Lecture 2

Chemical Data: Representation

Contents

The primary inputs in chemical machine learning are *chemical systems*, including small molecules, polymers, crystals, liquids, and biomolecules, etc. Other forms of chemical information–such as physical properties, reaction rates, or spectroscopic data–are generally treated as derivations of these primary inputs.

In this lecture, we will focus on molecular representation. Understanding how to encode a single molecule provides the foundational framework necessary to model more complex chemical systems.

We will explore the three hierarchies of molecular representation:

- 1D Strings (e.g., SMILES, InChI).
- 2D Graphs (Topological connectivity).
- 3D Geometric Representations (Atomic coordinates).

1 Representing a molecule

When you visualize a molecule, what representation comes to mind first? Most people envision a chemical formula or a skeletal structure. However, while these are chemically intuitive, they possess drawbacks for computational modeling.

The limits of intuition

- **Chemical formulas:** Computer-readable but not unique. For example, the formula C_6H_6 can be represented as a simple string in a computer, but it corresponds to several distinct structural isomers, as shown in Fig. 1.
- **Skeletal formulas:** Structurally unique, but not computational friendly. They are image-based data, and computers cannot "read" a standard figure without complex, and often error-prone, computer vision preprocessing.

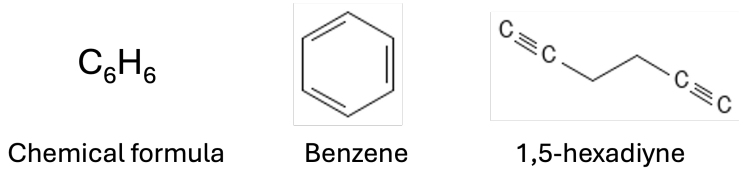


Figure 1: Possible structures of C_6H_6.

Minimal requirements

Therefore, to be effective for machine learning, a chemical representation must meet two minimal requirements:

- **Uniqueness.** A single representation must correspond to exactly one chemical structure. If one representation could mean two different things, the model cannot learn a clear relationship.
- **Machine readability:** The data must be stored in a structured format that a computer can process mathematically.

It is important to note that the requirement for uniqueness is *one-way*: one representation must not represent more than one molecule. However, a single molecule can often be described by

multiple valid representations. For example, as we will see in the section on SMILES, the same molecule can be "spelled" in several different ways depending on where one starts counting atoms.

In practice, the fact that one molecule can map to multiple representations creates redundancy that can confuse a machine learning model. To improve performance, we generally apply constraints or strategies to manage this:

- **Canonicalization:** We use standardized algorithms to choose a unique "canonical" representation for each molecule. This ensures that the input is consistent across the entire dataset.
- **Data Augmentation:** Alternatively, we can feed multiple valid representations of the same molecule into the model during training. This teaches the model that different representations can refer to the same chemical identity, a property known as *representation invariance*.

Next, we will examine some specific hierarchies of representations that fulfill these criteria, starting with 1D strings.

2 1D string representations

To resolve the ambiguity inherent in chemical formulas (as exemplified by the different structures of C_6H_6 in Fig. 1), we must incorporate information regarding the connectivity and order of atoms. This can be achieved through the SMILES (Simplified Molecular Input Line Entry System) encoding.

2.1 SMILES

The SMILES syntax is built upon four elements:

1. **Atoms:** Atoms are represented by their chemical symbols. Standard organic elements (C, N, O, P, S, etc.) can stand alone, while other elements (e.g., metals) or atoms with specific properties (e.g., formal charges or isotopes) are enclosed in brackets, e.g., `[Mg]`. Hydrogen atoms (H) are typically omitted (implicit hydrogens).
2. **Bonds:** Single bonds are implied between adjacent atoms in the string. Double, triple, and aromatic bonds are explicitly represented by symbols: `=`, `#`, and `:`, respectively.
3. **Branches:** Branching is indicated using parentheses `()` around the subordinate group. The group inside the parentheses is attached to the atom immediately preceding the opening parenthesis.
4. **Rings:** Ring structures are represented by breaking a bond and assigning a *matching digit* to the two atoms where the "cut" occurred. This digit follows the atom symbol (e.g., `C1...C1`).

Example 1 The SMILES representation for the two isomers in Fig. 1 are:

- Benzene: C1=CC=CC=C1 (A six-membered ring with alternating double bonds)
- 1,5-hexadiyne: C#CCCC#C (A linear chain with triple bonds at each end)

Example 2: Aspirin Let's examine a more complex case: the aspirin molecule (Acetylsalicylic acid). Because SMILES is a 1D string, we must *flatten* the 2D structure into a 1D chain. As shown in

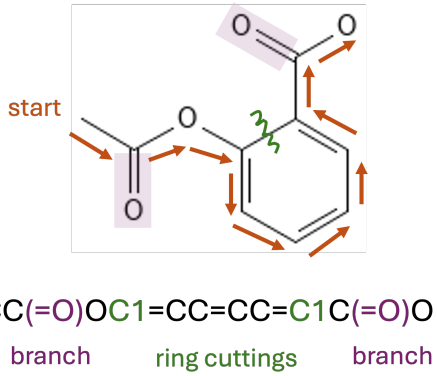


Figure 2: SMILES representation for the aspirin molecule.

Fig. 2, the process follows these steps: (1) The backbone: We identify a continuous path of atoms to serve as the main chain (indicated by the orange arrows). (2) Branching: When the path encounters a functional group that is not part of the main chain (highlighted in purple), we enclose that group in parentheses `()` immediately after the atom to which it is attached. (3) Ring Closure: To represent the aromatic ring, we cut one of its bonds (indicated by the green line). The two atoms that were originally connected are both labeled with the same integer (e.g., 1) to tell the computer to stitch them back together into a cycle.

2.1.1 Obtaining SMILES

While it is important to understand these rules, you do not need to generate complex SMILES strings manually. Several tools can automate the conversion from a 2D drawing to a SMILES string:

1. **ChemDraw:** Draw the structure → select the molecule → Edit → Copy As → SMILES.
2. **PubChem Sketcher:** An online tool where the SMILES string is dynamically generated as you draw (see Fig. 3). `https://pubchem.ncbi.nlm.nih.gov/edit3/index.html`
3. **Chemical Databases:** Most databases (PubChem, ChemSpider, ChEMBL) list SMILES as a standard metadata field.
4. **Programmatic Conversion:** We can convert other molecular formats to SMILES using libraries like RDKit, which we will explore in Lab 2.

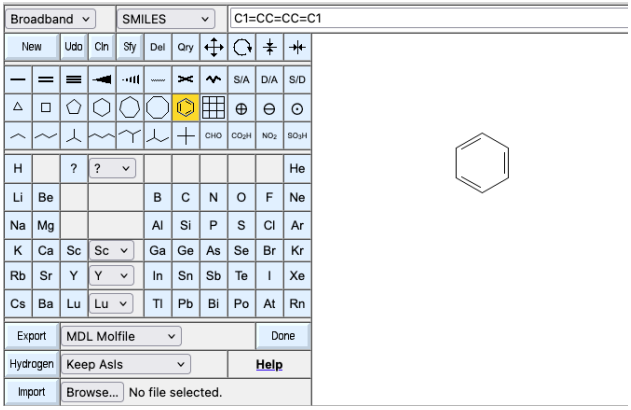
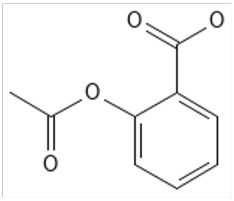


Figure 3: Screenshot of the PubChem Sketcher interface.

2.1.2 Canonical SMILES



CC(=O)OC1=CC=CC=C1C(=O)O

O=C(O)C1=CC=CC=C1OC(=O)C

CC(=O)Oc1ccccc1C(=O)O (canonical)

C1=CC=C(C=C1C(=O)O)OC(=O)C

Figure 4: Valid SMILES for aspirin.

As noted previously, a single molecule can have multiple valid SMILES strings depending on the starting atom and the path taken through the structure (e.g., Ethanol can be `CCO` or `OCC`). In Fig. 4, we list 4 valid SMILES for aspirin.

To eliminate this redundancy for machine learning tasks, we use *canonical SMILES*. A canonicalization algorithm reorders the atoms in a deterministic way so that every unique molecule maps to exactly one unique string. In practice, we use the `rdkit` package to generate these canonical representations automatically.

2.2 SMARTS

A closely related concept to SMILES is SMARTS (SMILES Arbitrary Target Specification). While their syntax is nearly identical, they serve different purposes:

- SMILES is used to describe a complete molecule.
- SMARTS is used to describe a substructure or a molecular pattern.

In practice, SMARTS acts like a regular expression (regex) for chemistry. It allows us to query a large database of molecules to identify those that contain a specific functional group or motif.

Any valid SMILES string can technically be used as a SMARTS query. If you use the SMILES for benzene (`c1ccccc1`) as a SMARTS pattern, the search will return every molecule in your dataset that contains a benzene ring as part of its larger structure.

However, SMARTS allows for much more generalized expressions that SMILES cannot handle, such as:

- Wildcards: Using `[*]` to represent any atom.
- Logical Operators: Using `[C,N]` to represent "Carbon OR Nitrogen."
- Connectivity Constraints. E.g. `[C;D3]` means the carbon atom must have exactly three neighbors.

2.3 SELFIES

A significant limitation of SMILES is its lack of syntactic robustness. Because SMILES relies on specific closing characters (like matching digits for rings or balanced parentheses for branches), a randomly generated string is often chemically "illegal." For example, `C1CC` is invalid because the ring is never closed.

This is a major obstacle for generative ML models, which might spend most of their "effort" learning the grammar of SMILES rather than the actual chemistry. SELFIES (Self-Referencing Embedded Strings) provides a solution by using a grammar where every possible string corresponds to a valid molecule. While we will not cover the inner workings of SELFIES in this course, interested readers should refer to [1] and the official implementation: `https://github.com/aspuru-guzik-group/selfies`.

2.4 Other unique identifiers

While Canonical SMILES is widely used in ML, other identifiers are standard in databases and regulatory contexts:

- **InChI and InChIKey:**
  - **InChI** (International Chemical Identifier) is a hierarchical, layered identifier. It separates information into layers (formula, connectivity, charge, isotopes, and stereochemistry), making it very precise.
  - **InChIKey** is a fixed-length (27-character) string. You can think of the InChIKey as the *index* of the InChI. Because it is compact and contains no “illegal” URI characters (like slashes or plus signs), it is the primary format used for web-based database lookups and URL-friendly indexing.
  - *Example:* The InChI for benzene is `InChI=1S/C6H6/c1-2-4-6-5-3-1/h1-6H`, and its corresponding InChIKey is `UHOVQNZJYSORNB-UHFFFAOYSA-N`.
  - Distinction: An InChI string *contains* the structural information of the chemical (it can be decoded back into a molecule), whereas the InChIKey is effectively the “bar code”—it identifies the record but cannot be decoded without a reference database.

- **CAS registry number:**
  - Assigned by the **Chemical Abstracts Service** (CAS), these numbers are the "social security numbers" of the chemical world.
  - **Note:** Unlike SMILES or InChI, CAS numbers are *not* algorithmic—you cannot determine a molecule's structure just by looking at the number. They are essential for experimental lookup but are not used as direct inputs for ML models.

3 Graph representation

In the previous lecture, we learned about graphs as a data structure. A graph representation of a molecule translates chemical structures into a mathematical format where atoms are treated as **nodes** (vertices) and chemical bonds as **edges**. Unlike 1D strings like SMILES, this 2D representation explicitly preserves the topological connectivity and branching of the molecule.

The mathematical representation of a graph is

$$G = (V, E), \tag{1}$$

where V to represent nodes, and E for edges.

For molecules,

- **Nodes (V):** Represent individual atoms and their properties (element type, charge).
- **Edges (E):** Represent chemical bonds and their properties (bond order, conjugation).

An example is shown in Fig. 5.

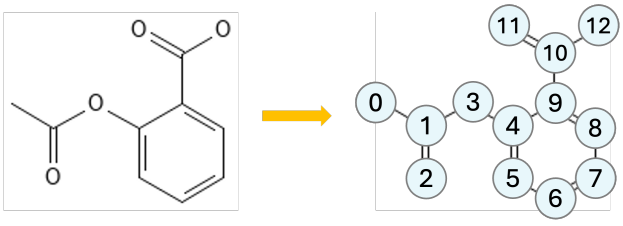


Figure 5: Representing Aspirin as a graph

3.1 Encoding the graph

Suppose we only care about the atomic number and connectivity. For a molecule with N atoms, we encode it as:

1. **Node list:** A list of N integers indicating atomic numbers (e.g., [8, 6, 8] for CO_2).

2. **Edge index:** A list of pairs (i, j) denoting a bond between atom i and atom j.

An example of a graph representation of CO_2 is shown in Fig. 6.

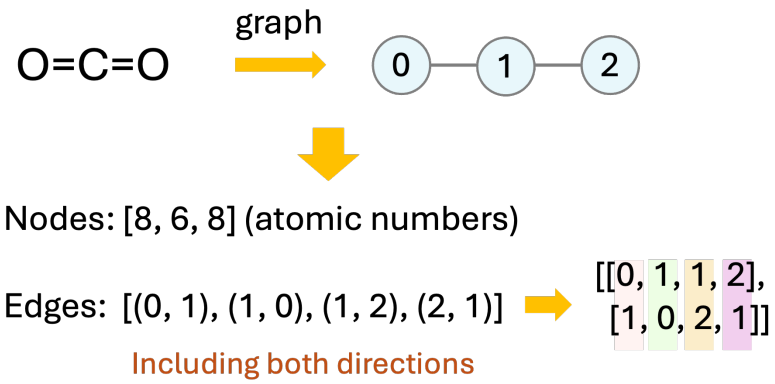


Figure 6: Simple graph representation of the CO_2 molecule.

Note: You might heard the term **adjacency matrix** for representing connectivity, but it is often inefficient. Most molecules are *sparse*—each atom is only bonded to a few neighbors. This means an adjacency matrix would be mostly zeros. Using an **edge index** (listing only active bonds) saves significant memory and computation.

3.2 Adding chemical features

A simple list of integers is rarely enough for high-performance ML models. Let's enrich our graph by attaching *feature vectors* to both nodes and edges.

3.2.1 Node features (The feature matrix X)

Instead of a single integer, we represent each atom as a vector (a list of numbers). Let's use the following three features:

1. **Atomic number:** (e.g., 8 for O, 6 for C).
2. **Hybridization:** (1 for sp, 2 for sp^2, 3 for sp^3).
3. **Formal charge:** The electrical charge on the atom.

For CO_2, each atom becomes a row in a **feature matrix** of shape $N \times 3$:

$$\text{Atoms: } [\text{O, C, O}] \longrightarrow X = \begin{bmatrix} 8 & 2 & 0 \\ 6 & 1 & 0 \\ 8 & 2 & 0 \end{bmatrix} \tag{2}$$

3.2.2 Edge attributes

In addition to connectivity, we can associate bond-specific features with edges, such as the bond order (e.g., 1 for single, 2 for double) and the conjugation status (1 for conjugated, 0 otherwise). Encoding these properties directly into the connectivity pairs can be cumbersome; instead, they are typically stored separately as edge attributes aligned with the edge list.

As an example, consider the CO_2 molecule. In its graph representation, there are four edges corresponding to the two C=O bonds represented in both directions. For each edge, we record the bond order and the conjugation flag. Since both bonds are double and conjugated, the resulting edge feature matrix is

$$\text{Edge Index: } \begin{bmatrix} 0 & 1 & 1 & 2 \\ 1 & 0 & 2 & 1 \end{bmatrix} \begin{matrix} \rightarrow \text{starting node} \\ \rightarrow \text{ending node} \end{matrix}, \qquad \text{Edge Attributes: } \begin{bmatrix} 2 & 1 \\ 2 & 1 \\ 2 & 1 \\ 2 & 1 \end{bmatrix}, \tag{3}$$

where each row corresponds to one edge, the first column denotes the bond order, and the second column indicates conjugation.

3.2.3 Summary on graph representation

Three components are needed: node feature matrix, edge index, and edge attribute matrix.

- Representing atoms (nodes):
  - **Node feature matrix:** A matrix of size $N \times L_v$, where N is the number of atoms (nodes) and L_v is the number of node features. Each row corresponds to the feature vector of one atom.
- Representing bonds (edges):
  - **Edge index:** A $2 \times M$ integer matrix, where M is the number of edges. The first row contains the indices of source nodes, and the second row contains the indices of target nodes.
  - **Edge attribute matrix:** A matrix of size $M \times L_e$, where L_e is the number of edge features. Each row stores the feature vector associated with one edge.

The representation above corresponds to an *undirected graph*, in which each bond is encoded symmetrically. In some applications, however, it is useful to work with *directed graphs*. In that case,

each edge is assigned a direction, and the edge list explicitly records the source node followed by the target node, rather than using the symmetric notation shown in Fig. 6.

4 3D geometric representation

While 2D graphs tell us "which atom is bonded to which," they ignore the reality that molecules are physical objects that exist in 3D space. 3D Geometric Representations account for the actual (x, y, z) coordinates of each atom. This is critical because a molecule's shape determines how it fits into a protein, how it reacts, and its physical stability.

The most direct way to represent a 3D molecule is to store an $N3$ matrix containing the spatial coordinates for every atom. This is the industry standard for chemical storage, most commonly seen in the `.xyz` file format.

Example: XYZ file for methane

```
5
Methane
  H      0.5288      0.1610      0.9359
  C      0.0000      0.0000      0.0000
  H      0.2051      0.8240     -0.6786
  H      0.3345     -0.9314     -0.4496
  H     -1.0685     -0.0537      0.1921
```

How to read this file:

- Line 1: The total number of atoms (5).
- Line 2: A comment line (useful for titles or energy levels).
- Lines 3+: The element symbol followed by its x, y, and z position in Angstroms.

Unlike the Graph representation (where you define the bonds), the XYZ file often contains no explicit bond information. Instead, the model must calculate distances between coordinates to "guess" where the bonds are.

This format serves standard input in many cheminformatics, ML/AI and quantum chemistry tasks. These coordinates can be downloaded from massive databases (like PubChem) or generated computationally using the `rdkit` library's `AllChem` embedding functions.

5 Labs

We have two labs in this lecture. We will introduce following packages:

- RDKit (`https://www.rdkit.org/`)
  An cheminformatics toolkit for molecular representation, manipulation, and analysis, widely used for SMILES parsing, substructure search, and descriptor generation.
- PyTorch (`https://pytorch.org/`)
  A general-purpose deep learning framework providing tensor computation and automatic differentiation, commonly used for building and training neural networks.
- PyTorch-Geometric (PyG, `https://pytorch-geometric.readthedocs.io/en/latest/`)
  A specialized library built on top of `PyTorch` designed to facilitate deep learning on geometrically structured data, such as graphs and point clouds.
- (Extra) Py3DMol (`https://pypi.org/project/py3Dmol/`)
  A lightweight Python interface for interactive 3D molecular visualization in Jupyter and Colab environments.

5.1 Lab 2-1: Chemical representation with RDKit

By the end of this lab, you will be able to:

- Represent molecules using SMILES, canonical SMILES, InChI, and InChIKey.
- Convert between representations programmatically.
- Generate 3D geometries and export XYZ files.
- Visualize molecules in 2D and 3D.
- Use SMARTS to detect functional groups

Link to Google Colab.

5.2 Lab 2-2: Building a graph for molecules

In this Lab, we will look at how to encode graphs of molecules with `PyTorch Geometric`. The link to the Lab is Link to Google Colab.

References

[1] Mario Krenn, Qianxiang Ai, Senja Barthel, Nessa Carson, Angelo Frei, Nathan C Frey, Pascal Friederich, Théophile Gaudin, Alberto Alexander Gayle, Kevin Maik Jablonka, et al. Selfies and the future of molecular string representations. *Patterns*, 3(10), 2022.