ChemScraper: Graphics Extraction, Molecular Diagram Parsing, and Annotated Data Generation for PDF Images

Ayush Kumar Shah[1], Bryan Amador[1], Abhisek Dey[1], Ming Creekmore[1], Blake Ocampo[2], Scott Denmark[2], Richard Zanibbi[1]

[1]Document and Pattern Recognition Lab, Rochester Institute of Technology, NY, USA.
[2]Department of Chemistry, University of Illinois at Urbana-Champaign, IL, USA.

Contributing authors: as1211@rit.edu; ma5339@rit.edu; ad4529@rit.edu; mec5765@rit.edu; blakeo2@illinois.edu; sdenmark@illinois.edu; rxzvcs@rit.edu;

Abstract

Existing visual parsers for molecule diagrams translate pixel-based raster images such as PNGs to chemical structure representations (e.g., SMILES). However, PDFs created by word processors including LaTeX and Word provide explicit locations and shapes for characters, lines, and polygons. We extract symbols from born-digital PDF molecule images and then apply simple graph transformations to capture both visual and chemical structure in editable ChemDraw files (CDXML). Our fast (PDF → visual graph → chemical graph) pipeline does not require GPUs, Optical Character Recognition (OCR) or vectorization. We evaluate on standard benchmarks using SMILES strings, along with a novel evaluation that provides graph-based metrics and error compilation using LgEval. The geometric information in born-digital PDFs produces a highly accurate parser, motivating generating training data for visual parsers that recognize from raster images, with extracted graphics, visual structure, and chemical structure as annotations. To do this we render SMILES strings in Indigo, parse molecule structure, and then validate recognized structure to select correct files.

Keywords: chemoinformatics, graphics extraction, graphics recognition, evaluation, data generation, PDF

1 Introduction

This work addresses a fundamental need for developing a scalable and reliable extraction and translation system for PDF-based chemical molecule drawings. Such a system will facilitate applications such as data mining and entity linking for multi-modal chemical search, along with chemical search in PDF documents. A key application is molecular search in PDF documents – in particular, supplementary materials documenting experiments associated with chemical papers. This would allow chemists to query molecules in PDF files, import retrieved molecules to chemistry-specific tools that enable adding or modifying sub-graphs, simulate novel reactions, etc.

Current approaches to recognizing molecule structure generally parse images from pixel-based raster images, and produce chemical structure descriptions such as SMILES strings as output.

Acknowledgements. This work was supported by the National Science Foundation (USA) through Grant No. 2019897 (Molecule Maker Lab Institute). We thank Matt Berry and his team at NCSA for their contributions to the ChemScraper online tool and related system improvements.

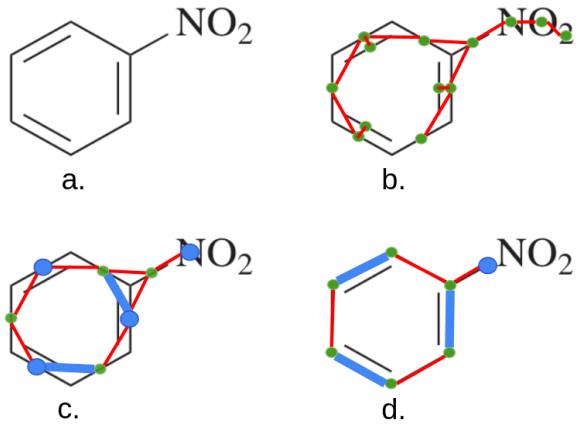


Fig. 1: Parsing Nitrobenzene ($C_6H_5NO_2$). (a) PDF image. (b) MST over lines/characters: green dots are nodes, red lines are edges. (c) Modified MST after updating connectivity and merging nodes: large blue dots are merged characters and bond lines, thick blue lines are added edges. (d) Final graph: thick blue lines are double bonds, and a large blue dot is a superatom group (NO_2).

A number of these approaches work well, and some include modern variations of encoder/decoder models that recognize structure with high accuracy (see Section 2).

However, many modern documents are produced using word processors that utilize vector representations to depict molecules. These representations encode diagrams as characters, lines, and other graphic primitives. We wish to use PDF drawing instructions directly as input to produce fast, accurate methods for converting molecule images at scale. We were motivated to use PDF drawing instructions directly by earlier math formula recognition work by Baker et al. [1].

In the early part of Section 4 we describe our improved SymbolScraper[36] that extracts PDF drawing instructions without the need for consulting rendered pages images. Later in Section 4 we describe the ChemScraper born-digital parser, which is both fast and simple in its design[1]. As illustrated in Figure 1, starting from PDF graphical primitives, a Minimum Spanning Tree (MST) is then built over these primitives to capture two-dimensional neighbor relationships (i.e., visual structure). Graphical primitives in the MST are tokenized/merged into molecule elements such as atom/superatom names and double/triple or wedge/hash bonds. Graph transformations using geometric features and simple chemical constraints augment and correct the tokenized MST into a final graph that represents the molecular structure.

We also use ChemScraper's parser to generate fine-grained annotated data for visual parsers, with primitive-level annotations for all graphical primitives, atoms, and bonds (see Section 5). The parser is also one component in the online ChemScraper molecule extraction tool[2], which includes a YOLOv8[43]-based diagram detection module not described in this paper.

We represent recognized visual structure and molecular structure in ChemDraw's CDXML format[3][26], which combines visual appearance with semantic annotations. CDXML can also be translated to standard chemoinformatics formats such as SMILES and MOL (see Section 3 for details). In Section 6, we use the translations to evaluate our model using three different representations: SMILES strings, molecular fingerprints, and labeled directed graphs. The use of direct comparisons of labeled graphs over PDF drawing primitives is a contribution of this paper; it allows direct comparison of graphical structures, and automatic and exhaustive compilation of structure recognition errors. In addition, we report some differences that are missed in the SMILES strings commonly use to evaluate molecular diagram parsers.

In the next Section, we begin with an overview of related work in chemical structure recognition.

2 Related Work

We provide below a summary of work in Chemical Structure Recognition (CSR), and contrast and compare this work with the ChemScraper born-digital parser presented in this paper.

Chemical structure recognition from heterogeneous scientific documents (containing text, images, charts, and tables) requires locating the region of the page where a 2D molecule diagram is drawn and then parsing the localized structure

[1] Code and tools from this paper are publicly available: https://gitlab.com/dprl/graphics-extraction/-/tree/icdar2024

[2] https://chemscraper.frontend.staging.mmli1.ncsa.illinois.edu/configuration

[3] https://revvitysignals.com/products/research/chemdraw

into a machine-readable form for further use (e.g., in search applications). While most systems focus on parsing the structure of individual molecules into common string representations like SMILES, DeepSMILES [27], InChI or SELFIES [17], some recent works also try to address localizing diagrams in documents, including YOLOv8, an updated version of Scaled YOLOv4 [43] with performance and efficiency enhancements. There are numerous standard datasets, including USPTO, CLEF, UOB, to benchmark parsing individual molecules, which is the focus of this paper.

In the following sections, we discuss traditional rules-based systems, neural-based systems, followed by systems that are rule- or neural-based, but generate molecular structures as explicit graphs rather than strings (e.g., in SMILES).

2.1 Rule-Based

The earliest structure parsing system for chemical diagrams in printed documents, which we know, was a rule-based approach by Ray et al. in the late 1950's [31]. This approach first enumerated atoms, and then the connections between atoms were established from molecule regions in scanned document images. Special chemical compound rules based on the number of connections for each atom were used to determine the type of bond between atoms. While this system worked well for common compounds, the rules were complex and worked for a limited set of compounds.

An important later development was the creation of the Kekulé system [22]. The main differences between Kekulé and Ray et al.'s system were additional pre-processing steps and the visual detection of bond types. Kekulé used thinning and vectorization of raster scans to eliminate subtle variations in bond lines and characters and ensured that a consistent set of characters and lines were recovered. Once a connection between a pair of atoms was established, their system visually detected their bond type instead of using chemical rules as Ray et al. did.

Ibison et al. developed CLiDE, [14] which also detected atoms and then connected them with bonds. CLiDE detects fewer bond types other than single, double, or triple such as solid and dashed wedge bonds that illustrate 3-dimensional structure for bonds (e.g., indicating that structure lies behind or in front of the page). Connected component analysis was used in disconnected bond groups to identify bond types, and OCR was used to identify atoms (characters). The final adjacency matrix for the molecular structure was created similar to Kekulé. Another system by Comelli et al. [5], used additional processing steps to identify charges as subscripts or superscripts attached to atoms.

A still-popular open-source system that extends the rules of CLiDE and Kekulé to improve performance is OSRA by Filipov et al. [7]. Their system uses methods similar to previous approaches but was refined to process images for born-digital documents which had well-defined encoded text lines, characters, and graphics. A similar system was MolRec [33], which used horizontal and vertical grouping to detect connected atoms, their charge, and stereochemical information. The system had some failures for molecules that use arcane representations of common bond types or complex structures including those with stereochemical information (e.g., isomerism). The CSR system developed by Bukhari et al. is a recent work that still uses rule-based graphical processing to output SMILES representations for molecules. However, they use a chemical naming toolkit, *OpenBabel* [28] to generate the correct connectivity table.

ChemScraper is also a rule-based system, with a series of graph transformation rules, using the geometry of characters and graphical objects, along with chemical constraints (e.g., neighboring parallel lines often represent double, triple, or hash bonds). However, unlike many previous systems, it does not rely on image processing, visual features, or OCR. Instead, it leverages PDF instructions, resulting in faster processing with less uncertainty (e.g., line and character locations and geometric properties are known before parsing). With this reduced uncertainty, ChemScraper's rules are robust and can handle complex structures.

2.2 Neural-Network Based

Recent advances in neural networks have shown promise in detecting and parsing chemical diagrams.

Sun et al. [40] used a single pass feedforward convolutional network to extract chemical diagrams from documents. To address the issues

of scale and size of diagrams, they used Spatial Pyramidal Pooling (SPP) [11]. This made their approach perform better than other popular object detection networks like Faster R-CNN and SSD, which were designed for images in the wild. Staker et al. [39] used an entirely neural approach to extract figures from documents and convert them into a SMILES representation. For diagram extraction, they used a U-Net [32] to segment the figures. The segmented figures were then passed through an attention-based encoder network [42] to predict the SMILES string.

Some neural systems focus on parsing chemical diagrams exclusively. DECIMER by Rajan et. al [30] follows a similar encoder-decoder approach, taking features extracted from a bitmap image of a molecule from an encoder and passing it through a decoder. The main difference is the structure of the outputs generated, as they used SMILES, DeepSMILES, and SELFIES. They found that SELFIES performed much better because of additional information encoded within them vs. SMILES strings.

Additional encoder-decoder parsing models include IMG2SMI by Campos et al. [2]. Instead of using the molecule image as an input to the encoder transformer, a Resnet-101 [10] backbone was used to extract image features that were then passed on to the encoder stage. The BMS (Bristol–Myers–Squibb) dataset [16] released by Kaggle provided one of the few datasets for a general baseline for the conversion of molecule images to InChI (International Chemical Identifier names). Li et al. [19] modified a TNT vision transformer encoder [8] by adding an additional decoder. This attempt at using a vision transformer was enabled due to the training dataset containing 4 million molecule images. Likewise, SwinOCSR by Xu et al. [46] use the Swin transformer to encode image features and another transformer-based decoder to generate DeepSMILES. They focus on the improvements due to the backbone (Swin transformer) and use focal loss to address the token imbalance problem in text representations of molecular diagrams.

Most current neural-based methods encode visual features using an encoder, and then decode these embedded representations into strings (e.g., SMILES or DeepSMILES) that do not correspond naturally to molecular structures. These string representations lack direct geometric representation between input objects (e.g., atoms and bonds) and the output strings, and require extensive training data [23].

In contrast, ChemScraper is designed to recognize structure and create annotated molecular images using the Indigo Toolkit, with additional primitive-level annotations from Symbol Scraper [36] and their visual as well as chemical structure. These additional annotations include labels and positions of characters, which are integral parts of atom groups, even if not directly linked to the main bond (e.g., `H` and `3` in CH_3). Datasets generated by ChemScraper's born-digital parser will be helpful for fine-grained training of visual parsers that consider these connections between input locations and output structure representation during training and recognition (e.g., the LGAP [37] parser, a visual parser originally designed for parsing mathematical formulas).

2.3 Graph Decoders and Graph-Structured Outputs

In recent years, novel molecular diagram parsing methods have emerged that combine rule-based and neural-based approaches and generate graph representations as outputs, rather than string representations such as SMILES. These methods often employ a graph decoder or a graph construction algorithm to create graph-based outputs. These outputs usually represent a supergraph of atoms and bonds or serve as an intermediate representation of the final graph structure.

MolScribe [29] employs a SWIN transformer to encode molecular images and a graph decoder, which consists of a 6-layer transformer, to jointly predict atoms, bonds, and layouts, yielding a 2D molecular graph structure. They also incorporate rule-based constraints to determine chirality (i.e., 3d topology) and design algorithms to expand abbreviated structures. MolGrapher [23] is another noteworthy method employing a graph-based output representation. It utilizes a ResNet-18 backbone to locate atoms, and constructs a supergraph incorporating all feasible atoms and bonds as nodes while imposing specific constraints. Subsequently, a Graph Neural Network (GNN) is applied to the supergraph, accompanied by external Optical Character Recognition (OCR) for node classification. Both these systems utilize

multiple data augmentation strategies, including diverse rendering parameters, such as font, bond width, bond length, and random transformations of atom groups, bonds, abbreviations, and R-groups to bolster model robustness.

Likewise, Yoo et al. [47] and OCMR [44] produce graph-based outputs directly from molecular images. Yoo et al. [47] leverage a ResNet-34 backbone, followed by a Transformer encoder equipped with auxiliary atom number and label classifiers. Their model includes a transformer graph decoder with self-attention mechanisms for edges. On the other hand, Wang et al. [44] employ multiple neural network models for different parsing steps. These steps include key-point detection, character detection, abbreviation recognition, atomic group reconstruction, atom and bond prediction. A graph construction algorithm is subsequently applied to the outputs.

These graph-based methods present exciting alternatives, offering improved interpretability and robustness while representing chemical structures naturally. Utilizing a graph output structure, as opposed to traditional SMILES strings, offers enhanced interpretability. Atom-level alignment with input images facilitates easy examination, geometric reasoning, and correction of predicted results.

As a result, ChemScraper uses graph representations for output. Unlike MolScribe [29], which initially converts a molecular graph to a MOL file, ChemScraper introduces a novel visual graph $\rightarrow$ CDXML converter, that encodes both physical locations as well as chemical information for one or more molecules. CDXML provides the flexibility to be directly used in many downstream tasks by chemists, read in ChemDraw-like tools as well as for conversion to other formats such as SMILES, MOL, and InChI [12, 13]. It is essential to again note that ChemScraper does not rely on OCR or other neural networks to recognize keypoints, characters or bond types.

The systems commonly used for molecule and reaction parsing system comparison baselines are OSRA, DECIMER (described above), and the reaction extraction work done by Lowe [20]. However, it should be noted that reaction extraction work by Lowe was done by tagging text-based reaction XML files from exclusively USPTO patents and converting IUPAC [38] names to SMILES. This involved classifying text into reactants and products.

3 Molecular Representations

Specialized molecular representations broadly enable various aspects of cheminformatics, information modeling, and cross-representation between formats. For instance, it enables a common representation and translation between molecule figures and their corresponding text-based IUPAC [38] (International Union of Pure and Applied Chemistry) name. Some of the most common text-based specialized representation formats are SMILES (Simplified Molecular-Input Line-Entry System) [45], InChI (International Chemical Identifier) [12, 13], and SELFIES (SELF-referencIng Embedded Strings) [17]. While these formats do not encode the precise layout of the molecule in 2D or 3D space, parsers (e.g., RDKit [18], Marvin molconvert [4], and OpenBabel [28]) for these formats have builtin knowledge to convert these representations using spatial geometry.

Representations that explicitly encode 3D geometry for atoms and their bond types include MOL (molecular data) file and an XYZ file (e.g., as used in Avogadro [9]). These explicitly capture the arrangement of carbon atoms with respect to each other, and the spatial arrangement of atoms often impacts the property of a molecule. For example, in a chiral molecule with a stereogenic carbon, the orientation of atoms around this carbon will result in a specific stereoisomer. In a 2D representation of this molecule, atoms connected to this carbon will be either on the plane, coming out of the plane (solid wedge bond), or going into the plane (hashed wedge bond).

Furthermore, a detailed understanding of the CDXML file format is essential for encoding visual graphs produced by the ChemScraper born-digital parser. The sections below summarize ChemDraw XML file contents, SMILES encodings and labeled graph (lg) representation. This has been used for evaluating math formula recognition tasks using the LgEval library [24, 25] and we use it for evaluation in this paper.

3.1 ChemDraw (CDXML) Files

CDXML is an XML encoding that captures how a molecule or a group of molecules are chemically structured, and their appearance on a 2D canvas. This format was created for the ChemDraw chemical diagram editor [15]. CDXML reading tools can modify structures at the molecule, sub–atom or sub–group level as needed. 3D properties such as stereogenic carbons are identified by tag attributes.

After the `<CDXML/>` and `<page/>` headers, every molecule is embedded in a `<fragment/>` tag, with individual atoms are represented in `<n/>` (node) tags, that include the atomic number for atoms. In some cases, multiple atoms are abbreviated in a drawing such as Et which corresponds to a CH_3CH_2 (Ehtyl) group, or Me for a CH_3 (Methyl) group, represented using nested `<fragment/>` tags associated with a node (`<n/>`) tag that defines the structure of the molecule represented by the abbreviation. Where a subgroup of atoms are not chemically interpretable, CDXML encodes it as a node of unknown type using the `NodeType` attribute.

Bonds tags `<b/>` identify the nodes acting as the bond start and end points, referenced using node identifiers. Wedge bonds for chiral carbons contain an additional `Display` attribute to signify the start or end of a chiral bond.

Brackets are encoded outside a fragment. Using separate tags to represent the brackets (`<graphic/>`) and the molecule sub-structure that lies within the brackets (`<bracketedgroup/>`. These are commonly used to represent Markush structures, which indicate repetitions for part of a molecule (e.g., a carbon chain).

3.2 SMILES Strings

Simplified Molecular-Input Line-Entry System or SMILES [45] is widely used in cheminformatics owing to its linear structure, compactness, and easy human readability for domain experts. Atoms are written in an order following a traversal of a chemical structure table (i.e., the adjacency matrix over atoms/atom groups). To translate CDXML to SMILES, the molecule table is generated by reading all the nodes and bonds for a `<fragment/>` and the conversion tool uses an internal heuristic to order atoms based on the spatial positions of the nodes available in a CDXML.

Single, double, and triple bonds are denoted by the symbols $-$, $=$ and $\#$ respectively. Single bonds and hydrogen atoms are generally omitted for clarity in the SMILES. Ethane (CH_2H_6) can be either written as `C-C` or `CC`. SMILES can encode additional properties such as aromaticity [34] and chirality. For instance, the SMILES for benzene (C_6H_6) is commonly written as `c1ccccc1` which. It is important to note that canonicalization is molecule and compound-specific, and different toolkits can have different ways of verifying if a given SMILES is canonical or not – in other words, canonical SMILES is not in fact 'normalized' or universal. One possible 'canonical' form (using RDKit) is `C1=CC=CC=C1`. The beginning and final `C1` signifies a closed loop around the molecule, i.e., a ring.

Although SMILES is generally reliable, it does not protect against invalid strings, i.e. not every combination of characters and symbols is a chemically valid molecule. This is not an issue when translation is done using off-the-shelf toolkits for valid CDXMLs; for invalid CDXML structures, SMILES strings may be invalid molecules.

3.3 Label Graph (Lg) Files

Labeled directed graphs, represented using 'label graph' files (.lg) are a widely adopted representation for training and evaluating the recognition of mathematical formulas. This format finds utility in various applications and is integrated into the LgEval library [24, 25]. Open-source tools, along with detailed file format specifications and tool usage guidelines are accessible to the research community[4].

Our labeled graphs have labels on both nodes and edges. These labels convey the organization of input primitives into objects and their relationships. Within the 'object-relationship' (`OR`) label graph file format, each object is defined by a label and associated list of primitive identifiers. These identifiers correspond to the set of individual elements within an object. In the case of a chemical bond object, this may represent the lines forming a bond, along with the bond type. Similarly, for atom groups, primitives may represent individual characters within the group.

[4] https://gitlab.com/dprl/lgeval

Most commonly, the `OR` format is used to define labeled edges between objects rather than individual primitives, although this effectively provides a more compact description of a graph defined at the primitive level (expressible in the node-edge (`NE`) format). These labeled edges encapsulate the structural relationships between objects and primitives, enabling fine-grained analysis and evaluation.

In the context of molecule diagram parsing, the choice of relationship labels depends on whether bonds are represented as edges or nodes in the graph. In our visual graph molecule representation, bond lines are represented using nodes. In this case, edges could be labeled as `CONNECTED`, `CONCATENATED`, and `ABSENT`, signifying the relationship between bonded atoms/atom groups or concatenating atomic characters within an atom group. In the final chemical structure graph, the bond lines are replaced by edges in the graph representing chemical bonds between atom nodes. In this case, relationship labels denote bond types such as `Single`, `Double`, `Triple`, `Solid Wedge`, and `Hashed Wedge`. These labels are used to characterize the chemical nature of bonds within the molecular structure.

Label graphs, as a representation format, are quite general. While they are prominently used in the context of mathematical formula recognition, their applicability extends to various other problem domains as well. These graphs support representing and evaluating structural similarity in a diverse range of applications. Consequently, in our work the label graph representation serves a dual purpose: firstly, in generating annotated data for the visual parser (as detailed in Section 5), and secondly, for calculating graph-based evaluation metrics to assess the parsing results of ChemScraper (explored in Section 6.3).

4 Parsing Algorithm

In this section, we present our ChemScraper born-digital parser for recognizing the structure of molecular diagrams from PDF images. This includes extracting characters and graphics from PDF using Symbol Scraper [36] to produce the parser inputs, and then use graph transformations to produce a visual and then chemical representation of the molecule.

Character/Graphics Extraction: SymbolScraper

SymbolScraper is a PDF-graphics extraction system [36] for reading drawn shapes and characters in their writing line order from instructions in PDF files, ignoring embedded images. This is made possible by identifying and extracting character glyphs (shapes) embedded in font profiles, and instructions for drawing objects, such as lines and polygons. These glyphs and shape drawing commands contain information on how a *graphics object* is drawn and where on a PDF canvas. Additionally, font profiles contain the symbol label embedded as a Unicode, helping identify character labels (e.g., a specific letter or number) and drawing command types (e.g., to identify whether straight line segments or a curve are drawn).

Each graphic object in a PDF file is delimited by an 'end-graphic' command, and formed by a sequence of drawing instructions. In PDF, the structure of graphics is given primarily by the drawing instructions for *line*, *rectangle* and *curve*. We take these instructions as the primitives of our graphical objects. We extract information on primitives including points, line width, whether they are filled, etc. We add additional information to support parsing later on, including translating the primitives to a topological space using the Java Topology Suite[5], in which we represent objects as *line strings*. From line strings, we can easily compute angles and lengths for lines. For curves, which are represented as a sequence of Bezier points in PDF, we approximate them to a sequence of lines (*line string*) based on the distance between the farthest point in the original curve, and the segment in the approximation.

Sometimes regular bonds in molecule diagrams are drawn as a filled polygon – to handle this we approximate these objects as lines; this is made by checking if the sum of the 2 longest segments of the geometry object correspond to more than 90% of the total perimeter of the polygon. All this information, along with characters, bounding boxes and more is written into a JSON file used by the ChemScraper parser, but SymbolScraper may also be used for other applications.

Listing 1 shows the raw PDF instructions for the leftmost line in the Propane molecule diagram

[5] https://locationtech.github.io/jts/

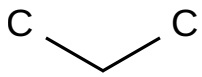


Fig. 2: Propane (C_3H_8) molecule, with implicit hydrogens (H). The bond line intersection at the bottom represents a Carbon (C).

Listing 1: PDF instructions for leftmost line in Fig. 2. `cm` denotes a context matrix defining affine transformations for subsequent graphic objects. `m` moves the cursor to a point. `l` draws a line from the cursor to the specified point.

```
...
1 0 0 -1 0 75 cm
45.926 36.102 m
106.832 71.266 l
...
```

Listing 2: JSON excerpt showing SymbolScraper output for the leftmost line of Fig. 2.

```
...
{
    "typeFromPDF": "line",
    "graphicObjectID": 0,
    "length": 70.32814383876341,
    "angle": 330.00006986692745,
    "lineWidth": 3.333334,
    "points": [
        {"x": 44.48262170992254,
        "y": 39.73133054974975},
        {"x": 108.27537771024348,
        "y": 2.9006694197326697}
    ]
},
...
```

of Fig. 2. Such instructions are in *postfix* notations and processed in a stack-based way. Note that the coordinates in the JSON file output at Listing 2 do not match the coordinates at Listing 1, this is because the actual endpoints of a line depend on factors such as its thickness or the previous context matrices (which are processed cumulatively as instructions are read).

Parsing Model Parameters

Parameters used in the graph transformations of the parser (Steps 1(a) – 1(f) in Fig. 3) are detailed in Table 1. In our work we tune these parameters using grid search over a training dataset, described later in Section 6.

In the remainder of this section, we describe the graph transformations used in the parsing algorithm to produce first a visual graph, and then a chemical structure graph.

.

INPUT: PDF character/graphic locations (JSON)
OUTPUT: Editable molecular diagram (CDXML)

1. **Create Visual Graph**
   (a) Tokenize characters, lines, and shapes using PDF character and shape information
   (b) Construct Minimum Spanning Tree (MST)
   (c) Detect negative charges from MST context
   (d) Add missing edges for touching objects, floating parallel lines and character/line connections. Remove edges for 'floating' objects
   (e) Merge characters into superatom names
   (f) Merge neighboring parallel lines
   (g) Correct bond structures in visual graph
   (h) Merge matching brackets
2. **Translate Visual to Molecular Graph**
   (a) Convert line intersections into carbons
   (b) Convert visual to molecular graph (nodes: atoms/superatoms, edges: bonds)
   (c) Identify nodes in bracketed structures
   (d) Generate CDXML from final graph

Fig. 3: Overview of Parsing Steps. A series of graph transformations convert characters and graphic locations/shapes into a molecular graph.

4.1 Tokenization

After obtaining characters and graphic objects as input primitives from SymbolScraper the Shapely library[6] is used to represent characters by their labels and bounding boxes, and the remaining graphic objects as either polygons or polylines (represented as `LineString` in Shapely).

After this, the following tokenization rules are used to label and group primitives by token type. Please note that the hashed wedge bonds below can only be identified if they are defined explicitly as a graphical object in PDF (e.g., from Indigo), otherwise, they are identified in later processing.

- **Character:** identifed by SymbolScraper.
- **Line:** as identified by Symbol Scraper.
- **Positive Charge (+):** i) graphic object in JSON consists of 2 or more lines, ii) must be a filled polygon, iii) lines are approximately perpendicular with a tolerance of `PERPENDICULAR_TOLERANCE`.
- **Solid Wedge Bond:** i) graphic object consists of 3 or more lines, ii) is a filled and a closed polygon, iii) two longest lines must be

[6] https://github.com/shapely/shapely

Table 1: Parameters for Graph Transformations in ChemScraper. Highlights all parameters for creating visual graph from PDF character/graphics (See Fig. 3)

Parameters (defaults)		Parsing Stages from Fig. 3					
		1(a) Tokenize	1(b) Create MST	1(c) Negative	1(d) Close MST	1(e) Merge chars	1(f) Merge parallel
LONGEST_LENGTHS_ DIFF_TOLERANCE	(0.1)	✓					
SOLID_WEDGE_ MIN_AREA	(50.0)	✓					
PARALLEL_TOLERANCE	(5.0)	✓		✓	✓		✓
PERPENDICULAR_TOLERANCE	(1.0)	✓					
COS_PRUNE	(0.15)		✓				
NEG_CHARGE_Y_POSITION	(0.3)			✓			
NEG_CHARGE_LENGTH_TOLERANCE	(0.5)			✓			
STRAIGHT_TOLERANCE	(20.0)				✓		✓
CLOSE_NONPARALLEL_ALPHA	(1.8)				✓		
CLOSE_CHAR_LINE_ALPHA	(1.5)				✓		
Z_TOLERANCE	(1.6)				✓		
REMOVE_ALPHA	(2.6)				✓		

approximately equal in length with a tolerance of `LONGEST_LENGTHS_DIFF_TOLERANCE`, iv) the minimum area must be less than `SOLID_WEDGE_MIN_AREA`

- **Hashed Wedge Bond:** i) graphic object must consist of 3 or more lines, ii) must not be a filled polygon, iii) all lines must be approximately parallel with a tolerance of `PARALLEL_TOLERANCE` degrees. iv) all line lengths must be in increasing or decreasing order.
- **Left and Right Parentheses:** i) graphic object must be a curve, ii) curve direction determines if it is a left or a right parenthesis.
- **Waves:** i) graphic object must be a list of curves, ii) must have a set of only 1 or 2 curve directions, iii) the polyline approximating the curve must not be closed.
- **Circles:** i) graphic object must be a list of curves, ii) must have a set of more than 2 curve directions, iii) the polyline approximating the curve must be closed.

4.2 Minimum Spanning Tree

After SymbolScraper characters and graphics objects have been tokenized, we compute a complete graph over all pairs of primitives and then extract a Minimum Spanning Tree (MST).

Seeding the Distance Matrix. Edge weights in the complete graph are defined by either (1) for pairs of lines, their minimum end-point distance, and (2) otherwise, the closest pair of points between two primitives. For lines, using the minimum end-point distances has the benefit of avoiding a distance of 0 between overlapping bond lines that are not connected. We also prevent invalid character merges by assigning an infinite distance between characters lying in a roughly superscript or subscript relationship. This is estimated using a limit on the minimum and maximum absolute values that the cosine between two characters center points may take (e.g., accepting angle cosine magnitudes between 0 and 0.15, and 0.85 and 1.0, and treating all other angles as having infinite distance).

MST extraction. Previously, MSTs have been used to recognize the structure of handwritten and typeset math formulas (early examples include Matsakis [21] and Eto and Suzuki [6]). However, typeset chemical diagrams seem even better suited to this technique than math formulas, as neighboring objects are generally grouped or associated with one another, and often touch (e.g., for bond lines between hidden carbons).

We use standard spanning tree algorithms to construct our MST, such as Prim's or Kruskal's algorithm to capture these neighbor relationships. While the MST captures many relationships that are already part of the final chemical structure graph that we will produce, MSTs do not contain cycles, so connections that close benzene rings or show that multiple lines intersect each other are missing. An MST gives a structure connecting every primitive; however, sometimes the molecule may have a 'floating' structure that is separate from the main molecule (e.g., an ion). Named groups (e.g., NO_2) are often separated into a connected chain of individual characters.

In the MST, structures such as brackets and multi-line bonds (double, triple, hashed wedge)

are also split into their component graphic objects. As a result, finding 'hidden' carbons from the line intersections using the raw MST may cause extra carbons to be identified or some carbons to be missed, resulting in an erroneous final graph.

Therefore, it is important to transform the MST so that it contains the correct atom/superatom labels and bond structures before generating the final chemical graph representation. We describe these transformations next.

4.3 Transforming Visual Structure to Chemical Structure

We perform a series of graph transformations on the MST that use geometric features from objects/node, as well as simple chemical constraints (e.g., a double bond is represented by 2 parallel lines). The sequence of steps are described below.

Adding and Removing Edges from the MST

The MST initially contains both spurious and missing edges, necessitating correction. For example, surplus edges may link 'floating' structures to the main graph, while edges are often missing at multi-line intersections, within closed rings, and floating double bond lines not connected with their paired line.

First, we address absent parallel line pairs (e.g., in double bonds) by leveraging MST information. Floating lines (degree 1 in MST, non-intersecting), parallel to another line are identified. A candidate is chosen to pair with a floating bond line if it is adjacent to the floating line (i.e., a perpendicular through the mid-point of the floating line crosses both lines), is among the 5-nearest neighbors of the floating line (there can be a maximum of 4 lines around a multi-line bond), and an average difference between the line-to-line end-point distances between the floating line and candidate parallel line smaller than that of between floating line and its currently connected line. The floating line is then disconnected from its current neighbor, and linked to the selected parallel line.

To close non-parallel line pairs (e.g., multi-line intersections, closed rings) a distance threshold, computed as a multiple of `CLOSE_NONPARALLEL_ALPHA` and the maximum distance between non-parallel line pairs in the updated graph facilitates connecting pairs of lines below this threshold distance. Connecting character-line pairs involves a similar approach, using a distance threshold, with an additional step to filter outliers. A statistical method removes distances falling outside `Z_TOLERANCE` standard deviations (Z-score). Missing character-line edges are then added by a similar distance threshold, computed as a multiple of `CLOSE_CHAR_LINE_ALPHA`, and a maximum distance of the character line pairs in the graph excluding the outliers.

To remove floating atom connections from the main graph, a multiple of `REMOVE_ALPHA` of the distance threshold for closing character line pairs, parallel line pairs, or non-parallel line pairs is applied, prioritized based on availability in the stated order.

Merging Character Groups

We assume all characters connected in an MST represent a named structure. If characters are separated by a graphical object, then they are assumed to not have a relation.

We first need to identify negative charges: they are generally represented as lines, but need to be merged with its parent atom character. They need to satisfy specific conditions: first, they need to be detected as lines by SymbolScraper. Additionally, the angle formed by these lines must be close to zero degrees, and they should be attached to the top-right position of a character in the MST, i.e. the vertical position of the line must be higher than it's parent atom centroid by at least `NEG_CHARGE_Y_POSITION` percentage of the parent's height. To distinguish negative charges from single or double bond lines at the top-right position of an atom, we impose an additional constraint that the length of the negative charge line should be less than `NEG_CHARGE_LENGTH_TOLERANCE` percentage of the mean length of all the bond lines.

Next, character groups are determined by creating sub-graphs that exclude graphical objects. The connected components of the resulting graph define character groups. These connected components are merged and relabeled by the complete character group as read left to right in the graph traversal order of the connected component characters in the MST. When these characters are merged and relabeled, the position of the entire

group is automatically changed to the position of the main atom connected to the graph. This position is relabeled to be the position of the character that is closest to one of the group's neighbors. If a character group has no neighbor, then it is declared as a 'floating' node that is not a part of the main molecule, and its position is left alone.

Merging Parallel Lines

Double bonds, triple bonds, and hashed wedges are represented by parallel lines in the MST that need to be merged. All parallel neighboring line pairs in the updated MST are merged into the same bond. These merged lines are relabeled using the number of lines merged, which determines the specific type of bond. In order to differentiate if lines are actually part of the same bond grouping or are colinear, the angle between the base parallel line and the comparison line formed from the midpoints of the two parallel lines is determined. This can be seen in Fig. 4 (a) and (b). When the calculated angle is perpendicular, the two lines are part of the same bond (as shown in Fig. 4 (a)). On the other hand, the angle will be a straight angle or close to zero, compared using `STRAIGHT_TOLERANCE` when the lines are colinear (as shown in Fig. 4 (b)).

Identifying/Updating Bond Types

After parallel lines are merged, bond types can be identified using graphic shapes and parallel line groups. This determination is necessary for bonds which were unable to be determined in the tokenization stage. Solid wedge bonds, wavy bonds, hashed wedge bonds are most likely to be already determined at the previous stage. In case of hashed wedge bonds, there could be cases where it was not determined if the list of parallel lines was extracted as separate individual graphic objects by SymbolScraper instead of a single grouped object with multiple lines. These bond types, including the missed hashed wedge bonds are identified using the parallel line groups formed earlier using the following simple rules:

- Single bond: a single line
- Double bond: two merged parallel lines
- Either Triple or Hashed wedge bond: three merged parallel lines
- Hashed wedge bond: a line with more than three merged parallel lines

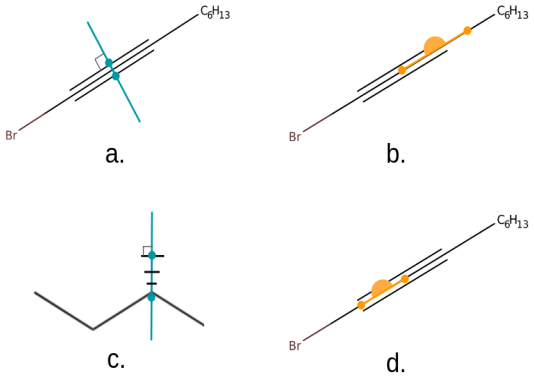


Fig. 4: Bonds for Adjacent Parallel Lines. (a) contains five lines: three for a triple bond, plus two single bonds at the triple bond ends. Here bond membership is determined using a line between the outermost parallel line midpoints, and the parallel lines' direction. (a) right angle: lines in same double or triple bond. (b) 0-degree difference: separate bonds. To differentiate 3 parallel lines as a hashed wedge or triple bond, a line is formed through the midpoints of the parallel line endpoints, and one neighbor's closest endpoint. (c) right angle: hashed wedge bond. (d) 0-degree difference: triple bond.

To distinguish hashed wedge bonds from triple bonds, we apply the logic illustrated in Fig. 4 (c) and (d). The comparison line is formed from a random neighbor's closest point to the merged parallel line and its midpoint. The angle between the comparison line and the merged parallel line is perpendicular for lines forming a hashed wedge, and a straight angle or close to zero degrees for a triple bond. A hashed wedge will always have a neighbor since it is used to declare a bond's three-dimensional position relative to other bonds. Therefore, if there are no neighbors, then the line cannot be a hashed wedge and is declared as a triple bond. This is the case where the molecule is carbon triple-bonded to carbon.

Wedge bonds have a shorter side that begins the bond and a longer side that ends the bond, showing the direction of the bond. A solid wedge bond represents this using a trapezoid. A hashed wedge bond represents this through parallel lines of increasing length. Unlike the other bond types, we cannot use the default endpoints. The beginning of a solid wedge bond is identified by the shortest line in the trapezoid. The opposite side is

the end of the bond. For a hashed wedge bond, the shortest line in the group is the beginning of the bond, and the longest line in the group is the end of the bond. The midpoints of the two identified lines are used as the bond's actual endpoints.

Merging Brackets

The MST already includes nodes with bracket labels. However, the opening bracket and the closing bracket are identified as two separate nodes. The opening and closing brackets constituting a pair need to be merged into a single node. There is no guarantee that there is only one bracket pair, and opening and closing brackets are not explicitly identified, so pairs are identified through positioning. Bracket nodes are arranged in a list sorted by increasing x-coordinates. This ensures that the initial items in the list correspond to opening brackets, while subsequent items represent closing brackets. Subsequently, bracket pairs are identified by their y-coordinates, assuming that bracket pairs are situated at the same height. These identified bracket pairs are then merged into a unified node.

After merging, the neighbors of the combined bracket node are sorted into three groups: bracket annotations (characters outside the bracket providing extra information, such as repeat count, assumed to be located at the bottom right of the closing bracket), nodes inside the bracket (fully contained in the bracket's bounding box), and crossing bonds (lines neigther inside nor outside but 'touching' the brackets). Annotations merge with the bracket pair node, while inside nodes and crossing bonds are later used to identify all nodes inside the bracket.

Connecting Bond Node Endpoints

At this stage, the MST lacks recognition of actual line intersection points, including those with characters, which are crucial for identifying atoms within the molecule. While edges indicate intersecting lines, they don't provide position details or identify where the line endpoints intersect. This becomes more complex when more than two lines intersect, a scenario not evident from the edges alone. A solution involves relabeling intersecting line endpoints to share the same position, establishing a common intersection point for those

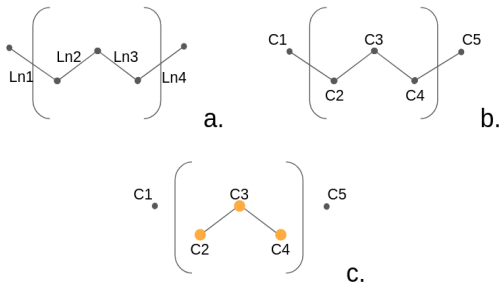


Fig. 5: Finding Bracketed Structures. (a) Visual graph (b) Molecular graph (c) Edges crossing brackets removed; orange dots indicate atoms/superatoms of the bracketed subgraph.

endpoints. This ensures that intersecting line endpoints are perceived as the same 'hidden' carbon or named group, rather than distinct entities.

To acquire this information, we start by annotating the intersection points of all edges. The intersection point between two lines is the midpoint of their closest endpoints, while the intersection point between a line and a character group is the position of the character group. Subsequently, the neighbors of a line node are sorted based on proximity to the first or second endpoint, determining which endpoint the neighbor intersects. This sorting simplifies the identification of attached nodes and their number. Using this information, the relevant calculated intersection points replace the original endpoint positions. For a single neighbor, the calculated intersection point is used; for more than one, the midpoint of related calculated intersection points is employed. The sorting information helps determine the atom type on endpoint nodes, specifically whether it represents a 'hidden' carbon or a named structure. An endpoint with no neighbors or a line neighbor signifies a 'hidden' carbon, while one with a character group neighbor indicates a named structure. This process transforms the modified MST into a dual graph (see Fig. 5 (a) and (b)).

Finding Nodes Inside Brackets.

This step is performed using the dual graph and a dictionary that maps the modified MST nodes to the dual graph nodes. Note that the modified MST nodes are bonds with endpoints that correspond with two dual graph atom nodes that

have an edge. The dictionary is used to map the crossing bond nodes marked during the bracket neighbor sorting to the corresponding atom nodes in the new graph. The edge between the nodes is marked as a 'crossing' edge. Since this is a crossing bond, one atom node will be inside the bracket's bounding box and the other will be outside. The atom node that is inside the bracket's bounding box is marked. A subgraph of the new graph is made, where the 'crossing' edges are filtered out (see Fig. 5 (c)). The subgraph is then broken into a list of connected components. The connected component that has the previously marked node inside it is annotated as the structure inside the bracket. To deal with the case where there are no crossing bonds, the inside nodes of the bracket are used to find which connected component is inside the bracket. In this case, the molecule inside the bracket is already separated from outside components so the 'crossing' edge step can be removed.

4.4 Translating Visual Graphs to CDXML

CDXML Nodes and Attributes: We first classify nodes in the visual graph by node type for use in the CDXML encoding. The most common CDXML node types were: (1) Hidden Carbon Nodes (2), Abbreviation Nodes (3), Atom Nodes, and (4) Unknown Block Nodes. Each node type has a corresponding bond information value as well. To capture the spatial information, visual graph node locations (see Fig. 1) are also encoded in CDXML nodes. This ensures that spatial properties of a molecular diagram used for accurate SMILES conversion are preserved; for example, this allows distinguishing between molecules with different chirality.

Abbreviation Nodes: Abbreviation nodes elide and name portions of molecular diagrams without losing information, provided that the named structure is known to the reader. Fig. 1 shows an abbreviation node NO_2, a nitro group with an external connection available. We used a manually compiled list of 612 common abbreviations along with an abbreviation dictionary from RDKit and ChemDraw for interpretation and then performed CDXML encoding at the atomic level. For the abbreviation NO_2, we insert the full structure $(* \rightarrow N_1, N_1 \rightarrow O_1, N_1 \rightarrow O_2)$ into the CDXML as a 'nested fragment.' $*$ represents where the structure can be connected to other structures; O_1, O_2 represents two oxygen atoms connected to the nitrogen N_1 through a single and double bond respectively.

5 Annotated Data Generation for Visual Graphs

In this section, we introduce a data generation strategy that addresses the crucial issue of obtaining annotated training data for training a visual parser, which is essential for parsing molecules directly from raw images. This data generation strategy presents a significant contribution to the community, as it serves as a viable solution for acquiring annotated training data in scenarios where such data is sparse. Furthermore, the adaptability of this approach to other application domains, broadens its potential impact.

Not all documents are readily available in born-digital form. A substantial number of documents incorporate molecule representations as images, devoid of typesetting instructions. As a result, the extraction of character and graphics information from such documents is impeded, rendering conventional parsing methods ineffective. Our ChemScraper system, tailored for parsing molecule diagrams, faces limitations in processing such documents, prompting the need for an alternative approach – a visual parser capable of extracting molecules from raw images. However, the development of such a visual parser necessitates a robust training dataset.

A key challenge is the paucity of training data in the chemical domain with atom and bond-level annotations, including precise coordinates and labels. While data is frequently available in the form of raw SMILES representations, these representations lack the comprehensive information required for training a visual parser. Even MOL files, although containing some data about atoms, bond types, and relative atom coordinates, fall short of providing the exact atom coordinates from the input images. Moreover, they do not encompass all primitive labels and coordinates, restricting themselves to main atoms and excluding detailed labels and coordinates for primitive constituents, such as missing labels and coordinates for `H` and `3` in CH_3 This absence of

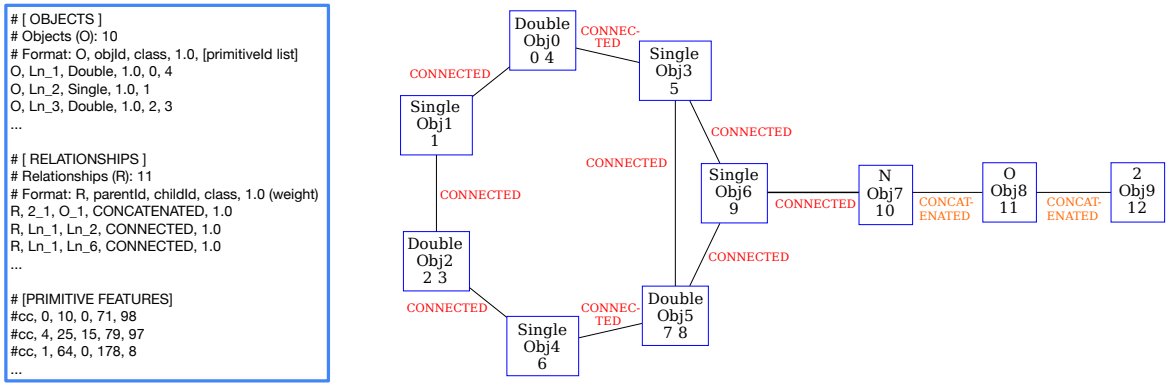


Fig. 6: Annotated data generation for Nitrobenzene ($C_6H_5NO_2$) using visual graph (modified MST) from Fig. 1 (c) with expansion of concatenated atom nodes. (a) Output label graph (lg) file with Object (O), Relationship (R) and primitive bounding box coordinates (b) Equivalent connection graph over atoms/bonds with labels and primitive ids, and edge labels.

detailed data about the visual primitives hampers comprehensive training.

To overcome these limitations, we devised a methodology integrated into the ChemScraper pipeline. In this approach, we employ the Indigo Toolkit to render PDFs from SMILES representations, rather than generating PNG images directly, as done by previous methods like MolScribe [29]. These rendered PDFs are then transformed into 300 DPI images, constituting the training images for the visual parser. The crucial step is the annotation of these training images, a process facilitated by our SymbolScraper [36]. This tool extracts character and graphics elements, providing detailed information including labels, coordinates, and additional geometrical properties of the shapes, as mentioned earlier. ChemScraper leverages these annotations to extract visual graphs from the images.

For training the visual parser, the final visual graph produced by ChemScraper is not used. Instead, we employ an intermediate graph structure, which captures all visual objects within the images as nodes and establish connections among them. To this end, we employ the graph structure from Fig. 1 (c), illustrated in step 1 of Fig. 3 and expand the merged character (atom) groups to introduce `CONCATENATED` edges between characters as shown in Fig. 6. This comprehensive graph structure accounts for all primitives and ensures that the parser can be trained to recognize both visual features of atoms and bonds as nodes.

Label Graph (.lg) Files. We create label graph (Lg) files that adhere to the LgEval format [24, 25] (see Section 3.3). These Lg files contain 'Objects' and 'Relationship' entries, along with primitive coordinates. 'Objects' encompass all primitive groups, comprising atom groups and bonds, and provide details about the individual primitives forming them (e.g., individual lines of bonds, and individual characters of the atom groups). These objects have corresponding labels for atom groups (e.g., 'CH3', 'NO2'), constituent atoms (e.g., 'C', 'N', 'O') or bonds ('Single', 'Double', 'Triple', 'Solid Wedge', 'Hashed Wedge'). The atom groups and bond objects also contain the primitive IDs of the constituent primitives (atoms and lines) as shown in Fig. 6 (a). 'Relationship' entries define the edge connections between these objects. It is imperative to note that we validate the bonds between atoms using the adjacency matrix of bond types obtained from the ground truth SMILES through the creation of an MOL object using the Indigo Toolkit. This ensures the creation of accurate label graph files for the ground truth.

Three types of relationship edges are identified: `CONNECTED`, `CONCATENATED`, and `ABSENT`. All edges in the graph carry a `CONNECTED` label, except the edges between the expanded characters of atom groups, which are marked as `CONCATENATED`. `ABSENT` labels denote non-existent edges, which serve as negative examples for training. These Lg files, in conjunction with the input

images, facilitate fine-grained training of a visual parser, enriched with comprehensive information about the primitives and their connections. ChemScraper allows to generate Lg files and images from a list of SMILES strings available in the standard datasets. Our future work will focus on leveraging these datasets to train LGAP (Line-of-Sight Graph Attention Parser) [37], originally designed for parsing mathematical formulas.

The approach outlined for data generation holds significant potential for broader applicability across various domains. SymbolScraper's ability to extract detailed information from born-digital documents can be leveraged to alleviate the scarcity of training datasets for neural models in diverse fields. This approach serves as a valuable method for addressing the challenge of obtaining annotated training data in scenarios where manual annotation is unfeasible, thus making it a valuable contribution to the scientific community. Its versatility allows for potential extensions to other application domains with similar data constraints.

6 Evaluation

In this Section, we evaluate the accuracy of our born-digital parser and explore its strengths and limitations. We also benchmark the system against existing molecular recognition systems, but it is important to remember that the ChemScraper parser utilizes different information than standard image-based visual parsers as input. Our model produces graph-based outputs stored in CDXML, or translated to SMILES, but the CDXML files contain additional visual and stereochemical information missing in standard SMILES strings.

Datasets. For parameter tuning, we used a subset of the MolScribe training set, which was extracted from the PubChem database. For evaluation of robustness using different rendering parameters, we used the USPTO dataset which contains a list of 5179 SMILES strings that we convert to PDFs using the Indigo Toolkit. For benchmarking against other systems, we evaluated on the public datasets UOB (5,740 molecules) and CLEF (992 molecules).

Implementation/Systems. Runs were made on a Ubuntu 20.04 server, with a 64-core Xeon Gold 6326 (2.9 GHz) CPU and 256 GB RAM. A run took on average 167 seconds for the USPTO-Indigo dataset, with an asymptotic run-time complexity of $O(n^2)$, where n is the number of nodes (PDF character/graphics primitives) in the input graph. A run uses on average a peak of 182 MB of memory for the USPTO-Indigo dataset. SymbolScraper is implemented in Java (based on Apache's PDFBox), while the ChemScraper born-digital parser is implemented in Python, making use of libraries including `Shapely` (for 2d geometry), `networkx` (for graphs and graph operations), `numpy`, and `mr4mp` for parallelization of parsing and other operations using map-reduce. The full processing pipeline is python-based.

6.1 Representations and Metrics

For evaluation, we adopt the common practice of evaluating molecular structure recognition using normalized SMILES strings. We also compute the Tanimoto similarity between molecular fingeprints describing molecular structure. Finally, we introduce a novel approach, where chemical structures are represented and compared using labeled graphs using LgEval library (see Section 3.3). This approach provides a more direct measurement of graph differences and concrete insights into the specific errors made by our parser.

We describe each of the metrics we use with each of these representations below.

SMILES Strings

ChemScraper CDXML files are translated to SMILES using ChemAxon's `molconvert` tool. Given that the order of atoms in SMILES can vary between strings representing the same molecule (see Section x3.2), we canonicalize both predicted and ground truth SMILES using the RDKit library, converting SMILES strings to a canonical form using a built-in function (`CanonSmiles()`, with `ignore_chiral=False`).

Exact Matches are the standard metric for evaluating molecular diagram structure recognition. It is effectively the recognition rate based on SMILES string output.

Normalized Levenshtein Similiarity computes a similarity based on a string edit distance, i.e., the minimum number of insertions, deletions, or substitutions needed to convert one SMILES string to the other [35]. This distance is normalized to $[0, 1]$ based on the minimum and maximum

number of possible edits (from the string lengths). This value is subtracted from 1 to produce a similarity metric. We report average normalized Levenshtein similarity over a test set.

Limitations. SMILES string-based evaluation metrics have inherent limitations for evaluating molecular formula parsing. Molecular formulas are most naturally represented as graphs, where atoms and bonds have well-defined relationships and spatial arrangements. In contrast, SMILES representations are linear sequences of characters that describe graph structure, but SMILES characters have no direct connection with the atoms and bonds present in an input image (i.e., where individual atroms appear in the diagram is not represented).

Recognizing these limitations, we have turned to additional graph-based evaluation metrics to assess accuracy and diagnose errors systematically. A Levenshtein distance only counts operations to convert SMILES strings, and the editing sequences may be non-unique. Ultimately, SMILES-based metrics do not identify which specific parts of the input were recognized incorrectly, or how.

overcome the shortcomings of string-based metrics and obtain a more fine-grained and comprehensive performance evaluation.

Molecular Fingerprints

Molecular fingerprints are bit vectors representing neighboring structures of nodes. We use RDKit fingerprints, a topological representation based on the Daylight fingerprint[7] that encode paths of the molecule graph by varying the path length in a given range, and then constructing a fixed-size binary vector indicating which structures (paths) are present in a given molecule[3]. In our case, the fingerprint vectors have a size of 2048, and path lengths used range from 2 to 7, the default values provided in RDKit.

Tanimoto Similarity. The Tanimoto coefficient [41] measures how similar 2 sets A and B are by computing their intersection over union, that is, the ratio between the number of common objects and the sum of all the objects in both sets. For the molecular fingerprints which are binary vectors, the calculation of the Tanimoto similarity between 2 fingerprint vectors $\vec{u}$ and $\vec{v}$ is given by:

$$Ts(\vec{u}, \vec{v}) = \frac{\vec{u} \cdot \vec{v}}{|\vec{u}| + |\vec{v}| - \vec{u} \cdot \vec{v}} \tag{1}$$

Tanimoto similarity provides additional structural information over the Levenshtein distance. However, while this analysis is structural, the fingerprints are computed from paths over structures represented in SMILES strings, somewhat abstracts the complete structure of a molecule.

Labeled Graphs

LgEval [24, 25] provides a systematic approach for graph-based evaluation of molecular recognition systems, providing an evaluation of structure recognition directly at the primitive (e.g., character), object (e.g., label), and relationship (e.g., bond) levels in graphs.

Label graphs offer a mechanism to calculate an absolute difference between two structural representations, allowing for the assessment of discrepancies even when the segmentation of input primitives (e.g., a series of atom characters) into objects (e.g., an atom group) differs, and even when some primitives are missing in one of the two interpretations. This disparity is directly quantified by contrasting node and edge labels and computing associated Hamming distances, which tally the mismatches in node and edge labels. It is important to note that input primitives are considered to be a fixed and indivisible; this requires that the input matches or over-segments target objects (e.g., atoms, bond line groups). Fortunately this is naturally the case for our PDF character and graphic primitives produced by SymbolScraper.

The LgEval library also offers visualization tools for errors in label graphs, at both the primitive and object levels (the graph-based *confusion histogram tool,* `confHist`). This tool facilitates the examination of specific errors, encompassing missing relationships and nodes, segmentation discrepancies, symbol and relationship classification inaccuracies - essentially, any classification, segmentation, and relationship error. These errors are made easily accessible through HTML pages.

We report detection metrics from LgEval as f-measures at the symbol (atom/node), relationships (bonds/edges), and molecule levels for chemical structure graphs. These entity detection measures are denoted by *DET.* We also report

[7] https://www.daylight.com/

f-measures for correctly labeled *and classified* entities of each type, denoted by *+CLASS*. These two groups report structural correctness for unlabeled (*DET.*)and labeled (*+CLASS*) graphs.

6.2 SMILES-Based Evaluation

Parameter tuning. From Table 1 we tuned the parameters that have a higher influence on the results (according to our experiences developing the tool). We defined an exploration range (which is indicated next to each parameter in the following listing between {}), choose a default value (which is indicated in bold in the following listing) and explored around that range; for each of the parameters, we fix the default values of the remaining parameters and vary the current parameter, we selected the highest of these combination as final values. The final values selected for all the parameters are indicated in Table 1.

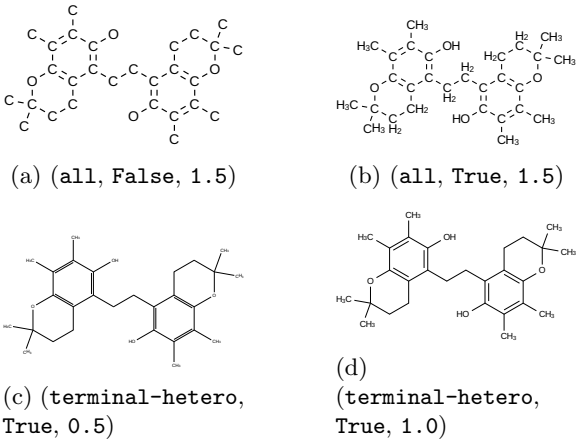
These parameters, the order in which they are searched and value ranges are:

REMOVE_ALPHA {2.0, 2.2, **2.4**, 2.6, 2.8, 3.0},
NEG_CHARGE_Y_POSITION {0.1, 0.2, **0.3**, 0.4, 0.5},
NEG_CHARGE_LENGTH_THRESHOLD {0.3, 0.4, **0.5**, 0.6},
Z_TOLERANCE {1.5, **1.6**, 1.7, 1.8, 1.9, 2.0},
CLOSE_NONPARALLEL_ALPHA
{1.5, 1.6, 1.7, **1.8**, 1.9, 2.0}, and
CLOSE_CHAR_LINE_ALPHA {**1.5**, 1.6, 1.7, 1.8, 1.9, 2.0}.

We selected 1,000 molecules from the MolScribe training set, which was extracted from the PubChem database. We created a dataset of 9,000 molecules by rendering the mentioned 1,000 molecules with different parameter combinations of the Indigo Toolkit. The resulting values of this tuning are used in the consequent runs.

Effect of rendering parameters. Since the datasets we are using contain just SMILES strings, and we need a PDF as input, we use the Indigo toolkit to generate PDFs from those strings. To test the robustness of our parser, we used different PDF rendering parameters, that affect how the molecules look as shown in Fig. 7. The parameters used are:

- `relative-thickness`: Boldness of all graphic and text objects in the molecule. Using the values {0.5, 1, 1.5}. The default is 1.
- `render-implicit-hydrogens-visible`: Show or not implicit hydrogens, {True, False}. The default is True.



(a) (`all`, `False`, `1.5`)

(b) (`all`, `True`, `1.5`)

(c) (`terminal-hetero`, `True`, `0.5`)

(d) (`terminal-hetero`, `True`, `1.0`)

Fig. 7: Same molecule with different rendering parameters (Indigo toolkit). Each sub-caption indicates the label mode, whether implicit hydrogens are shown, and relative thickness, respectively. Parameters in 7c are the defaults. ChemScraper parses all four versions correctly.

- `render-label-mode`: Which labels of the atoms to show,{none,hetero,terminal-hetero,all}. *all* shows all the atoms in the molecule, *terminal-hetero* shows heteroatoms, terminal atoms, atoms with radical, charge, isotope, explicit valence, and atoms having two adjacent bonds in a line, *hetero* is the same as terminal-hetero, but without terminal atoms and *none* does not show any label [8]. We omit the *none* option because it leads to ambiguous molecules. The default is *terminal-hetero*.

This produced a total of 18 combinations for rendering. We evaluated our parser in each of them for the Indigo dataset (USPTO SMILES rendered by Indigo Toolkit.

Fig. 8 shows how the different types of atom labels affect the performance of the parser. We can observe that having all the atom labels performed worse, this is because the more dense becomes the molecule, the more probable it is for the parser to connect atoms incorrectly.

Fig. 9 shows the effect of rendering molecules with different thicknesses. There is a tendency that the lower the thickness, the better. This is again related to the density of the molecule; as

[8] https://lifescience.opensource.epam.com/indigo

shown in Fig. 7, the lower thickness makes graphical objects that must not be connected farther from each other, decreasing the probability of incorrectly connecting atoms.

Initially, the parser struggled with these parameter variations, such as very thick lines, leading to a performance drop to 0% exact matches in certain conditions. This was because, previously, for closing edges in the MST, we used multiple parameters linked to a percentage of the longest bond lines, which varied with thickness as seen in Fig. 7. To address this, we reevaluated and replaced such parameters by incorporating information from the MST, such as node degree, nearest neighbors, and structural attributes. This shift not only enhanced the parser's resilience but also significantly increased the number of exact matches – from 0% to 80%, demonstrating its adaptability to diverse and challenging molecule rendering parameters.

6.2.1 Benchmark

To compare against other systems, we used the default rendering parameters of the Indigo Toolkit. It is worth mentioning that we obtained additional exact matches using a different combination of rendering parameters, but we compared using the defaults for fair comparison. Table 2 compares ChemScraper and existing molecule parsing models. In part, because we have more information available (from PDF instructions) than other benchmark models, we outperform them. This is a good sign that our model can be used for data generation to enhance existing and future visual parsers working from raster images. Note that the percentage of exact matches in the CLEF-2012 dataset is lower, in part because 71 SMILES could not be rendered into PDF by the Indigo Toolkit. Something similar happened with the USPTO (Indigo) dataset, where 15 SMILES strings were empty.

6.3 Graph-Based Evaluation Results

Qualitative & Quantitative Analysis. For fine-grained evaluation of ChemScraper, we require molecule graph representations for both ground truth and the predicted molecules. Given we have already created chemical structure graphs subsequently converted to CDXML format, we can readily employ these graphs for evaluation. However – it is important to note that the graph utilized for evaluation slightly differs from the one used in the data annotation process for creating visual parser training data.

The predicted graph corresponds to the final stage in the parsing algorithm, shown in Fig. 1 (d) generated during Step 2 of the parsing process (see Fig. 3). This graph assumes the representation of atoms or atom groups as nodes, these nodes are portrayed as edges with associated bond types. The bond type each bearing an atom or superatom label, such as N of NO_2, and bonds between could be

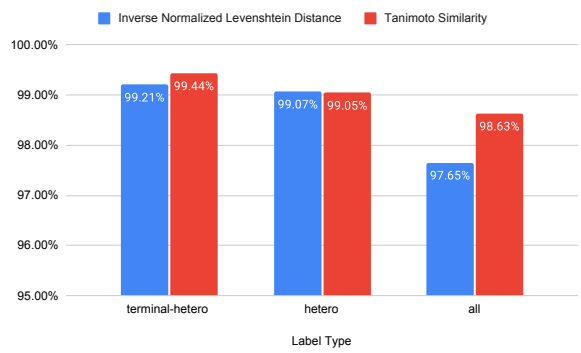


Fig. 8: Effect of using different label types. This run is made using the default parameters of Indigo (`render-implicit-hydrogens-visible` to True and `render-relative-thickness` to 1).

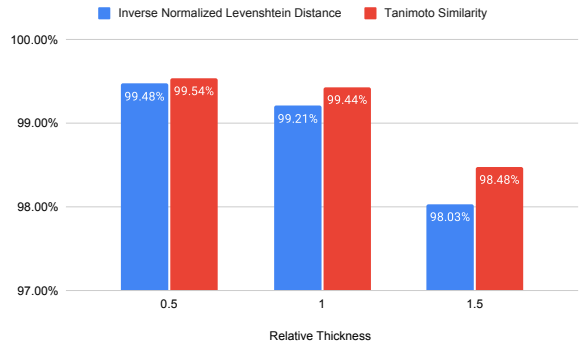


Fig. 9: Effect of using different thicknesses. Higher thickness leads to more parsing errors. This run is made using the default parameters of Indigo (`render-implicit-hydrogens-visible` to True and `render-label-mode` to terminal-hetero).

Table 2: SMILES-based benchmarking of ChemScraper against other molecule parsing models. Percentages shown are for exact matches in SMILES strings. Note: ChemScraper is evaluated on synthetic data, and uses information from PDF; other systems parse from pixel-based raster images (e.g., PNG).

		Synthetic	Real	
Models		Indigo (5719)	CLEF-2012 (992)	UoB (5740)
Rule-based	MolVec 0.9.7	95.40	83.80	80.60
	OSRA 2.1	95.00	84.60	78.50
	Imago 2.0	-	68.20	63.90
Neural Network	Img2Mol	58.90	18.30	78.18
	DECIMER	69.60	62.70	88.20
Graph Outputs	OCMR	-	65.10	85.50
	SwinOCSR	74.00	30.00	44.90
	Image2Graph	-	51.70	82.90
	MolScribe	**97.50**	88.90	87.90
	MolGrapher	-	**90.50**	**94.90**
		Synthetic (SMILES → PDF Using Indigo Toolkit)		
	ChemScraper			
	(PDF render errors)	(15) **97.90**	(71) 84.27	(0) **95.45**
	*Skipping render errors	**98.16**	**90.77**	**95.45**

Table 3: LgEval Metrics for two different runs for the Indigo Dataset (5719 molecules). Shown are f-measures (the harmonic mean, 2RP/(R+P) for Recall and Precision) for correct detection, and correct detection+classes (labeling) for symbols, relationships, and complete molecule graphs.

Runs	Rendering Parameters			Symbols		Relationships		Molecules	
	`label_mode`	`implicit_hydrogens_visible`	`relative_thickness`	Det.	+Class	Det.	+Class	Struct.	+Class
Default	terminal-hetero	true	1	99.97	99.97	99.92	99.53	98.62	88.32
Weakest	all	true	1.5	99.49	99.39	98.54	98.50	79.86	79.33

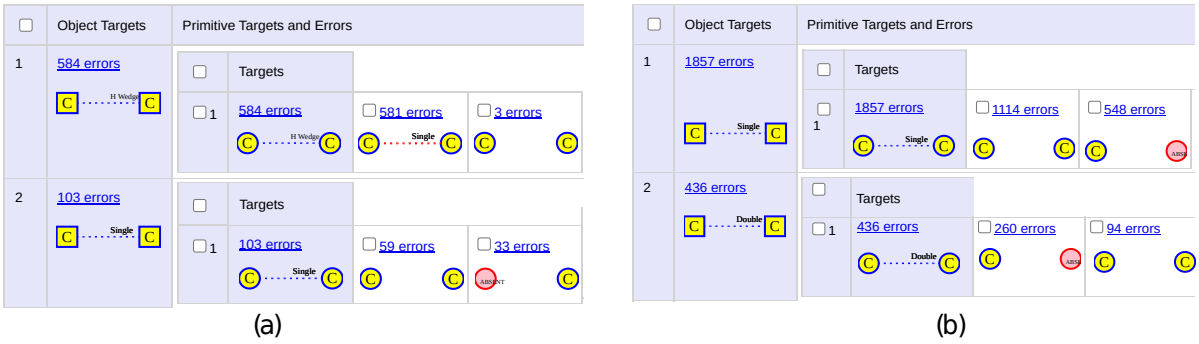


Fig. 10: Confusion Histogram results showing the most frequent relationship errors for (a) Default run and (b) Weakest run in Table 3

one of the following: {`Single, Double, Triple, Solid Wedge, Hashed Wedge`}. To construct a comparable ground truth graph, we leverage the Indigo Toolkit from a `MOL` object using the ground truth SMILES representation. We then extract the graph, including atom coordinates, labels, and an adjacency matrix capturing bonds between atoms. This extraction is facilitated using MolScribe [29] with minor modifications. The adjacency matrix employs values ranging from 1 to 6 to signify bond types {`Single, Double, Triple, Aromatic, Solid wedge, Hashed wedge`}. It is noteworthy that the solid wedge and hashed wedge bonds are functionally identical, but oriented in opposite directions: for example, if there exists a solid wedge bond from 'C' to 'N', there will be a corresponding hashed wedge bond from 'N' to

'C'. All other bonds are undirected. We establish correspondences between the nodes in the two graphs using atom coordinates extracted from the Indigo Toolkit (ground truth) and Symbol Scraper (predicted graph). Minor discrepancies in atom coordinates are resolved using minimum distances between corresponding atom pairs.

Finally, we create object-relationship (`(OR)`) label graph (Lg) files as described in Section 3.3. In this context, 'Object' entries represent individual atoms or atom groups, and the 'Relationship' entries denote bond edges with bond type labels between the atoms, as opposed to specifying the type of connections between visual elements.

The metrics in Table 3, illustrate a disparity the molecule recognition rate (last column) and exact SMILES matches shown in Table 2. This arises because SMILES string-based metrics lack sensitivity to direction and errors for 3D bonds, such as hashed and solid wedge bonds. In this way, SMILES exact matches may be misleading in terms of identifying correct molecular structures. In contrast, our graph-based metrics readily identify and highlight such errors. For example, the first row of Fig. 10 (a) shows hashed wedges incorrectly identified as single bonds.

LgEval played a significant role in identifying errors during our development. Through an analysis of confHist results, we discovered a notable issue: our system incorrectly predicted the direction of solid wedges, causing numerous errors where solid wedges were mistakenly identified as hashed wedges. The insights from confHist allowed us to locate and address the specific part of our system with a bug related to solid wedge direction. This example highlights the utility of LgEval in conducting fine-grained analyses and improving system accuracy. This capability sets LgEval apart from SMILES-based metrics, which yield identical exact matches despite this underlying issue.

Table 3 show a large decline in molecule recognition rates for the weakest run, despite only a 1% reduction in relationship-level metrics. This is mainly due to the intricate network of edges and relationships, particularly in large structures with rings. Even a 1% error in relationships, as seen in the Indigo dataset with 382,058 target relationships for 5,719 molecules, substantically affects accuracy. In confHist (Fig. 10), prevalent errors for the default run involve predicting hashed wedge bonds as single bonds or overlooking them, with occasional missing single bonds. The weakest run exhibits a notable increase in errors, particularly in detecting single and double bonds. This unexpected difficulty with supposedly easier-to-detect bonds is attributed to the inherent complexity of molecules in the weakest run, featuring short bond lines and a compact structure (See Fig. 7 (b)). Such cases pose challenges for graph transformation algorithms in accurately detecting bonds or establishing correct connections between entities, emphasizing the need for more complex visual deep neural-based models.

7 Conclusion

In this paper, we introduce the ChemScraper born-digital molecular diagram parser, along with an improved tool for extracting characters and graphics from PDF (SymbolScraper) and applying our parser to data generation. Conversion of the molecule structure graphs to CDXML was chosen as an intermediate format as it can be ingested by common chemical drawing tools (ChemDraw, Marvin) as well as be converted to other machine-readable formats (SMILES, MOL, and InChI).

Our graph-based evaluation metrics, coupled with the use of LgEval tools, offer a detailed assessment of our parser's performance. This methodology extends beyond chemical diagrams, proving valuable for parsers handling diverse graph-based outputs, such as charts and road networks. The current limitations exist in tackling visually intricate molecules and ensuring robustness across varying rendering parameters, as well as parsing directly from raw images. These challenges underscore the need for enhanced visual parsers. Our annotated data generation tool provides a resource for training sophisticated visual parsers, and we plan to leverage it to train our visual parser for parsing raster images.

References

[1] Baker, J.B., Sexton, A.P., Sorge, V.: A linear grammar approach to mathematical formula recognition from PDF. In: Carette, J., Dixon, L., Coen, C.S., Watt, S.M. (eds.) Intelligent Computer Mathematics, 16th Symposium, Calculemus 2009, 8th International Conference, MKM 2009, Held as Part of CICM 2009, Grand Bend,

Canada, July 6-12, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5625, pp. 201–216. Springer (2009). https://doi.org/10.1007/978-3-642-02614-0_19, https://doi.org/10.1007/978-3-642-02614-0_19

[2] Campos, D., Ji, H.: IMG2SMI: Translating Molecular Structure Images to Simplified Molecular-input Line-entry System pp. 1–12 (2021), http://arxiv.org/abs/2109.04202, arXiv: 2109.04202

[3] Cereto-Massagué, A., Ojeda, M.J., Valls, C., Mulero, M., Garcia-Vallvé, S., Pujadas, G.: Molecular fingerprint similarity search in virtual screening. Methods **71**, 58–63 (2015). https://doi.org/https://doi.org/10.1016/j.ymeth.2014.08.005, https://www.sciencedirect.com/science/article/pii/S1046202314002631, virtual Screening

[4] ChemAxon: Marvin suite version 22.12.0 (marvin sketch + molconvert) (2022)

[5] Comelli, P., Ferragina, P., Granieri, M.N., Stabile, F.: Optical Recognition **44**(4), 627–631 (1995), iSBN: 0818649607

[6] Eto, Y., Suzuki, M.: Mathematical formula recognition using virtual link network. In: ICDAR. pp. 762–767. IEEE Computer Society (2001)

[7] Filippov, I.V., Nicklaus, M.C.: Optical structure recognition software to recover chemical information: OSRA, an open source solution. Journal of Chemical Information and Modeling **49**(3), 740–743 (2009). https://doi.org/10.1021/ci800067r

[8] Han, K., Xiao, A., Wu, E., Guo, J., XU, C., Wang, Y.: Transformer in transformer. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (eds.) Advances in Neural Information Processing Systems. vol. 34, pp. 15908–15919. Curran Associates, Inc. (2021), https://proceedings.neurips.cc/paper/2021/file/854d9fca60b4bd07f9bb215d59ef5561-Paper.pdf

[9] Hanwell, M.D., Curtis, D.E., Lonie, D.C., Vandermeersch, T., Zurek, E., Hutchison, G.R.: Avogadro: an advanced semantic chemical editor, visualization, and analysis platform. Journal of Cheminformatics **4**(1), 17 (Dec 2012). https://doi.org/10.1186/1758-2946-4-17, https://jcheminf.biomedcentral.com/articles/10.1186/1758-2946-4-17

[10] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2015), http://arxiv.org/abs/1512.03385

[11] He, K., Zhang, X., Ren, S., Sun, J.: Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **37**(9), 1904–1916 (2015). https://doi.org/10.1109/TPAMI.2015.2389824, publisher: IEEE

[12] Heller, S.: InChI – the worldwide chemical structure standard. Journal of Cheminformatics **6**(S1), 1–9 (2014). https://doi.org/10.1186/1758-2946-6-s1-p4

[13] Heller, S.R., McNaught, A., Pletnev, I., Stein, S., Tchekhovskoi, D.: InChI, the IUPAC International Chemical Identifier, vol. 7. Journal of Cheminformatics (2015). https://doi.org/10.1186/s13321-015-0068-4, http://dx.doi.org/10.1186/s13321-015-0068-4, publication Title: Journal of Cheminformatics Issue: 1 ISSN: 17582946

[14] Ibison, P., Jacquot, M., Kam, F., Neville, A.G., Simpson, R.W., Tonnelier, C., Venczel, T., Johnson, A.P.: Chemical Literature Data Extraction: The CLiDE Project. Journal of Chemical Information and Computer Sciences **33**(3), 338–344 (1993). https://doi.org/10.1021/ci00013a010

[15] Informatics, P.: Chemdraw professional. (2012)

[16] Kaggle: Bms-molecular-translation (2021)

[17] Krenn, M., Häse, F., Nigam, A., Friederich, P., Aspuru-Guzik, A.: Self-referencing embedded strings (SELFIES): A 100% robust molecular string representation. Machine Learning: Science and Technology **1**(4), 045024 (2020). https://doi.org/10.1088/2632-2153/aba947, arXiv: 1905.13741

[18] Landrum, G.: Rdkit: Open-source cheminformatics (2010)

[19] Li, Y., Chen, G., Li, X.: Automated Recognition of Chemical Molecule Images Based on an Improved TNT Model. Applied Sciences **12**(2), 680 (Jan 2022). https://doi.org/10.3390/app12020680, https://www.mdpi.com/2076-3417/12/2/680

[20] Lowe, D.M.: Extraction of chemical structures and reactions from the literature. University of Cambridge (Doctoral Thesis) (Jan 2012). https://doi.org/10.17863/CAM.16293

[21] Matsakis, N.E.: Recognition of Handwritten Mathematical Expressions. Master's thesis, Massachusetts Institute of Technology (1999)

[22] McDaniel, J.R., Balmuth, J.R.: Kekule: OCR-Optical Chemical (Structure) Recognition. Journal of Chemical Information and Computer Sciences **32**(4), 373–378 (1992). https://doi.org/10.1021/ci00008a018

[23] Morin, L., Danelljan, M., Agea, M.I., Nassar, A., Weber, V., Meijer, I., Staar, P., Yu, F.: MolGrapher: Graph-based Visual Recognition of Chemical Structures (Aug 2023). https://doi.org/10.48550/arXiv.2308.12234

[24] Mouchère, H., Viard-Gaudin, C., Zanibbi, R., Garain, U., Kim, D.H., Kim, J.H.: ICDAR 2013 CROHME: Third International Competition on Recognition of Online Handwritten Mathematical Expressions. In: 2013 12th International Conference on Document Analysis and Recognition. pp. 1428–1432 (Aug 2013). https://doi.org/10.1109/ICDAR.2013.288, iSSN: 2379-2140

[25] Mouchère, H., Zanibbi, R., Garain, U., Viard-Gaudin, C.: Advancing the state of the art for handwritten math recognition: the CROHME competitions, 2011–2014. International Journal on Document Analysis and Recognition (IJDAR) **19**(2), 173–189 (Jun 2016). https://doi.org/10.1007/s10032-016-0263-5, https://doi.org/10.1007/s10032-016-0263-5

[26] Nguyen, A., Huang, Y.C., Tremouilhac, P., Jung, N., Bräse, S.: Chemscanner: extraction and re-use(ability) of chemical information from common scientific documents containing chemdraw files. Journal of Cheminformatics **11**, 77 (12 2019). https://doi.org/10.1186/s13321-019-0400-5

[27] O'Boyle, N., Dalke, A.: DeepSMILES: An Adaptation of SMILES for Use in Machine-Learning of Chemical Structures. ChemRxiv pp. 1–9 (2018). https://doi.org/10.26434/chemrxiv.7097960

[28] O'Boyle, N.M., Banck, M., James, C.A., Morley, C., Vandermeersch, T., Hutchison, G.R.: Open Babel: An open chemical toolbox. Journal of Cheminformatics **3**(1), 33 (Dec 2011). https://doi.org/10.1186/1758-2946-3-33, https://jcheminf.biomedcentral.com/articles/10.1186/1758-2946-3-33

[29] Qian, Y., Guo, J., Tu, Z., Li, Z., Coley, C.W., Barzilay, R.: Molscribe: Robust molecular structure recognition with image-to-graph generation. Journal of Chemical Information and Modeling **63**(7), 1925–1934 (2023). https://doi.org/10.1021/acs.jcim.2c01480, https://doi.org/10.1021/acs.jcim.2c01480, pMID: 36971363

[30] Rajan, K., Zielesny, A., Steinbeck, C.: DECIMER: towards deep learning for chemical image recognition. Journal of Cheminformatics **12**(1), 1–9 (2020). https://doi.org/10.1186/s13321-020-00469-w, https://doi.org/10.1186/s13321-020-00469-w, publisher: Springer International Publishing

[31] Ray, L.C., Kirsch, R.A.: Finding chemical records by digital computers. Science **126**(3278), 814–819 (1957). https://doi.org/10.1126/science.126.3278.814

[32] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. pp. 234–241. Springer International Publishing, Cham (2015)

[33] Sadawi, N.M., Sexton, A.P., Sorge, V.: MolRec at CLEF 2012 | Overview and analysis of results. CEUR Workshop Proceedings **1178** (2012)

[34] Schleyer, P.v.R.: Introduction: Aromaticity. Chemical Reviews **101**(5), 1115–1118 (May 2001). https://doi.org/10.1021/cr0103221, https://pubs.acs.org/doi/10.1021/cr0103221

[35] Schulz, K.U., Mihov, S.: Fast string correction with levenshtein automata. International Journal on Document Analysis and Recognition **5**, 67–85 (2002)

[36] Shah, A.K., Dey, A., Zanibbi, R.: A math formula extraction and evaluation framework for pdf documents. In: Document Analysis and Recognition – ICDAR 2021: 16th International Conference, Lausanne, Switzerland, September 5–10, 2021, Proceedings, Part II. p. 19–34. Springer-Verlag, Berlin, Heidelberg (2021). https://doi.org/10.1007/978-3-030-86331-9_2, https://doi.org/10.1007/978-3-030-86331-9_2

[37] Shah, A.K., Zanibbi, R.: Line-of-Sight with Graph Attention Parser (LGAP) for Math Formulas. In: Fink, G.A., Jain, R., Kise, K., Zanibbi, R. (eds.) Document Analysis and Recognition - ICDAR 2023. pp. 401–419. Lecture Notes in Computer Science, Springer Nature Switzerland, Cham (2023). https://doi.org/10.1007/978-3-031-41734-4_25

[38] Skonieczny, S.: The IUPAC rules for naming organic molecules. Journal of Chemical Education **83**(11), 1633–1637 (2006). https://doi.org/10.1021/ed083p1633

[39] Staker, J., Marshall, K., Abel, R., McQuaw, C.M.: Molecular Structure Extraction from Documents Using Deep Learning. Journal of Chemical Information and Modeling **59**(3), 1017–1029 (2019). https://doi.org/10.1021/acs.jcim.8b00669, arXiv: 1802.04903

[40] Sun, P., Lyu, X., Li, X., Wang, B., Yi, X., Tang, Z.: Understanding Markush Structures in Chemistry Documents with Deep Learning. Proceedings - 2018 IEEE International Conference on Bioinformatics and Biomedicine, BIBM 2018 pp. 1126–1129 (2019). https://doi.org/10.1109/BIBM.2018.8621264, publisher: IEEE ISBN: 9781538654880

[41] Tanimoto, T.: An Elementary Mathematical Theory of Classification and Prediction. International Business Machines Corporation (1958), https://books.google.com/books?id=yp34HAAACAAJ

[42] Vaswani, A., Shazeer, N.M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: Neural Information Processing Systems (2017), https://api.semanticscholar.org/CorpusID:13756489

[43] Wang, C.Y., Bochkovskiy, A., Liao, H.Y.M.: Scaled-YOLOv4: Scaling Cross Stage Partial Network. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 13024–13033. IEEE, Nashville, TN, USA (Jun 2021). https://doi.org/10.1109/CVPR46437.2021.01283, https://ieeexplore.ieee.org/document/9577489/

[44] Wang, Y., Zhang, R., Zhang, S., Guo, L., Zhou, Q., Zhao, B., Mo, X., Yang, Q., Huang, Y., Li, K., Fan, Y., Huang, L., Zhou, F.: OCMR: A comprehensive framework for optical chemical molecular recognition. Computers in Biology and Medicine **163**, 107187 (Sep 2023). https://doi.org/10.1016/j.compbiomed.2023.107187

[45] Weininger, D.: SMILES, a Chemical Language and Information System: 1: Introduction to Methodology and Encoding Rules. Journal of Chemical Information and Computer Sciences **28**(1), 31–36 (1988). https://doi.org/10.1021/ci00057a005

[46] Xu, Z., Li, J., Yang, Z., Li, S., Li, H.: SwinOCSR: End-to-end optical chemical structure recognition using a Swin Transformer. Journal of Cheminformatics **14**(1), 41 (Jul 2022). https://doi.org/10.1186/s13321-022-00624-5

[47] Yoo, S., Kwon, O., Lee, H.: Image-to-graph transformers for chemical structure recognition. In: ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 3393–3397 (May 2022). https://doi.org/10.1109/ICASSP43922.2022.9746088