2AMM10 Assignment 2: Molecular Property Prediction

1 Introduction

Professor M. Vlenkovski is a world-renowned chemist, having designed and invented many molecules used in various applications, from drug design to material synthesis. However, the design of these molecules comes at a large expense, as for even a single molecule, thousands of expensive experiments are required. As a result of recent global events, the funding of professor Vlenkovski's lab has dried up, and, to make matters worse, the price of most components used in his experiments has skyrocketted. As such, professor Vlenkovski cannot afford to carry out exthensive experiments for the discovery of new and useful molecules. Desparate to find a solution to keep his lab alive, professor Vlenkovski has made the decision to turn to machine learning.

Through his many years of experience, he has found that molecules which have good properties in theory are often hard to synthesize as they are too unstable. To quantify the instability of a molecule, one useful feature to look at is the formation energy, which is the difference between the total energy of the molecule and the sum of the energies of individual atoms (isolated). Here, a negative formation energy means that the molecule releases energy when it forms (and would require energy to break apart), as a result of which the molecule is more stable. Inversely, a molecule with a higher formation energy is more unstable, as they require energy to form.

Realizing the power of being able to predict the formation energy of a molecule before attempting to synthesize it, professor Vlenkovski initially turned to computational quantum chemistry methods, like Density Functional Theory. However, he quickly found out that these calculation were incredibly expensive, and could take weeks for a single molecule. Thankfully, machine learning methods offer a promising alternative for predicting the formation energy of a molecule, and professor Vlenkovski has gathered a large dataset of molecules and their formation energies over his many years of experience. Since he has no experience in using machine learning, and therefore turned to you, the AI experts, to help him with making a solution.

2 Assignment Description

For his dataset, professor Vlenkovski has two representations of molecules: a geometric one, where the positions and types of each atom are stored, and a Simplified Molecular Input Line Entry System (SMILES) representation, which represents the molecule as a string of text. The dataset consists of 129,012 molecules, of which 10,000 are used as a test set. With the rise of large language models, he is curious which representation will be more powerful, and wants you to analyze both as part of your task. Sadly, professor Vlenkovski is quite paranoid about AI technologies, and therefore expects you to train and develop models from scratch, since he is scared that existing models will steal his data.

2.1 SMILES

SMILES provides a way to represent molecules as text strings. Each atom is represented by its chemical symbol. Atoms that belong to the organic subset—B, C, N, O, P, S, F, Cl, Br, and I—are written without brackets, while all others must be enclosed in square brackets (e.g., `[Na]` for sodium). Bonds between atoms can be written using `-` for single, `=` for double, `#` for triple, and `$` for quadruple bonds. However, in practice, single bonds are often omitted, and hydrogen atoms are typically implicit[1]. For example, both `CC` and `[H]C([H])([H])-C([H])([H])[H]` represent ethane, even though the latter makes all hydrogens and the single bond explicit.

Importantly, SMILES strings are not unique: the same molecule can have multiple valid SMILES representations that differ in the order of atoms and parentheses. For instance, `CCO` and `OCC` both describe ethanol. On the other hand, different SMILES strings using the same atomic symbols can correspond to different molecules; for example, `CCO` (ethanol) and `COC` (dimethyl ether) contain the same atoms but have distinct connectivity and chemical properties.

Additional syntax includes the use of lowercase letters (e.g., `c`) to indicate aromatic atoms, numeric labels to indicate ring closures (e.g., `C1CCCCC1` for cyclohexane) or higher atomic charges (e.g., `[NH4+]` for ammonium), and square brackets for specifying formal charges (e.g., `[NH4+]`). Parentheses `()` are used to indicate branches in the molecule, specifying groups or fragments attached to the main chain. For instance, in the SMILES `CC(C)O`, the `(C)` represents a methyl group branching off the second carbon of the main chain. The `@` symbol is used to encode stereochemistry, which describes the 3D orientation of atoms; for instance, `CC@HC` and `CC@@HC` represent two mirror-image forms of a chiral alcohol.

Professor Vlenkovski uses canonical SMILES to ensure a consistent representation of molecules. Although SMILES notation allows for multiple valid representations of the same molecule, variations can result from differences in atom order, bond placement, and implicit hydrogen atoms. Canonical SMILES addresses this by generating a unique, standardized string for each molecule, applying a specific algorithm that sorts atoms and bonds in a predefined, deterministic way.

Note: Professor Vlenkovski understands that as an AI scientist, you might have difficulty understanding some of the chemical terms presented in the assignment. He encourages you to reach out to his assistants to clarify any misunderstandings and answer your questions.

2.2 Task 1: Formation Energy Prediciton

For the first task, professor Vlenkovski wants you to develop a model for predicting the formation energy of the molecules. A separate model should be created for each of the molecular representations (geometric and SMILES) and analyzed in terms of performance on the provided test set. When developing a model architecture, make sure it is alligned with the properties of that data representation (symmetries, structure, etc.).

2.3 Task 2: Model Efficiency

In addition to predicting the formation energy, professor Vlenkovski would like to use machine learning for a variety of different prediction tasks. For some of these tasks, only very limited training sets are available. As such, professor Vlenkovski wants to have an insight into how data-efficient both of your models are. To this end, he wants you to analyze the performance of both models when training on subsets of the original training data, with varying sizes: 100, 300, 1000, 3000, and 10000 molecules.

[1] When representing a molecule, the amount of hydrogen atoms connected to each atom can be implicitly inferred by inspecting the other bonds of that atom. For example, carbon is tetravalent, meaning it has 4 electrons that can be used to form bonds. In ethene, `C=C`, each carbon has a double bond, resulting in two implicit hydrogens connected to each carbon.

For this task, you should use the model architectures developed in the previous task, for which you should use **two** metrics to compare the data efficiency of the two models on different sized training sets. The two metrics should be different in a non-trivial way. E.g., using MAE and MSE as the two metrics does **not** satisfy the requirement.

2.4 Task 3: SMILES generation

Professor Vlenkovski is keen to see whether SMILES can be used not only to predict molecular properties but also to design new compounds. For this final task he asks you to build a model that generates *valid*, *unique*, and *novel* molecules. A molecule is valid if the generated string complies with SMILES syntax. The uniqueness is defined as the percentage of distinct SMILES strings among the valid molecules. Finally, the novelty is the percentage of valid SMILES strings that do not occur in the training set.

In this task, you should develop a model that can generate SMILES strings, and use it to generate 5000 molecules, for which you should report these three metrics. If helpful, you may reuse or fine-tune any of the weights from the models you developed earlier—Professor Vlenkovski trusts the quality of your previous work.

3 Deliverables

The submission of this assignment is done in ANS. There are two deliverables:

- Implementation of the solution
- Description and explanation of your approach

For the implementation, a skeleton Jupyter Notebook with functions that can load the provided data for each task are provided. The notebooks also contain short bits of self-explanatory code on how the data is formatted and how to load the data in python. Please submit your code in the provided skeleton Jupyter notebook. You need to upload the Jupyter Notebook code (.ipynb file extension) and a PDF version (.pdf file extension) with the execution output. The maximum upload size is 25MB. So, you may need to clean some image output from the code Notebook before uploading. If you are using Google colab, you can generate the PDF by going to "File → Print → Save as PDF". Please generate the PDF after running all cells of your solution (with possibly images removed if necessary for file size constraints).

The description of your approach is submitted as a digital group test in ANS. Rather than an unstructured report, the group test is formatted in a number of questions that you need to answer to describe your solution and understanding of the assignment. Please see ANS for the details.