Chapter 8

Proposed Work: mCLM: modular Chemical Language Model

Although language models have greatly advanced the capabilities for understanding and designing small molecules, they tend to employ naive tokenization schemes or utilize character-based tokenization of molecules (e.g., SMILES [70] or SELFIES [72]) [8], [67], [296]. Here, we propose a modular chemical language model (mCLM) which is both synthesis- and function-aware. Our model will allow us to work directly with molecular building blocks which can easily be joined together using automated synthesis hardware.

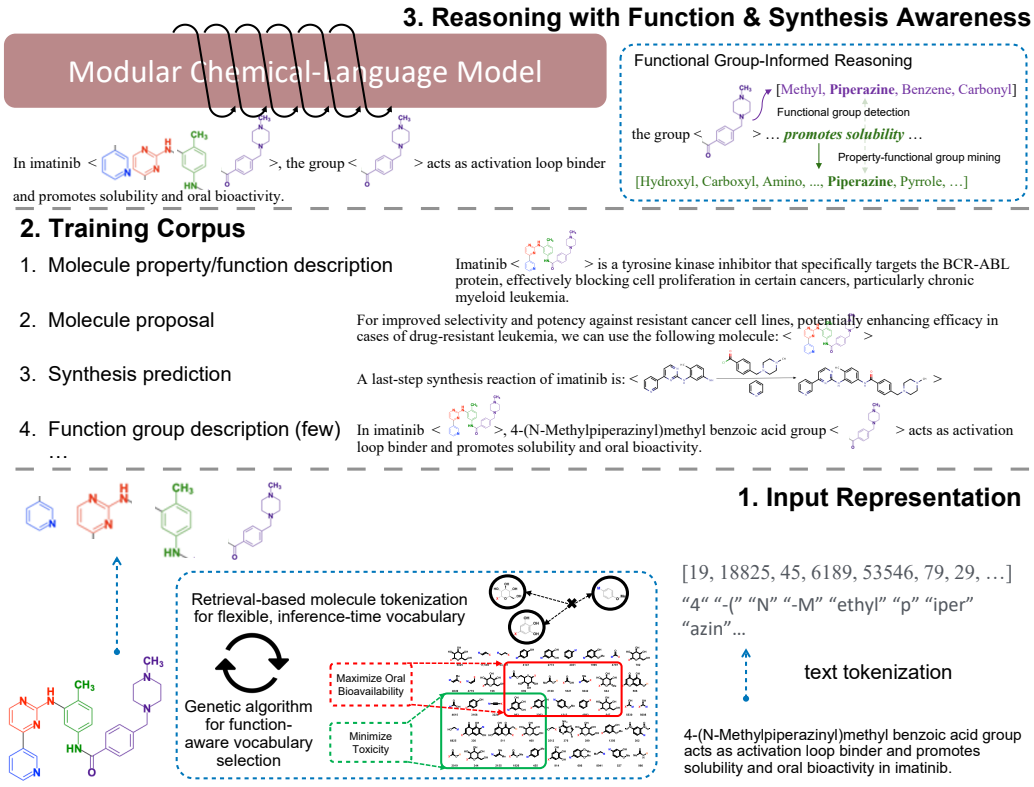

Figure 8.1: An overview of the proposed mCLM. The model 1) accepts both text and molecules as input using a flexible, inference-time molecule token vocabulary, 2) is trained on a variety of synthesis- and function-aware tasks, and 3) can reason about the function of different moieties at inference time. Further, new synthesis rules can be incorporated during inference thanks to the flexible, retrieval-based decoding.

8.1 A flexible, inference-time vocabulary

As part of our ongoing work on kinase inhibitor drug discovery, we have indentified synthesis as a key limiting factor for evaluating new drugs. To alleviate this problem, we plan to train a molecule-language model LM which can flexibly adapt to any possible vocabulary, V. Essentially, we will train a model which can *retrieve* the best molecule token to generate from V. The goal of this approach is to allow our model to easily adapt to different chemistries by working with a different set of tokens V, which can be decided after training the model. Further, V can be modified mid-inference to allow adaption of the allowed synthesis procedures based on the molecule generated so far. As an example, we may want to avoid generating a molecule with too many nitrogens, so we can remove nitrogen-containing blocks from the vocabulary in the middle of inference if there are already too many.

The model will take interleaved text and molecule fragments as input and similarly generate either text or molecules. For each input molecule, we will consider molecule fragments created by breaking apart the molecule. In our initial model, we plan to break molecules along three synthesis-friendly bonds: Suzuki-Miyaura, Buchwald-Hartwig, and Amide. Further, each molecule will be tokenized in a linear fashion; this enables compatibility with automated synthesis machines, but it is also beneficial for integrating with language models, since a linear representation is much easier to model for standard autoregressive language models. Each fragment will be encoded using a GNN G to create a latent representation which can 1) be used as input to the model in place of a typical token, and 2) be used for retrieving the next molecule fragment during decoding. Notably, the usage of a GNN encoder in this setting will allow us to easily consider extending the model to 3D molecular conformations.

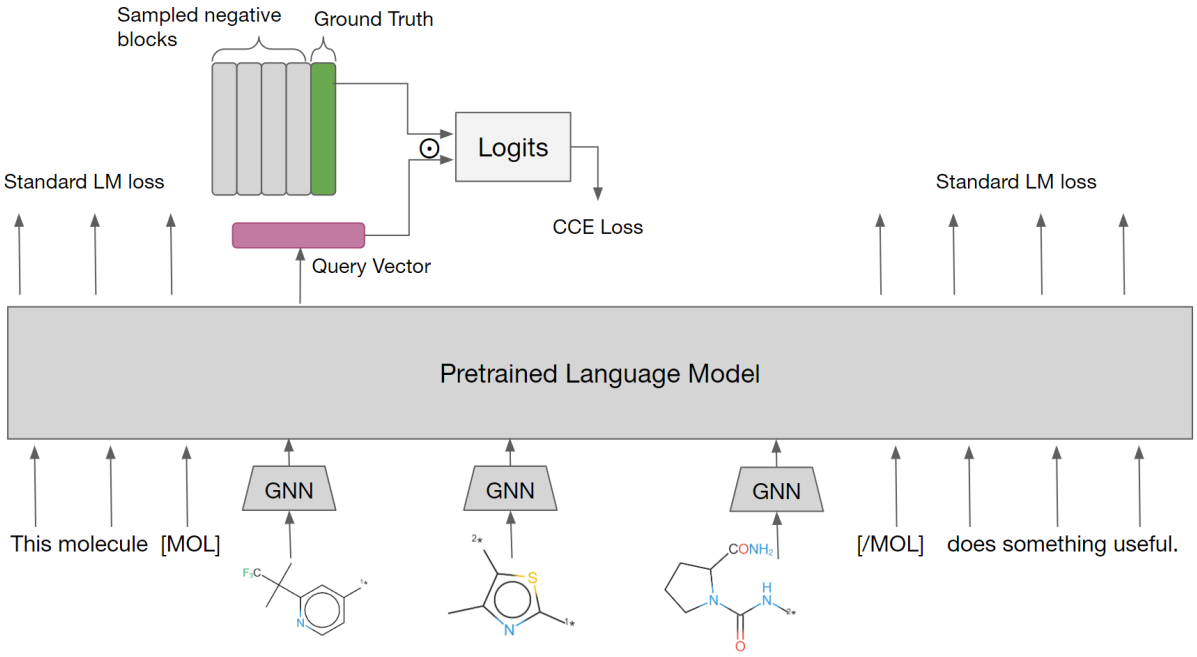

Figure 8.2: The architecture and training procedure for the mCLM.

To train the model, we will adopt the standard text prediction loss for natural language tokens, and a retrieval-based loss for molecule tokens. For some given molecular token t_i, the retrieval loss will be formulated

as follows:

$$L(t_i) = \text{CCE}(Q \cdot G(GT), Q \cdot G(n_1), ..., Q \cdot G(n_j))$$

where $Q = LM(t_i|t_1, ..., t_{i-1})$ is a query vector produced by the model for selecting the molecule token, $G(\cdot)$ is the representation of the molecule token produced by the GNN, GT is the ground truth molecule fragment, and $G_{n_1}, ...G_{n_j}$ are j negative molecule fragments sampled from the training vocabulary. Here, this process can be thought of as similar to contrastive learning.

8.2 Joint molecule and function description representation

To train the model, our goal is to cover a wide variety of different molecular functions and classes of molecules. This will comprise several different data sources, including those developed earlier in this dissertation. This will include several large-scale molecular datasets, including ZINC [153], the natural product database [297], and ChEMBL [298]. For natural language-molecule pairs, we will leverage the large-scale MoMu dataset [9] based on S2ORC [299], the L+M-24 dataset [283], and ChEBI-20 [6]. We plan to construct this model using an arbitrary language model as the "core" with LoRA parameter-efficient finetuning [300]. This will enable us to take advantage of the large-scale, natural language information already learned by the model, saving us computational resources from thousands of GPUs. Currently, we plan to train using the Llama3.1-8B and Llama3.3-70B models [293].

8.3 Function-Aware Molecule Library Design

Finally, after training the mCLM, we will use our model as a supervision signal for identifying motifs and molecular fragments which can be used to maximize coverage of some given molecular functional space while minimizing the required amount of molecular fragments needed to synthesize those molecules. For example, our goal may be to find a set of building blocks which can be combined to produce molecules that inhibit kinases. For the purpose of enabling cheap and rapid synthesis, we want to minimize the number of building blocks required to cover this functional space. Many different molecules can inhibit the same kinases, so we want to select the smallest number of building blocks to produce a set of molecules which does so. Notably, this idea is related to how proteins consist of a small vocabulary (20 amino acids) of building blocks which covers a large functional space.

Past efforts to solve this problem have failed largely due to computational limitations. For example, [301] attempted to cover the natural product database, but the combinatorial, rule-based approach required multi-route retrosynthesis planning for a large number of molecules which limited its feasability. Instead, we plan to use a metaheuristic-based approach using the trained mCLM as a supervision signal. Initially, we will start with a genetic algorithm [218] to select an optimal subset of molecule tokens $V_{opt} \subset V$. Given a function space F consisting of $|F|$ functions, we can use the mCLM to predict candidate molecules M for a given function $f \in F$.

$$M_f = \{m \sim LM(\text{"This molecule has function } f\text{:"})\} \ni |M_f| = h$$

$$M = \bigcup_{f \in F} M_f$$

Here, h is an arbitrary, experimentally determined integer such as 5. We will use an algorithm such as beam

search to sample M_f from the mCLM.

We will then use the mCLM to score these molecules:

$$s(f) = \frac{1}{|M_f|} \sum_{m \in M_f} \sum_{j=1}^{k} -\log\left[G(m_j) \cdot LM(m_j | \text{"This molecule has function } f\text{:" } m_1, ..., m_{j-1})\right]$$

where the molecule m has k fragments $m_1, ..., m_k$. A total score can then be calculated given a set of building blocks $V_s \subset V$:

$$S(V_s, F) = \frac{1}{|F|} \sum_{f \in F} s(f)$$

The genetic algorithm will be used to efficiently find a set V_{opt} of desired size $|V_{opt}| = g$ which maximizes this score.

$$V_{opt} = \max_{V_s \in V} S(V_s, F) \ni |V_s| = g$$

Here, g can be selected experimentally to maximize this score or it can be selected by chemistry experts given a budget for creating g new molecular building blocks for automated synthesis machines.

By taking this approach, we can use the mCLM to optimize our coverage of functional molecular space while avoiding the computational limitations of previous approaches. We note that it will also be possible to maximize coverage of existing datasets (e.g., the natural product database) using this method.