Lecture 3

Chemical Data: Acquisition and Feature Engineering

Contents

High-quality data is essential for successful data-driven research. In this lecture, we discuss how to acquire chemical data and how to process and analyze it at the cheminformatics level, with an emphasis on feature engineering.

1 Chemical Data Acquisition

Chemical data can be obtained from three main sources: (1) laboratory experiments and computation, (2) online databases/datasets, and (3) the published literature.

- **Laboratory data**. This is how new data are generated. When producing new data, it is critical to adopt standardized data formats and consistent data organization practices. We will discuss data publication and data sharing standards in later lectures. However, now that you have learned about data schemas, you should already have a clearer idea of how to organize and store data generated in your own laboratory.
- **Online databases/datasets**. These are the primary sources for most data-driven and ML tasks in chemistry. In this lecture, we will introduce several widely used chemical databases and datasets, and discuss how to access them properly. In most cases, the retrieved data are not curated for a specific research or training task, so data cleaning, filtering, and feature engineering are necessary.
- **Literature**. This source is closely related to laboratory data. Traditionally, new experimental or computational results are reported as tables, figures, or plots in journal publications. Although many modern databases curate data extracted from the literature, it is still common to encounter valuable data that are only available in PDF format. We will briefly demonstrate how such data can be programmatically extracted from published articles.

1.1 Online Databases

A comprehensive list of commonly used online chemical databases and datasets is provided in Appendix A.

When retrieving data from online databases, there are three widely used access modes:

1. single (interactive) downloads
2. bulk downloads
3. official APIs (application programming interfaces).

Single-download options are offered by most databases and are suitable for retrieving a small number of molecules or structures. However, this approach is inefficient for large-scale data acquisition. Bulk downloads and API-based access provide more scalable and automated solutions, and are therefore preferred for data-driven research.

In the tables in Appendix A, we provide direct links to bulk download pages and documentation describing the API conventions for each database, when such options are available.

1.1.1 API Access

An **Application Programming Interface (API)** is a set of rules and protocols that allows software applications to communicate, exchange data, and share functionality. APIs can be classified as:

- **Private APIs**: require a unique token or API key to authenticate requests. These are common for commercial databases, such as DrugBank.
- **Public APIs**: can be accessed without an API key, but may impose limits on request volume. Examples include major databases such as PubChem, ChEMBL, and ChEBI. Some public APIs may still require user registration before access, such as ChemSpider and ChEBI.

A commonly used API style is **REST** (REpresentational State Transfer), which is widely adopted by chemical databases. We will demonstrate a short example in the lab session.

Example 1: PubChem

PubChem provides the PUG-REST service, a RESTful API. A detailed explanation is available on their official API documentation: `https://pubchem.ncbi.nlm.nih.gov/docs/pug-rest-tutorial#section=How-PUG-REST-Works`. A recipe of composing the request url is in the section *Input: Design of the URL*.

Through the API, you can access PubChem data including molecules, bioassays, genes, proteins, pathways, taxonomies, and cell lines. For most chemistry applications, you only need to know the URL patterns (**endpoints**) for querying specific compounds. Below are some examples (append the url provided after `https://`).

- By compound name (replace `<compound_name>` with the desired name):

  ```
  pubchem.ncbi.nlm.nih.gov/rest/pug/compound/name/<compound_name>/JSON
  ```

- By CID (PubChem ID):

  ```
  pubchem.ncbi.nlm.nih.gov/rest/pug/compound/cid/<cid>/JSON
  ```

- A set of molecules within a certain range:

  ```
  pubchem.ncbi.nlm.nih.gov/rest/pug/compound/molecular_weight/range/<min>/<max>/cids/JSON
  ```

The last segment (`JSON`) specifies the returned format. Other supported formats include `SDF`, `XML`, `CSV`, `PNG`, and `TXT`.

Example 2: Protein Data Bank (PDB)

For PDB database, programmatic access is documented here: `https://www.rcsb.org/docs/programmatic-access/web-apis-overview`. You should look for how to compose the querry

url using `https://search.rcsb.org/index.html#search-api`

To download a protein structure, append the following URL after `https://` (replace <PDB_ID> with the desired PDB code):

```
files.rcsb.org/download/<PDB_ID>.pdb
```

1.1.2 Database-specific Python Libraries

Several Python libraries provide convenient wrappers for interacting with specific chemical databases. Below we show some examples with sample codes.

- `pubchempy`: query PubChem [documentation] [examples]

```python
import pubchempy as pcp
compound = pcp.get_compounds('aspirin', 'name')[0]
print(compound.molecular_formula)
```

- `chembl_webresource_client`: query ChEMBL [examples]

```python
from chembl_webresource_client.new_client import new_client
molecule = new_client.molecule
mols = molecule.filter(pref_name__iexact='aspirin')
print(mols)
```

- `nistchempy`: query NIST Chemistry WebBook [documentation]

```python
import nistchempy as nist
results = nist.run_search("acetone", "name")
print(results.compounds[0].formula)
```

- `rcsb-api`: RCSB PDB documentation

```python
from rcsbapi.search import TextQuery
q1 = TextQuery(value="Hemoglobin")
for rId in q1():
    print(rId)
```

Note: these libraries are primarily wrappers around API calls or HTTPS requests. While convenient for specific databases, they require installation and familiarity with each library's syntax.

For general-purpose data acquisition across multiple sources, it is often more flexible to understand how to make *direct API requests* using standard Python packages such as `requests`.

1.2 Curated Training Datasets

As data science evolves, another important source of chemical data is **curated training datasets**. These datasets are typically derived from online databases but have been cleaned and organized for specific ML/AI tasks.

Examples include the QM9 [2] and GuacaMol [1] datasets. They are usually accompanied by a publication and an online link for downloading. General-purpose repositories such as Zenodo (`https://zenodo.org/`) and Hugging Face (`https://huggingface.co/`) also host curated chemical datasets suitable for ML applications.

Another approach is to follow recent big-data-related research in your field. The open-source culture in the ML community encourages sharing both code and datasets on platforms such as GitHub, GitLab, or Bitbucket. Researchers often download data from online databases and clean it for their own modeling purposes. If your research task is similar, starting from these curated datasets can save time. For example, the CDVAE package (`https://github.com/txie-93/cdvae/tree/main/data`) provides curated datasets used to train a materials design model.

1.3 Data Extraction from the Literature

Programmatic extraction of data from PDF documents can be useful for exploratory data collection or small-scale curation. However, PDF files are designed for visual presentation, not structured data storage. As a result, automated extraction can be *imperfect and may introduce errors*.

Extraction methods generally fall into three categories:

- **Text-based extraction**: parses selectable text directly from PDFs. Common Python packages: `pdfplumber`, `PyPDF2`, `fitz/PyMuPDF`.
- **Table extraction**: reconstructs tabular structures from page layouts. Common Python packages: `pdfplumber`, `camelot`, `tabula-py`.
- **Image-based extraction** (OCR, Optical Character Recognition): converts scanned images of tables or figures into text. Common Python packages: `pytesseract`, `easyocr`, `opencv-python` (for preprocessing).

In practice, extracted data should always be manually validated or cross-checked against the original publication or an independent data source. For large-scale or high-accuracy applications, relying on curated databases is strongly preferred over direct PDF extraction.

We will demonstrate a brief example in the Lab session.

1.4 Ethical and Responsible Practices

When acquiring chemical data, it is essential to follow ethical and responsible practices:

1. **Terms of use**: always respect the database or website terms of use.
2. **Downloading manners**: use official APIs or bulk downloads whenever possible. Avoid unregulated web scraping, which may violate terms and risk your IP being blocked.
3. **Rate limiting**: most websites impose limits on queries per IP per day. Apply rate limiting and batch queries to avoid overloading servers.
4. **Cite the source**: include database version, access date, or DOI to ensure reproducibility and give proper credit to data curators.
5. **Record provenance**: clearly indicate whether the data came from an API, bulk download, or another legitimate source.

1.5 File Formats

In Lecture 1, we discussed data schemas. In chemistry, several widely accepted schemas are associated with specific file types. When downloading data from online databases, you will encounter either *chemistry-specific* file formats or *general-purpose* data formats.

Just as different software applications are needed to open files with different extensions, in Python there are specialized functions and packages to handle these files. Below we summarize common file extensions, their usage, and the corresponding Python tools for loading them.

1.5.1 Chemistry-specific File Formats

A table of common chemistry-specific file formats, their typical usage, and loading methods in Python is provided in Table 1.

1.5.2 General Data File Formats

Table 2 summarizes common data file formats, their typical usage, and how they can be loaded in Python.

Most of these files can be categorized as either **flat (tabular)** or **hierarchical (nested)** data structures. Some formats are text-based, meaning you can open and read them directly, while others are binary and require loading into Python to access their contents.

For example:

- `.csv/.tsv` and `.xls/.xlsx` store flat/tabular data (like a single table). They can be represented as a one-layer dictionary or a list of rows and are human-readable.
- `.json` stores hierarchical or nested data (multiple layers of dictionaries and lists) and is human-readable. Binary formats such as `.pkl` (pickle), `.h5/.hdf5`, and `.parquet` also store hierarchical data. These formats can include complex Python objects, such as an

Table 1: Common Chemistry-Specific File Formats

Suffix	Usage	Opening Apps	Python Modules
.sdf	Multiple small-molecule structures with properties	ChemDraw, Avogadro, PyMOL	RDKit, OpenBabel, Pandas
.mol	Single small-molecule structure	ChemDraw, Avogadro	RDKit, OpenBabel
.mol2	Molecule with atom types, charges, 3D coordinates	Sybyl, AutoDock, Avogadro	RDKit, OpenBabel
.xyz	3D Cartesian coordinates only (QM input/output)	Avogadro, VMD	OpenBabel, ASE, PySCF
.cube	Electron density or molecular orbital cube grid	GaussView, VMD, Avogadro	PySCF, Py3DMol
.pdb	Macromolecule structures (proteins, nucleic acids, ligands)	PyMOL, Chimera, VMD	Biopython, py3Dmol, MDAnalysis
.cif	Crystal structures / lattice info	Mercury, VESTA, Chimera	ASE, pymatgen
.pdbqt	Docking ligand (PDB + torsions)	AutoDock, PyMOL	OpenBabel
.cml	XML-based chemical markup	ChemDraw, Web browsers	RDKit, OpenBabel, lxml

ML model (`.pkl`) or large numerical datasets (`.h5`, `.parquet`) that are optimized for performance.

- Because flat/tabular data is a special case of hierarchical data, file formats designed for hierarchical structures can also store flat data.

2 Feature Engineering

Feature engineering transforms raw data into a machine-readable format. A *feature*—also called a dimension—is an input variable used by a ML model directly.

In Lecture 2, we introduced different representations of chemical systems. These representations can already be considered as the earliest and most fundamental layer of feature engineering. A flowchart illustrating the role of feature engineering in ML tasks is shown in Fig. 1.

A standard feature engineering workflow begins with selecting an appropriate *representation* of the chemical systems. For example, in Lecture 2 we introduced 1D, 2D, and 3D representations of molecules.

Next, these representations are converted into vectors (or higher-order tensors) that are directly readable by ML models. This process is called *vectorization*. For instance, a graph representation

Table 2: Common Data Science File Formats

Suffix	**Data Type**	**Usage**	**Python Modules**
.csv/.tsv	Tabular, plain text	Store columns of data	pandas, numpy, csv
.xls / .xlsx	Tabular, Excel sheets	Structured datasets with multiple sheets	pandas, openpyxl, xlrd
.json	Hierarchical / nested	API responses, nested descriptors, structured metadata	json, pandas
.parquet	Columnar binary	High-performance storage for large datasets	pandas, pyarrow
.h5 / .hdf5	Hierarchical binary	Large structured datasets, scientific data	h5py, pandas
.pkl	Serialized Python object	Save Python objects, preprocessed data, ML models	pickle, joblib
.feather	Columnar binary	Fast read/write for tabular data	pandas, pyarrow
.npy	A single Numpy array, binary	Store matrices	numpy
.npz	Multiple Numpy arrays, binary	Store matrices with descriptors	numpy
.db / .sqlite	Relational database tables	Local storage, complex queries without loading all data	sqlite3, SQLAlchemy, pandas

of a molecule can be transformed into three matrices: the feature matrix X, the edge index matrix, and the edge attribute matrix. At this stage, the features are already ML-readable.

Optionally, one can perform *dimension reduction* to remove redundancy among features, which can enhance ML performance. We will cover dimension reduction techniques in future lectures.

In this lecture, we introduce another vectorization approach widely used in cheminformatics: **molecular fingerprints**. Unlike standard vectorization in general ML, fingerprints encode chemical intuition into the feature vectors. The advantage is that fingerprints often allow for shorter feature vectors while retaining meaningful chemical information. The downside is that this approach may sometimes overlook rare or unusual features (outliers) that could be important.

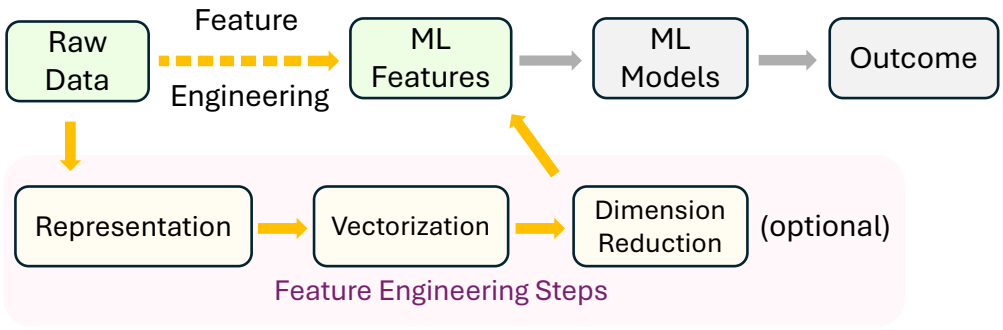

Figure 1: Feature engineering workflow: from raw chemical data to ML-ready features.

2.1 Fingerprints

Fingerprints are a type of molecular representation that encode the presence or absence of specific substructures or features within a molecule as a binary or hexadecimal[1] string. They are widely used in cheminformatics for tasks such as similarity searching, clustering, and classification of chemical compounds.

To understand the concept, we can create a simple, illustrative fingerprint for molecules.

2.1.1 Demo: CHEM542-8 Fingerprints

We define a toy fingerprint called **CHEM542-8**, which uses an 8-bit binary vector to represent each molecule. Each of the 8 bits corresponds to a specific feature (1 = present, 0 = absent):

1. Isotope
2. Aromatic ring
3. At least one halogen atom
4. H-bond donor[2]
5. H-bond acceptor[3]
6. Double or triple bond
7. Formal charge $\neq 0$
8. More than one ring

For example, the CHEM542-8 fingerprints of two common molecules are:

- Chlorobenzene 

: [0, 1, 1, 0, 0, 1, 0, 0] $\rightarrow$ `01100100`

[1] Hexadecimal string has 16 digits: 0, 1, . . . , 9, A, B, . . . F, with A = 10, etc.
[2] H-bond donor: –X-H, where X = N, O, or S
[3] H-bond acceptor: an electronegative atom with at least one available lone pair

- Acetone  : [0, 0, 0, 0, 1, 1, 0, 0] → `00001100`

These binary strings are the vectorized form of the molecular features, ready for cheminformatics or ML tasks.

In practice, 8-bit fingerprints are far too small to capture the complexity of chemical space. Next, we introduce standard fingerprints used in real applications.

2.1.2 Standard Fingerprints

- **Structural fingerprints**: encode the presence or absence of specific substructures, functional groups, or atom patterns. *Examples*: MACCS keys (166-bit), PubChem fingerprints, custom SMARTS-based substructure keys.
- **Topological fingerprints**: encode the connectivity and arrangement of atoms in the molecule (graph structure). *Examples*: ECFP / Morgan fingerprints (circular, neighborhood-based), topological torsion fingerprints, atom pair fingerprints, Daylight fingerprints (path-based).
- **Pharmacophore fingerprints**: encode the spatial arrangement of functional groups important for biological activity (e.g., H-bond donors/acceptors, aromatic centers, charges, hydrophobic sites). *Examples*: UNITY, MOE pharmacophore fingerprints, atom environment pharmacophore (AEP) fingerprints, shape- or field-based fingerprints (e.g., ROCS color-shape).

The `RDKit` package provides functions to generate most of these fingerprints, which we will explore in the Lab session.

3 Chemical similarity

One important task in cheminformatics is determining the similarity of molecules.

Let's see an example of drug discovery. Starting from one or a set of known inhibitors, one can perform similarity evaluation of a larger pool of molecules, and pick those with high similarity (drug-like molecules), forming the initial set of candidates.

Starting from these candidates, one can directly perform more rigorous screenings by metrics such as pharmaproperties and binding affinity. However, with generative model, we could go further. We can train a generative model to further learn the hidden patterns in the candidates, and generate novel molecules that resemble these hidden patterns. The generated molecules will be put into high throughput virtual screening (HTVS) to find more potential inhibitors. This approach has been standard in AI-driven novel molecular generation. A pipeline is shown in Fig. 2.

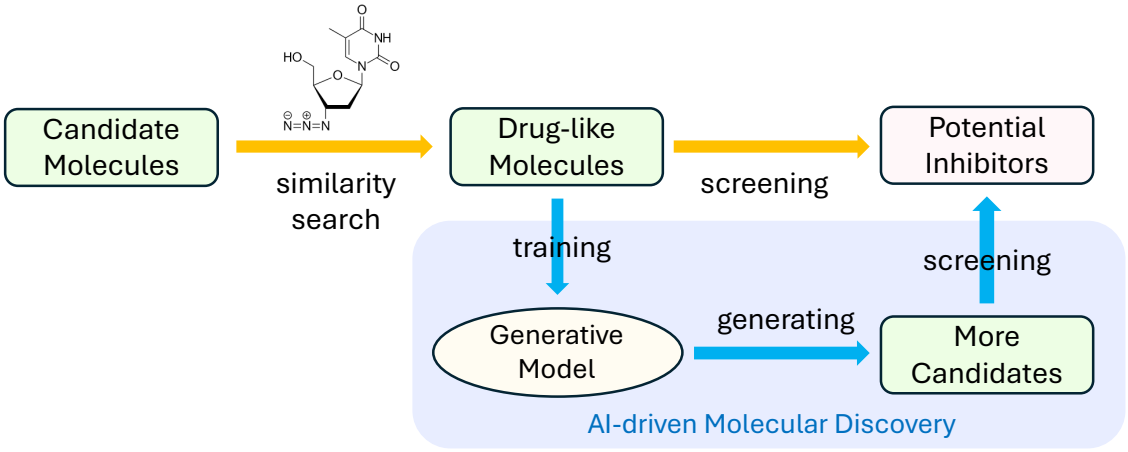

Figure 2: Similarity-based inhibitor design. The target molecule is Zidovudine (AZT), a known HIV inhibitor.

3.1 Fingerprint-based similarity

The **Tanimoto metric** is used to compare the fingerprints of molecules. The Tanimoto metric between molecule A and B is defined as

$$T = \frac{c}{a + b - c}, \tag{1}$$

where a = number of 1s in molecule A, b = number of 1s in molecule B, and c = number of 1s that occur in the same positions in both A and B. $T = 1$ means full similarity. $T = 0$ means no similarity.

Now we can evaluate the Tanimoto similarity using our CHEM542-8 fingerprints. Let's still look at the chlorobenzene and acetone

- Molecule A (clorobenzene): `01100100`.
- Molecule B (acetone): `00001100`.

In the above setting, $a = 3$, $b = 2$ and $c = 1$, therefore

$$T(A, B) = \frac{1}{3 + 2 - 1} = 0.25, \tag{2}$$

which indicates low similarity between the two molecules.

RDKit package provides the function to evaluate the Tanimoto similarity given the fingerprints of the two molecules, which is included in this lecture's Lab session.

3.2 Descriptor-based similarity

In addition to binary fingerprints, molecules can also be represented by numerical descriptors that capture physicochemical and topological properties. Common examples include molecular weight, log P[4], topological polar surface area (TPSA)[5], the number of hydrogen bond donors and acceptors, etc.

Similarity between molecules in descriptor space is evaluated using standard distance or similarity metrics, such as Euclidean distance, cosine similarity, and Mahalanobis distance. Unlike fingerprint-based similarity, descriptor-based similarity emphasizes global molecular properties rather than detailed structural patterns.

- Euclidean distance (requires rescaling)

$$d_E(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_i (x_i - y_i)^2} \tag{3}$$

- Cosine similarity (scale-invariant)

$$d_c(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} \in [-1, 1] \tag{4}$$

- Mahalanobis distance

$$d_M(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T \Sigma^{-1} (\mathbf{x} - \mathbf{y})}, \tag{5}$$

  where Σ is the covariance matrix estimated by the data set. The covariant matrix measures how do different descriptors co-vary (change together), which is a way to see the correlation between different descriptors. It will be explained in the next lecture.

Example

Now let's evaluate the similarity between Molecule A and B using the aforementioned metrics. The two vectors of Molecule A and B are

$$\mathbf{x}_A = [180.2, 1.2, 63.3, 1, 4], \quad \mathbf{x}_B = [194.2, 1.5, 66.5, 1, 4]. \tag{6}$$

[4] $\log P$: Logarithm of the partition coefficient, measures a molecule's lipophilicity (fat-loving) or hydrophilicity (water-loving).

[5] TPSA: the surface sum over all polar atoms or molecules, primarily oxygen and nitrogen, also including their attached hydrogen atoms.

Table 3: Example molecular descriptors for two molecules

Descriptor	Molecule A	Molecule B
Molecular weight (g/mol)	180.2	194.2
log P	1.2	1.5
TPSA (Å^2)	63.3	66.5
H-bond donors	1	1
H-bond acceptors	4	4

We evaluate the Euclidean and cosine distances, and leave the covariant one for later.

- Euclidean distance (`numpy.linalg.norm(diff)`).

$$\begin{aligned} \mathbf{x}_A - \mathbf{x}_B &= [14.0, 0.3, 3.2, 0, 0] \\ d_E(\mathbf{x}_A, \mathbf{x}_B) &\approx 14.36 \end{aligned} \tag{7}$$

  We can see that because we did not rescale the descriptors, d_E is dominated by the molecular weight.

- Cosine distance.

$$d_c(\mathbf{x}_A, \mathbf{x}_B) \approx 0.999967, \tag{8}$$

  which indicates high similarity.

4 Labs

In this session, we have two labs. The following Python packages are introduced:

- `requests`: a widely used library for making HTTP requests. We will use it to programmatically query online chemical databases.
- `json`: for reading and writing JSON files.
- `numpy`: for handling vectors, matrices, and higher-order tensors.
- `seaborn`: a scientific plotting library, complementary to `matplotlib`.

4.1 Lab 3-1: API Access

In this lab, we will demonstrate how to access online databases via APIs. We will work with three types of data files: `SDF`, `PDB`, and `JSON`, and learn how to download, read, and process them in Python. Link: `https://colab.research.google.com/drive/1D7Vj3KtW9zqdbkLRVobmmw0DGTVvZGSI?usp=sharing`

4.2 Lab 3-2: Chemical Similarity

In this lab, we will generate molecular fingerprints and evaluate similarity between molecules using both fingerprint-based (Tanimoto) and descriptor-based metrics. This will give hands-on experience in feature engineering and similarity evaluation in cheminformatics.

Link: `https://colab.research.google.com/drive/1pM4o01qT9GbDaTvn22jg0XP5jZ9_SGCY?usp=sharing`

A A long table of online databases/datasets (to be finished)

Note: explicit URLs are not provided for a compact view.

Table 4: Online Databases for Chemistry (I)

Databases	**Description**
General Chemistry	
PubChem (Bulk)(API)	Structures, properties, bioassays, identifiers
ChemSpider (API)	Structures and linked chemical data
ChEBI (API)	Biologically relevant small molecules
Drug-like Molecules	
ZINC	Commercially available drug-like compounds
PubChem	Subset for bioactive molecules
ChEMBL (Bulk)(API)	Bioactive molecules with activity data
DrugBank	Approved and experimental drugs with targets
Biology	
GenBank (NCBI)	DNA and RNA sequences
UniProt	Protein sequences and annotation
PDB (API)	3D biomolecular structures
Materials Science	
Materials Project	Computed materials properties
AFLOW	High-throughput materials data
COD	Experimental crystal structures
Spectroscopy	
NIST Chemistry WebBook	Spectroscopic and thermochemical data
SDBS	NMR, IR, MS, Raman spectra
MassBank	Mass spectra with metadata
Thermochemical	
NIST TDE	Thermochemical data
JANAF Tables	Experimental thermochemical tables
IUPAC Solubility Data Series	Solubility data for compounds

Table 5: Online Databases for Chemistry (II)

Databases	**Description**
Reaction and Kinetics	
NIST Chemical Kinetics Database	Reaction rate constants and kinetics data
RMG Database	Reaction mechanisms and kinetics for combustion/chemical processes
Reaxys (subscription)	Experimental reactions, properties, and synthesis routes
SciFinder (subscription)	Chemical reactions, literature, and patents
Computational Properties	
Materials Cloud	DFT and computational materials data
NOMAD Repository	Raw and processed DFT calculations
Open Materials Database (OMDB)	Computed properties of inorganic crystals
Polymers and Soft Materials	
PolyInfo	Polymer structures and properties
PoLyInfo (NIMS)	Polymer data from NIMS Japan
Polymer Genome	Polymer property prediction and descriptors
Nanomaterials	
Nanomaterial Registry	Characterization data for nanomaterials
caNanoLab	Nanotechnology characterization and experimental metadata

Table 6: Curated Datasets for Chemistry

Databases	**Description**
Online Repositories	
GitHub	Platform for shared code and curated datasets
Zenodo	Open repository for research datasets
Hugging Face	Hub for ML datasets and models, including chemistry
Curated ML Datasets	
GuacaMol	Benchmark dataset for molecule generation
DFT	
QM9	Quantum chemical properties for small molecules
ML Potential	
SPICE	Drug-like molecules and peptides for training ML potential
QDπ	ML potential developed by York Group

References

[1] Nathan Brown, Marco Fiscato, Marwin HS Segler, and Alain C Vaucher. Guacamol: benchmarking models for de novo molecular design. *Journal of chemical information and modeling*, 59(3):1096–1108, 2019.

[2] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.