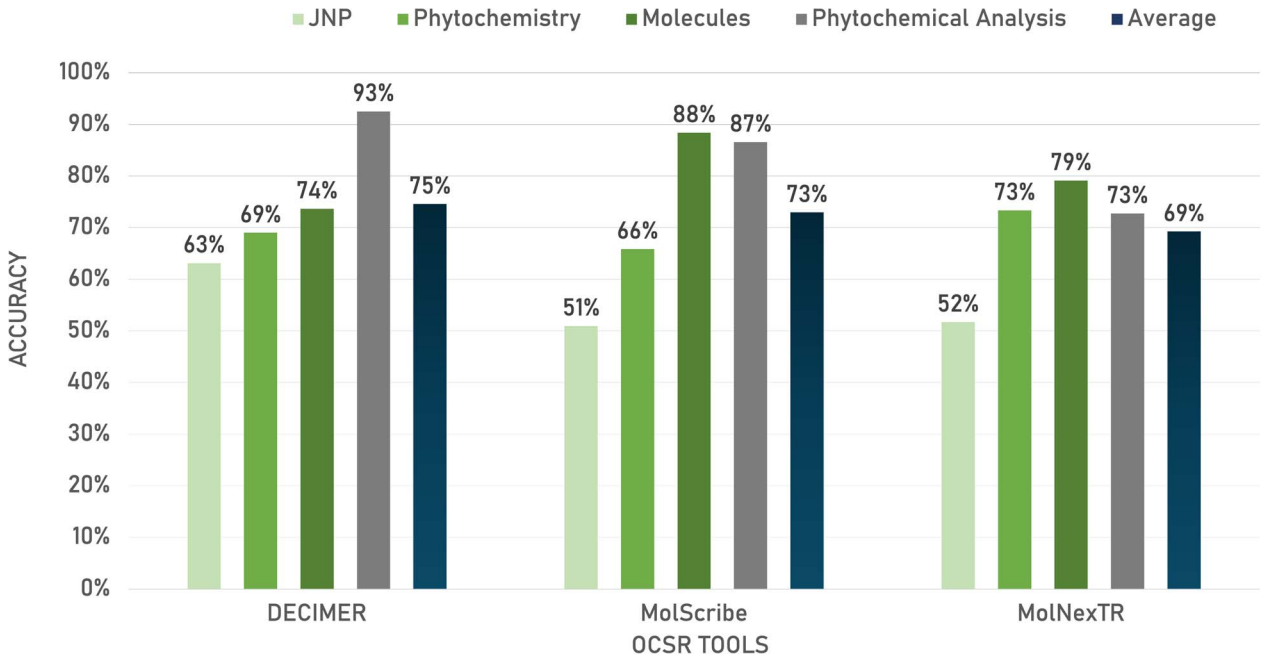


Fig. 4 Comparative OCSR model performance across different journal types and overall accuracy. Bar chart showing accuracy percentages for DECIMER, MolNexTR, and MolScribe across four chemistry journals: *Journal of Natural Products* (JNP, light green), *Phytochemistry* (green), *Molecules* (dark green), and *Phytochemical Analysis* (deep green). The overall average accuracy across all journals is shown in blue.

Contrary to expectations, across four journal types, the lowest performance still occurs on *Journal of Natural Products* (JNP): DECIMER 63%, MolScribe 51%, and MolNexTR 52%. In contrast, the highest performance was observed in Phytochemical Analysis articles, where DECIMER reached 93%, MolScribe 87%, and MolNexTR 73%. MolNexTR shows the most consistent behaviour across journal types (52–73%), while MolScribe exhibited the greatest variability (51–87%); DECIMER spans 63–93%. Despite the competitive overall performance, DECIMER showed a unique advantage in handling Markush structures by representing variable groups with 'R' notations rather than asterisks directly in SMILES, facilitating downstream processing for structure enumeration. DECIMER also includes a dedicated model specifically trained for hand-drawn chemical structure diagrams, which can be used when needed. Both MolScribe and MolNexTR generate MolFiles with atomic coordinate information, enabling structure depictions that match the original orientation found in the literature. This allows curators to visually compare predicted structures with the source images, aiding in validation and accuracy assessment.

The observed performance levels (69–75% average accuracy) were notably lower than reported benchmark accuracies for these models, reflecting the challenges of real-world literature processing compared to curated test datasets. Critical factors contributing to reduced performance include varying image quality across different publishers, diverse drawing styles and conventions, complex stereochemical representations in natural product structures, and the absence of standard molecular depictions.

Performance evaluation revealed that the three OCSR models possess complementary strengths, exhibit varying accuracy across different structure types, and display diverse error patterns. These findings support the implementation of all three OCSR engines in MARCUS, enabling users to cross-validate predictions and select the most appropriate results for their specific applications. The Human-in-the-loop ensemble approach provides both improved overall accuracy through model consensus and transparency in cases where models disagree, enabling users to make informed decisions about structure quality.

This evaluation directly influenced the development of comparison features in MARCUS, including Tanimoto similarity calculations and Maximum Common Substructure (MCS) analysis for ensemble validation. It also guided the implementation of confidence scoring mechanisms and user interface elements that highlight consensus or disagreement across models. The evaluation confirms that, despite the limitations of individual OCSR models across diverse literature sources, the human-in-the-loop ensemble approach in MARCUS improves extraction accuracy and provides users with clear tools to evaluate prediction reliability.

## Web application

The web application provides an intuitive, browser-based interface that integrates all processing components into a seamless workflow. The platform can handle concurrent user sessions through its queue-based management system, maintaining stable performance with up to three simultaneous users processing PDFs of varying complexity.

Using MARCUS, users can improve their data extraction process significantly compared to manual curation processes.

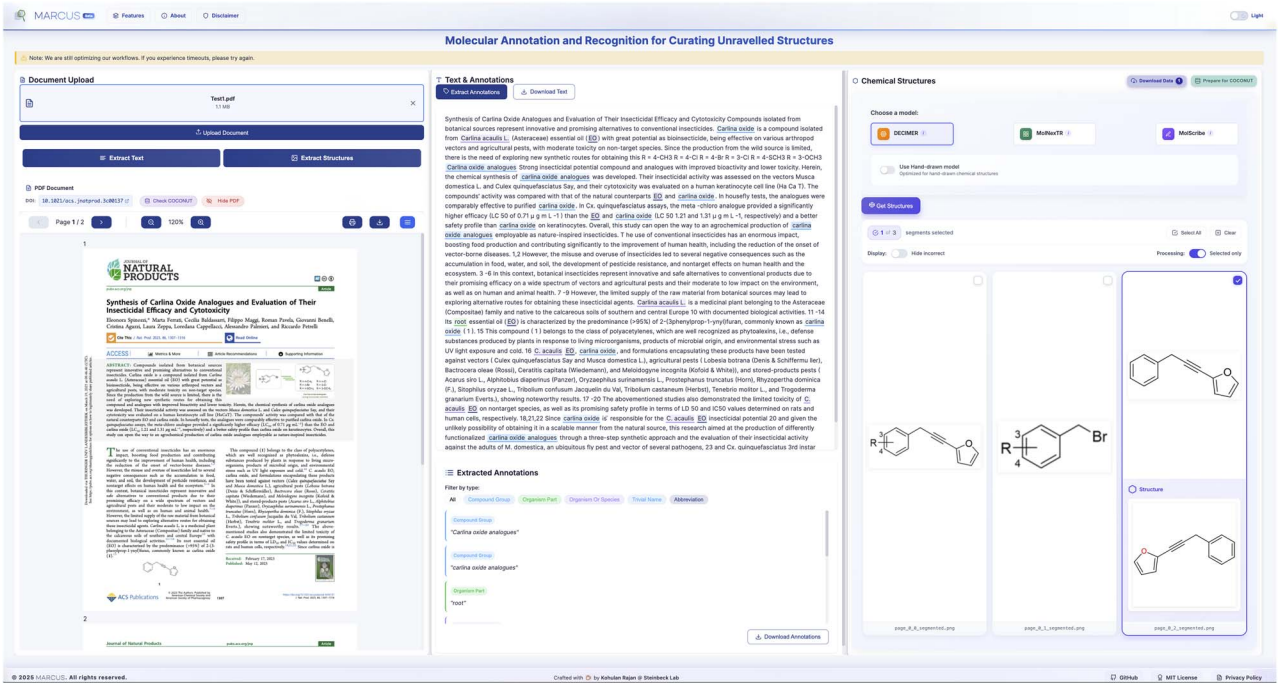


Fig. 5 The MARCUS web application interface, featuring the document upload panel with PDF visualisation, the text and annotations panel displaying extracted text and annotated entities, and the chemical structures panel showing segmented structure images with multi-engine OCSR model selection. The interface demonstrates the integrated workflow from PDF processing through text annotation to structure recognition, enabling users to validate and compare results across different OCSR engines.

The integrated three-panel interface (Fig. 5) enabled seamless navigation between PDF viewing, text annotation review, and structure analysis within a single browser session. Processing times vary by document complexity, with typical natural products papers (10–15 pages) completing text extraction within 10 seconds, text annotation within 10–20 seconds, and structure segmentation within 60–90 seconds. The ensemble OCSR processing adds 2–3 minutes per structure set, depending on the number of detected chemical diagrams. Real-time progress indicators and status updates enhanced the user experience, particularly during longer processing operations.

### Data export, download and COCONUT database submission integration

MARCUS provides a data compilation and export functionality where all extracted information, including text annotations, segmented images, predicted structures (SMILES and molfiles), and metadata, is compiled into structured JSON files for the processed PDF files.

The exported data maintains complete traceability, with each structure segment linked to its original page coordinates and segment index, enabling users to verify predictions against source images. Batch selection functionality allowed users to export subsets of structures, with selected data packages ranging from individual structures to complete document sets.

The automated submission workflow for COCONUT is integrated with the COCONUT database schema. The platform extracts and formats metadata, including compound names, organism information, and structural classifications for database submission. DOI extraction and linking provided proper citation information for submitted entries.

The integrated Ketcher editor enables structure refinement as needed, before database submission. The submission interface maintains the approach of one molecule per submission, required for proper tracking and provenance. Form auto-population reduces manual data entry requirements, although the extent varies depending on the completeness of the extracted metadata from individual publications.

The streamlined process eliminates many of the manual steps traditionally required in database curation workflows, though expert oversight remains necessary for quality assurance and validation of automated results.

## Conclusion

MARCUS combines several established methods of document conversion, language–model–based entity recognition, structure segmentation, and an ensemble of three OCSR engines into a workflow that curators can run from a single browser tab. In the evaluation carried out on fifteen recent natural-product papers, the platform accurately recovered textual metadata with high precision and successfully converted roughly two-thirds of the extracted structure images into a correct, machine-readable form without requiring manual intervention. While these figures leave room for improvement, they already shorten the path from PDF to database record and demonstrate

the practical value of combining complementary OCSR models rather than relying on just one.

The web interface, backed by FastAPI and containerised services, proved stable on modest hardware and enabled concurrent sessions with real-time feedback. Curators can inspect every prediction, adjust structures in the embedded editor, download the full annotation package as a single JSON file, or hand the record straight to COCONUT. Keeping all code, model checkpoints, and documentation openly available should make it easier for others to adapt the system to additional journals or target databases.

Several limitations remain. The accuracy of image-to-structure conversion still depends strongly on drawing style and image quality, and the current queue length is restricted by GPU capacity. Future work will focus on expanding training data, refining routing strategies for difficult images, and exploring lighter-weight inference back-ends so that more users can work simultaneously. We also plan to add automated consistency checks between the text-derived metadata and the decoded structures, as well as export options for other open-access repositories. The current system is optimised for natural products literature and may show reduced performance on medicinal chemistry or synthetic chemistry documents due to domain-specific training data.

In its present form, MARCUS cannot replace the expert curator, but it does take over many routine steps that previously consumed most of the curation time. We hope that the community will test the system, provide feedback, and help us turn MARCUS into a dependable companion for translating the growing body of natural products literature into FAIR, machine-actionable data.

## Author contributions

KR initiated, designed, tested, applied, and validated the software features and wrote the paper. VW implemented an annotation platform, curated data and wrote the paper. LL did the OCSR analysis. CS and AZ conceived the project and supervised the work. All authors contributed to and approved the manuscript.

## Conflicts of interest

AZ is co-founder of GNWI – Gesellschaft für naturwissenschaftliche Informatik mbH, Dortmund, Germany. The remaining authors declare no financial and non-financial competing interests.

## Data availability

Project name: MARCUS (Molecular Annotation and Recognition for Curating Unravelled Structures). Project home page: **https://marcus.decimer.ai**. Source code: **https://github.com/Kohulan/MARCUS**. Archived Version: **https://doi.org/10.5281/zenodo.17212920**. License: MIT Models: All automatically download from Zenodo DECIMER: **https://doi.org/10.5281/zenodo.8300489**. Operating system(s): Platform-independent (web-based application). Programming language: Python/FastAPI (backend), JavaScript/Vue.js (frontend).

## Acknowledgements

KR acknowledges the funding by the German Research Foundation under project number 239748522-SFB 1127 ChemBioSys (Project INF). VW acknowledges the funding by the Federal Ministry of Education and Research under project number 03ZU1214OA ThWIC (Project AutoQSPR). We are pleased to dedicate MARCUS to Dr Marcus Ennis, the longest-serving curator of the ChEBI database, on the occasion of his 75th birthday. We are grateful that Google provides us with free computing time on its TPU Research Cloud infrastructure. KR acknowledges the support through Cloud TPUs from Google's TPU Research Cloud (TRC). The authors would like to thank Mr Venkata Chandrasekhar Nainala and Dr Jonas Schaub for their assistance with data annotation. The authors wish to thank Dr Peter Ertl for his valuable contribution to the discussion.

## References

1 Chemical Abstracts Service (CAS), *Annual Report 2015*, American Chemical Society, 2016.
2 K. Rajan, H. O. Brinkhaus, M. Sorokina, A. Zielesny and C. Steinbeck, DECIMER-Segmentation: Automated Extraction of Chemical Structure Depictions from Scientific Literature, *J. Cheminform.*, 2021, **13**, 20, DOI: **10.1186/s13321-021-00496-1**.
3 O. A. von Lilienfeld, Introducing Machine Learning: Science and Technology, *Mach. Learn.: Sci. Technol.*, 2020, **1**, 010201, DOI: **10.1088/2632-2153/ab6d5d**.
4 P. Reiser, M. Neubert, A. Eberhard, L. Torresi, C. Zhou, C. Shao, H. Metni, C. van Hoesel, H. Schopmans, T. Sommer, *et al.*, Graph Neural Networks for Materials Science and Chemistry, *Commun. Mater.*, 2022, **3**, 93, DOI: **10.1038/s43246-022-00315-6**.
5 M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, *et al.*, The FAIR Guiding Principles for Scientific Data Management and Stewardship, *Sci. Data*, 2016, **3**, 1–9, DOI: **10.1038/sdata.2016.18**.
6 M. C. Swain and J. M. Cole, ChemDataExtractor: A Toolkit for Automated Extraction of Chemical Information from the Scientific Literature, *J. Chem. Inf. Model.*, 2016, **56**, 1894–1904, DOI: **10.1021/acs.jcim.6b00207**.
7 H. Kim, M. Sung, W. Yoon, S. Park and J. Kang, Full-Text Chemical Identification with Improved Generalizability and Tagging Consistency, *Database*, 2022, **2022**, baac074, DOI: **10.1093/database/baac074**.
8 K. Rajan, H. O. Brinkhaus, M. I. Agea, A. Zielesny and C. Steinbeck, DECIMER.ai: An Open Platform for Automated Optical Chemical Structure Identification, Segmentation and Recognition in Scientific Publications, *Nat. Commun.*, 2023, **14**, 5045, DOI: **10.1038/s41467-023-40782-0**.

9 Y. Qian, J. Guo, Z. Tu, Z. Li, C. W. Coley and R. Barzilay, MolScribe: Robust Molecular Structure Recognition with Image-to-Graph Generation, *J. Chem. Inf. Model.*, 2023, **63**, 1925–1934, DOI: **10.1021/acs.jcim.2c01480**.

10 K. Rajan, H. O. Brinkhaus, A. Zielesny and C. Steinbeck, A Review of Optical Chemical Structure Recognition Tools, *J. Cheminform.*, 2020, **12**, 60, DOI: **10.1186/s13321-020-00465-0**.

11 A. Krasnov, S. J. Barnabas, T. Boehme, S. K. Boyer and L. Weber, Comparing Software Tools for Optical Chemical Structure Recognition, *Digital Discovery*, 2024, **3**, 681–693, DOI: **10.1039/D3DD00228D**.

12 R. Malhotra and T. Ito, The Doubly Librating Plutinos, *arXiv*, 2025, preprint, arXiv:2501.12345, DOI: **10.48550/arXiv.2501.12345**.

13 D. M. Wilary and J. M. Cole, ReactionDataExtractor 2.0: A Deep Learning Approach for Data Extraction from Chemical Reaction Schemes, *J. Chem. Inf. Model.*, 2023, **63**, 6053–6067, DOI: **10.1021/acs.jcim.3c00422**.

14 S. Yan, S. Spangler and Y. Chen, Cross Media Entity Extraction and Linkage for Chemical Documents, In *Proceedings of the Proceedings of the AAAI Conference on Artificial Intelligence*, 2011, vol. 25, pp. 1455–1460, doi: DOI: **10.1609/aaai.v25i1.7832**.

15 Y. Chen, C. T. Leung, J. Sun, Y. Huang, L. Li, H. Chen, H. Gao, Towards Large-Scale Chemical Reaction Image Parsing via a Multimodal Large Language Model, *arXiv*, 2025, preprint, arXiv:2503.08156, DOI: **10.48550/arXiv.2503.08156**.

16 L. Morin, V. Weber, A. Nassar, G. I. Meijer, L. Van Gool, Y. Li, P. Staar, Joint Visual and Textual Recognition of Markush Structures, *arXiv*, 2025, preprint, arXiv.2503.16096, DOI: **10.48550/arXiv.2503.16096**.

17 C. Auer, M. Lysak, A. Nassar, M. Dolfi, N. Livathinos, P. Vagenas, C. B. Ramis, M. Omenetti, F. Lindlbauer, K. Dinkla, *et al.*, Docling Technical Report, *arXiv*, 2025, preprint, DOI: **10.48550/arXiv.2408.09869**.

18 H. Nakayama, T. Kubo, J. Kamura, Y. Taniguchi, X. Liang, *Doccano: Text Annotation Tool for Human*, 2018, **https://github.com/doccano/doccano**.

19 V. Chandrasekhar, K. Rajan, S. R. S. Kanakam, N. Sharma, V. Weißenborn, J. Schaub and C. Steinbeck, COCONUT 2.0: A Comprehensive Overhaul and Curation of the Collection of Open Natural Products Database, *Nucleic Acids Res.*, 2024, gkae1063, DOI: **10.1093/nar/gkae1063**.

20 Pallets Flask, 2024, **https://github.com/pallets/flask**.

21 K. Rajan, H. O. Brinkhaus, A. Zielesny and C. Steinbeck, Advancements in Hand-Drawn Chemical Structure Recognition through an Enhanced DECIMER Architecture, *J. Cheminform.*, 2024, **16**, 78, DOI: **10.1186/s13321-024-00872-7**.

22 Y. Chen, C. T. Leung, Y. Huang, J. Sun, H. Chen and H. Gao, MolNexTR: A Generalized Deep Learning Model for Molecular Image Recognition, *J. Cheminform.*, 2024, **16**, 141, DOI: **10.1186/s13321-024-00926-w**.

23 Y. Qian, J. Guo, Z. Tu, Z. Li, C. W. Coley and R. Barzilay, MolScribe: Robust Molecular Structure Recognition with Image-to-Graph Generation, *J. Chem. Inf. Model.*, 2023, **63**, 1925–1934, DOI: **10.1021/acs.jcim.2c01480**.

24 S. Eleonora, M. Ferrati, C. Baldassarri, F. Maggi, R. Pavela, G. Benelli, C. Aguzzi, L. Zeppa, L. Cappellacci, A. Palmieri and R. Petrelli, Synthesis of Carlina Oxide Analogues and Evaluation of Their Insecticidal Efficacy and Cytotoxicity, *J. Nat. Prod.*, 2023, **86**(5), 1307–1316, DOI: **10.1021/acs.jnatprod.3c00137**.

25 E. L. Willighagen, J. W. Mayfield, J. Alvarsson, A. Berg, L. Carlsson, N. Jeliazkova, S. Kuhn, T. Pluskal, M. Rojas-Chertó, O. Spjuth, *et al.*, The Chemistry Development Kit (CDK) v2.0: Atom Typing, Depiction, Molecular Formulas, and Substructure Searching, *J. Cheminf.*, 2017, **9**(1), DOI: **10.1186/s13321-017-0220-4**.

26 C. Steinbeck, Y. Han, S. Kuhn, O. Horlacher, E. Luttmann and E. Willighagen, The Chemistry Development Kit (CDK): An Open-Source Java Library for Chemo- and Bioinformatics, *J. Chem. Inf. Comput. Sci.*, 2003, **43**, 493–500, DOI: **10.1021/ci025584y**.

27 R. S. Cahn, C. Ingold and V. Prelog, Specification of Molecular Chirality, *Angew Chem. Int. Ed. Engl.*, 1966, **5**, 385–415, DOI: **10.1002/anie.196603851**.

28 B. Karulin and M. Kozhevnikov, Ketcher: Web-Based Chemical Structure Editor, *J. Cheminform.*, 2011, **3**, 1, DOI: **10.1186/1758-2946-3-S1-P3**.

29 S. Ramírez, *Fastapi: FastAPI Framework, High Performance, Easy to Learn, Fast to Code, Ready for Production*; **https://github.com/fastapi/fastapi**.

30 Pydantic, *Data Validation Using Python Type Hints*, **https://github.com/pydantic/pydantic**.

31 D. Way, *Fastapi-Versioning: Api Versioning for Fastapi Web Applications*, **https://github.com/DeanWay/fastapi-versioning**.

32 Swagger-Ui, *Swagger UI Is a Collection of HTML, JavaScript, and CSS Assets That Dynamically Generate Beautiful Documentation from a Swagger-Compliant API*, **https://github.com/swagger-api/swagger-ui**.

33 S. Kumar, Python-Dotenv: Reads Key-Value Pairs from a .env File and Can Set Them as Environment Variables, *It Helps in Developing Applications Following the 12-Factor Principles*, **https://github.com/theskumar/python-dotenv**.

34 D. Weininger, SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules, *J. Chem. Inf. Comput. Sci.*, 1988, **28**, 31–36, DOI: **10.1021/ci00057a005**.

35 D. Weininger, A. Weininger and J. L. Weininger, SMILES. 2. Algorithm for Generation of Unique SMILES Notation, *J. Chem. Inf. Comput. Sci.*, 1989, **29**, 97–101, DOI: **10.1021/ci00062a008**.

36 G. Landrum, *RDKit: Open-Source Cheminformatics*, 2006, **https://github.com/rdkit/rdkit**.

37 A. Dalke and J. Hastings, FMCS: A Novel Algorithm for the Multiple MCS Problem, *J. Cheminf.*, 2013, **5**, 1, DOI: **10.1186/1758-2946-5-S1-O6**.

38 G. Van Rossum, R. Hettinger, PEP 3156 - Asynchronous IO Support Rebooted: The "Asyncio" Module. Python Enhancement Proposal, 2012, **https://peps.python.org**.

39 P. Leach, M. Mealling and R. Salz, RFC 4122: A Universally Unique IDentifier (UUID) URN Namespace, *Internet Engineering Task Force*, 2005.
40 I. Fette; A. Melnikov *RFC 6455: The WebSocket Protocol*. *Internet Engineering Task Force* 2011.
41 Vue: This Is the Repo for Vue 2. For Vue 3, **https://github.com/vuejs/core**.
42 Vue-Cli: Webpack-Based Tooling for Vue.js Development, **https://github.com/vuejs/vue-cli**.
43 Sass: Sass Makes CSS Fun!, **https://github.com/sass/sass**.
44 T. Atkins, *CSS Flexible Box Layout Module Level 1. W3C Recommendation*, 2017.
45 Axios: Promise Based HTTP Client for the Browser and Node.js, **https://github.com/axios/axios**.