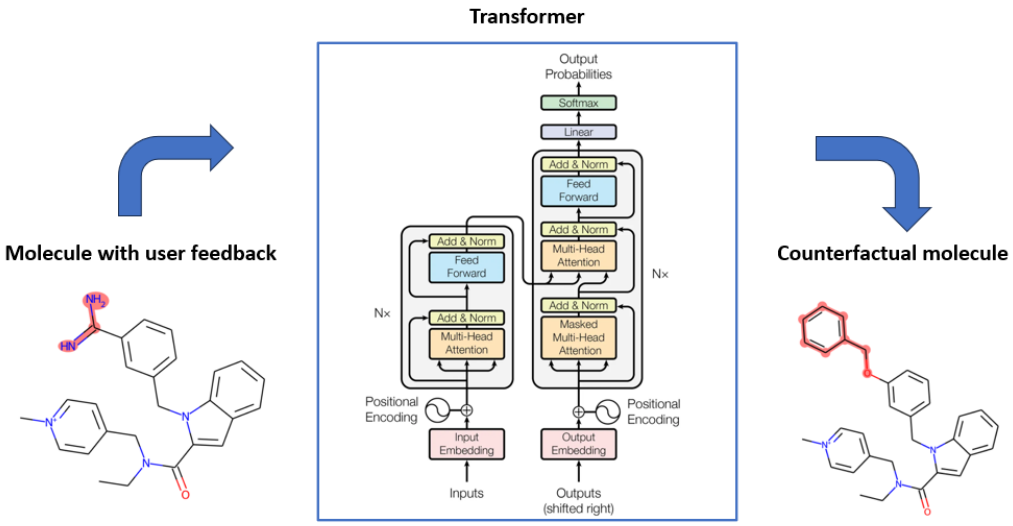
CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG



Substructure Constrained Molecular Generation for Counterfactual Design

Transformer-Based Masked Language Modeling for Constrained Molecular Design in Drug Discovery

Master's Thesis in Computer Science and Engineering

DYLAN OSOLIAN
LISA SAMUELSSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Master's thesis 2024

Substructure Constrained Molecular Generation for Counterfactual Design

Transformer-Based Masked Language Modeling for Constrained Molecular Design in Drug Discovery

DYLAN OSOLIAN

LISA SAMUELSSON

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2024

Substructure Constrained Molecular Generation for Counterfactual Design
Transformer-Based Masked Language Modeling for Constrained Molecular Design in Drug Discovery
DYLAN OSOLIAN, LISA SAMUELSSON

Supervisor: Janosch Menke, Computer Science and Engineering
Industrial Supervisor: Jiazhen He, AstraZeneca
Examiner: Ola Engkvist, Computer Science and Engineering

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: An illustration of a counterfactual molecule being generated by passing a molecule, along with user feedback, through a Transformer model.

Typeset in LaTeX
Gothenburg, Sweden 2024

Substructure Constrained Molecular Generation for Counterfactual Design
Transformer-Based Masked Language Modeling for Constrained Molecular Design in Drug Discovery
DYLAN OSOLIAN, LISA SAMUELSSON
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

In this thesis, masked modeling is used in de novo drug design to create substructure-constrained counterfactual molecules. Utilizing Natural Language Processing (NLP) methods, SMILES strings are processed through a transformer-based architecture. The research explores diverse masking and training strategies, demonstrating the ability to produce a wide range of relevant molecules. These strategies impact the properties of the molecules produced, showing that the choice of masking influences the molecular structures generated. Findings suggest that masking strategies must be carefully chosen based on the models intended use in molecule design. This work highlights the potential of masked modeling to effectively generate diverse molecules and enhance molecular drug design, opening possibilities for further exploration and application.

Keywords: Machine Learning, Natural Language Processing, Cheminformatics, Transformers, Deep Learning, Molecular Design, Drug Discovery, Molecular Generation

Acknowledgements

We want to express our gratitude to our supervisors Janosch Menke and Jiahzen He for their guidance and support throughout this project. Thanks also to our examiner, Ola Engkvist. We are grateful for the valuable input and advice from our colleagues and fellow masters thesis students in our department at AstraZeneca. Lastly, we thank our families and friends for their constant encouragement.

Dylan Osolian & Lisa Samuelsson, Gothenburg, 2024-05-31

Contents

List of Figures

List of Tables

1

Introduction

Drug design focuses on discovering new chemical compounds for treating or preventing diseases. A challenge in drug design is finding molecules that meet multiple criteria to ensure the drug's safety, effectiveness, and commercial viability. Finding such a compound is not trivial, since there are many desirable properties to optimize.

From a computational view, the drug design process can be viewed as a multi-parameter optimization (MPO) process. It is more difficult to find an optimal solution among a set of possible candidates to an MPO problem compared to a one-dimensional optimization problem, as the objective values might conflict [1]. The number of relevant objectives may also be very large [2]. Additionally, the size of the chemical space, consisting of up to 10^{60} possible molecules [3], also contributes to the challenge of discovering satisfactory compounds. Having a method to quickly identify these compounds is of great interest, as it saves time and money.

In recent years, deep learning, a subset of machine learning and artificial intelligence (AI), has proven effective in the drug design process. It accelerates the identification of potential drug candidates, enables the prediction of molecular behavior, and enhances the precision of targeting specific diseases. Multiple deep learning-based approaches have been tested, such as variational autoencoders, recurrent neural networks, and generative adversarial networks [4].

Deep learning is used in various aspects of drug design, such as toxicity prediction and reaction prediction, through tools like DeepTOX and IBM RXN [5], [6]. Another area where deep learning is used is in molecular generation, such as de novo drug design. De novo design tools aim to generate molecules with user-specified properties from scratch. Typically, these models are trained in two stages: In the initial phase, the models are trained to generate valid molecules. Then, in the second stage, reinforcement learning is used to refine the generation, targeting molecules that meet the specific criteria and properties defined by the chemist.

However, in practice, chemists reject many of the generated molecules from these models, even though they are valid and fulfill the specified requirements [2]. Instead, the molecules from the generative models are often used as a starting point for further manual improvement. Given the importance of molecule generation as a part of the drug design process, creating tools to aid chemists in this process is a highly relevant task. In this thesis, we aim to extend the capabilities of existing de novo models to allow for more fluid interactions between the chemist and the model.

This project is conducted in collaboration with AstraZeneca, a global pharmaceutical company. The Molecular AI division at AstraZeneca has developed multiple de novo design tools, and this thesis extends on prior efforts in developing models for molecular generation.

1.1 Problem description

While deep learning methods in de novo design have already seen some success, they are still in the early stages of development and are evolving to meet the specialized demands of drug design. The frequent rejection of AI-generated molecules by chemists underscores the need for enhanced models that can handle the complex demands of chemists. The project aims to explore models that allows chemists to select substructures of molecules for modification, aiming to generate similar molecules but where the selected parts are modified.

In the field of explainable AI, counterfactuals are generally referred to as the smallest change in a model's input needed to obtain a different output [7]. Counterfactual molecules would then be defined as those that closely resemble the original molecule, but are modified in small yet significant ways to yield alternative behavior [8].

In the context of this project, however, the focus lies on a specific application of counterfactuals: adapting molecules based on chemist feedback. Here, the chemist specifies a substructure of the molecule that is undesirable. In this scenario, a counterfactual molecule is a molecule from which this selected substructure has been removed, but the rest of the molecule is kept intact. This approach allows chemists to iteratively refine the original molecule by identifying and modifying substructures considered problematic.

In this project, a transformer model will be used to accomplish this. The transformer architecture, proposed by Vaswani et. al. [9], was developed for natural language processing. It uses so-called self-attention mechanisms, which allows the model to focus on different parts of the data based on their importance, without relying on the order they appear. The transformer will be trained using masked modeling, a method where parts of the input data are hidden from the model during training [10]. The model then learns to predict these hidden parts based on the remaining visible data. The performance of the model will be evaluated based on its ability to generate valid counterfactual molecules, and its capability to handle more complex cases.

1.2 Purpose of the study

The problem this thesis aims to solve is how to generate counterfactual molecules with the help of AI. More precisely, the aim is to develop a model that allows chemists to select a substructure of a molecule, and generate a similar molecule where the selected parts are modified.

Specifically, the questions this thesis aims to answer are the following:

- Can a transformer model trained with masked modeling techniques generate counterfactual molecules?
- Can human feedback be integrated into the de novo drug design process to generate relevant molecular structures?
- How well does the transformer model manage complex substructure replacements when trained with masked modeling techniques?

1.3 Limitations

To ensure a realistic scope of the thesis project, certain limitations have been established.

Training times for machine learning models can be extensive, especially if optimal performance is the goal. Given that the model is developed primarily for research and demonstration purposes, achieving optimal performance is not the top priority. Additionally, the project has a limited time frame, making it necessary to limit the model size. This means that the model will undergo training for a shorter, yet justifiable duration, rather than extending the training process for an extended period.

As the use of machine learning models continues to grow, the demand for explainable AI is increasing to build trust in decisions taken by models [7]. Implementing explainability features for a tool like the one being researched in this project would be advantageous for the chemists to understand and trust the model. However, researching explainability in-depth would complicate the project, and as the focus lies elsewhere, this aspect will not be investigated.

1.4 Thesis Outline

The structure of this thesis is organized as follows:

Chapter 2 (Theory) presents essential terminology and concepts for this project.

Chapter 3 (Methods) describes the methodology of the project.

Chapter 4 (Results) presents the results of the project.

Chapter 5 (Conclusion) discusses the results of this project and presents the conclusions.

2

Theory

2.1 Machine Learning

Machine learning is a discipline within artificial intelligence focusing on the development of algorithms and models enabling computers to perform tasks without specific instructions. Instead of following a predetermined set of rules, these systems learn to make predictions based on data. Machine learning is used in various subfields, such as computer vision, natural language processing, and robotics [11]. There are multiple applications of machine learning in a diverse array of sectors, ranging from healthcare to autonomous vehicles and machine translations.

2.1.1 Supervised Learning

Supervised learning is an approach in machine learning where the model learns from labeled training data. The data consists of pairs of input variables and their corresponding desired outputs. The objective of supervised learning is to create a predictive model that can generate an accurate output or prediction for previously unseen input [11]. Through training on the labeled input-output pairs, the model improves its predictive accuracy, and its ability to generalize to new, unseen data.

2.1.2 Unsupervised Learning

Unsupervised is another approach in machine learning. Contrary to supervised learning, which utilizes labeled data pairs during training, unsupervised learning uses unlabeled training data. During training, the model analyzes and clusters the input data, identifying patterns in it. This allows the model to learn directly from the inherent structure of the data [11].

2.2 Natural Language Processing

Natural language processing (NLP) is a field that lies in the intersection between linguistics and artificial intelligence [12]. It uses a broad range of computational techniques to achieve human-like language processing. There are many applications of NLP, including generation, summarization, and translation [13].

2.2.1 Sequence Generation Models

Sequence generation models are a class of models in machine learning designed to generate sequences of outputs. Such a model aims to learn patterns from the input data to generate new and plausible sequences based on that information [14]. This type of model is especially common in the field of NLP, where they are used for multiple tasks, such as text generation or machine translation. There are multiple different architectures used to build sequence generation models, such as Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTMs), and Transformer models [9], [15].

Sequence-to-sequence (seq2seq) models are a specific type of sequence generation model specialized in converting sequences from one domain to another. These models are often used for machine translation, where the input sequence is a sequence of words in one language, and the output sequence is a sequence of words in another language.

Generally, seq2seq models consist of two main components, an encoder and a decoder. The encoder reads the input sequence and generates a fixed-length vector representation which is fed to the decoder. From this, the output sequence is generated [16]. The encoder-decoder setup introduces flexibility, as it allows seq2seq models to handle input and output sequences of variable lengths [17]. This is a key advantage that makes them particularly suitable for NLP tasks, as these often include data of different lengths.

Auto-regressive models are a subset of sequence generation models that generate the next segment of the output by conditioning on prior segments. This forms a feedback loop that uses the past to predict the immediate future. This method is helpful for generating sequences that require contextual coherence, such as sentences in a target language [16].

2.3 Transformers

The transformer is a modern neural network model commonly used in NLP [9]. A transformer is a seq2seq model with an architecture based on an encoder and a decoder, and it works in an auto-regressive manner. The encoder processes the input into an encoded representation with lower dimensionality, while still retaining the relevant information. The decoder then uses the encoded representation to create the output of the model. The transformer architecture is illustrated in Figure 2.1.

2.3.1 Attention

One of the key components in a transformer is the use of the attention mechanism. Attention allows the model to weigh the importance of different input elements differently. In NLP, this could mean that a model can decide which words in a sentence are most relevant when generating a new sequence. Self-attention, used in transformers, specifically enables the model to contextualize each word in the input by evaluating the relationship between words in the sentence.

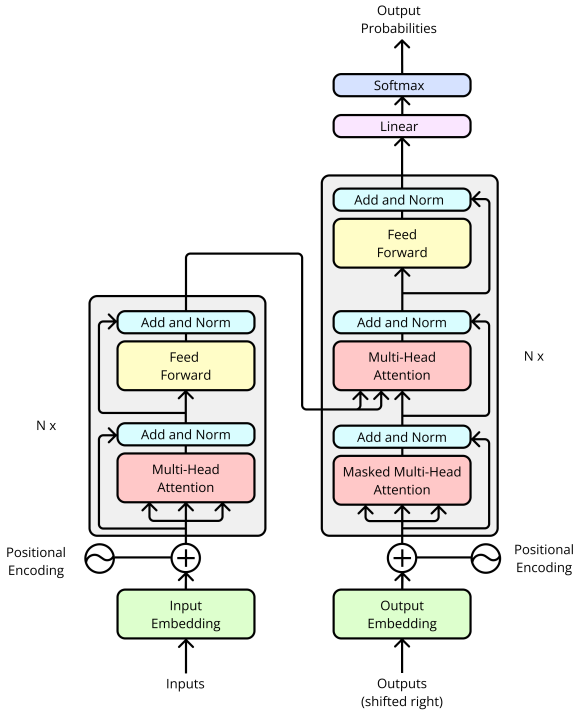


Figure 2.1: This figure shows the transformer model based on the multi-head attention architecture proposed in [9], from which the original figure is reproduced. The encoder is shown to the left in the figure, and the decoder is shown to the right.

2.3.2 Scaled Dot-Product Attention

The self-attention mechanism can be modeled as a function, where a set of key-value pairs and a query are mapped to an output. The result of the attention function is a weighted sum of the values, where the weight of each value is determined by its corresponding key's relevance to the query.

In the paper Attention Is All You Need [9], which introduced transformers, Vaswani et. al. present their way of calculating attention, called scaled dot-product attention. If the queries and keys have dimension d_k, and values have dimension d_v, the scaled dot-product attention is calculated by first taking the dot product of the query with all keys, then scaling the result by dividing with $\sqrt{d_k}$. The scaled dot-product attention is thereafter passed through a softmax function. The softmax function transforms these values into a probability distribution, producing a vector whose elements sum to one, with each value lying in the range of 0 to 1. This transformation allows the values to be interpreted as probabilities. After that, these probabilities are the weights used to calculate the weighted sum of the values, making up the final step. In practice, the queries, keys, and values are all matrices, and the scaled dot-product attention matrix can be computed as:

$$\text{Scaled Dot-Product Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.1}$$

2.3.3 Multi-Head Attention

One attention block may not be enough to enable the attention mechanism to capture all relevant relations in the input sequence. Therefore, the transformer uses multi-head attention, which can be seen as calculating attention in parallel on projected versions of queries, keys, and values. The final attention output is received by concatenating the attention scores from each head. The number of heads can be adjusted in different transformer implementations. Multi-head attention offers advantages over single attention heads, where unique attributes may be lost due to the merging of all information into a single, averaged output. Multi-head attention is calculated as follows:

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \end{aligned}$$

Where the projections are parameter matrices $W_i^Q \in \mathbb{R}\text{d}_{\text{model}} \times d_k, \text{W}_i^K \in \mathbb{R}\text{d}_{\text{model}} \times d_k, \text{W}_i^V \in \mathbb{R}\text{d}_{\text{model}} \times d_v$, and $W\text{O} \in \mathbb{R}\text{hd}_v \times d_{\text{model}}$.

2.3.4 Embeddings and Positional Encoding

In the transformer architecture, the input and output tokens are converted to vectors, and so-called learned embeddings are used to capture the semantic relationship between words. These embeddings are used to encapsulate the correlation between words through an unsupervised learning method. Another important feature of the architecture is the positional encoding. Positional encoding enables the model to take the order of the tokens into consideration. This is done by assigning a unique vector representation to each token's position within the sequence. Accounting for the sequential nature of the input data is important in natural language tasks, as it helps improve the models' ability to understand context and meaning.

2.3.5 Encoder and Decoder

As mentioned in Section 2.3, the transformer is based on an encoder-decoder architecture. The encoder is made up of N identical stacked layers. Each layer in the encoder has two sublayers: a multi-head attention mechanism followed by a fully connected feed-forward layer. After every sublayer, normalization is applied, which means that the activation of each layer is shifted to have a normal distribution. Normalization stabilizes and speeds up the training, and also improves the performance of the model [18]. There are also residual connections for each sublayer, a way for the data to skip the sublayer before being normalized. Residual connections have been shown to lead to faster convergence [19].

The decoder is similar to the encoder. It also consists of N identical layer, each having two sublayers similar to those found in the encoder. The decoder also has an extra sublayer, that considers multi-head attention over the output from the encoder layers. Furthermore, the self-attention sublayer in the decoder is modified with masking. The masking in this sublayer makes sure that the predictions for a position in a sequence only depend on the known previous outputs.

2.4 De Novo Molecular Design

De novo molecular design is the process of constructing novel molecules with desired properties with the help of computational tools. This process is often used in the field of drug design, where new molecules are being designed to fit and bind to a biological target. Historically, rule-based methods have dominated de novo design, but these methods are often time-consuming and limited by the scope of the rules [20]. Machine learning has recently seen a surge in de novo design, and multiple methods have been applied to aid chemists in finding new drugs.

Some of the methods that have been used for de novo models include recurrent neural networks (RNN), reinforcement learning, generative adversarial networks (GAN), and different encoder-decoder-based methods [4]. De novo design through machine learning provides significant advantages over older methods, but it should still be implemented with care. For example, choosing a good training set is critical for machine learning models. If the data used to train the model does not represent the drug-like chemical space, the properties of the generated molecules will not give a good representation of the desired chemical space [4].

Machine learning models for de novo design can be based on varying types of representations of molecules, as different tasks require different representations. The most common representations used are simplified molecule input line entry system (SMILES) [21], molecular graphs [22], fingerprints [23], and three-dimensional geometries [24]. SMILES has grown to become the most popular representation of molecules and has served as a universal language of chemical structures. A reason for the popularity of SMILES in deep learning models can be explained by the success and growth of NLP. Since SMILES is a text representation of molecules many of the technologies used for NLP can be used for de novo molecular design as well. Selecting the correct representation of molecules for the task is key to getting a well-performing model [4].

2.4.1 Evaluating De Novo Models

There are many ways to evaluate deep learning models for de novo design. One possible metric is validity, which checks the proportion of valid molecules generated. Another metric is uniqueness, which is a measure of the diversity of the generated molecules.

There also exists complete benchmark suits specifically designed for de novo molecular design models. An example of this is GuacaMol [25], an evaluation framework based on a suite of standardized benchmarks. Some factors the benchmarks capture include the model's ability to reproduce the property distribution of the training set, its ability to explore the chemical space, as well as multiple metrics on novelty.

In addition to these methods, the Tanimoto similarity score is another metric that can be used to assess de novo drug design models. Tanimoto similarity measures the similarity between two so-called molecular fingerprints. A molecular fingerprint is a digital representation of a molecule that encodes its structure [26].

2.5 Molecular Representation with SMILES

SMILES (Simplified Molecular Input Line Entry System) is a chemical notation system that uses ASCII strings to describe chemical structures. It is designed for efficiently processing chemical information with computers while maintaining readability for humans [27]. An important asset of the SMILES format is its compactness, requiring less storage space than many other methods of representing chemical structures [28]. The key symbols in the SMILES notation are described in Figure 2.2. SMILES strings can be used together with various computer programs or toolkits to generate two-dimensional drawings or three-dimensional models of molecules. One example of such a toolkit is RDKit [29].

Function	Symbol	Note
Single bond	-	usually omitted
Double bond	=	
Triple bond	#	
Positive charge	[C+]	square bracket and sign
Negative charge	[C-]	square bracket and sign
Aromatic atom	c	
Disconnected component	.	
Branch	(C)	enclosed in parantheses
Ring opening / closing	1	any number, starting from 1

Figure 2.2: This figure presents the key symbols used in SMILES to represent chemical structures. Each entry specifies the symbol, its corresponding function, and additional notes on its usage where applicable.

All elements in the periodic table are supported by the SMILES system, and atoms are represented by their atomic symbol. There is a set of rules that a SMILES string must follow [21]. An uppercase letter indicates a non-aromatic atom and a lowercase letter indicates an aromatic atom. By default, molecules are represented without hydrogen atoms.

Single bonds between atoms can be denoted by ”-”, though they are usually omitted. Double bonds are denoted by ”=”, and triple bonds are denoted by ”#”. A branch from a chain in a molecule is represented by placing parentheses around the atoms that make up the branch. Numbers are used to indicate ring openings and closings. One ring will have a certain number identifying its opening and closing. The next ring will have another number identifying its opening and closing, and so on. Charged atoms are shown by placing the atomic symbol followed by sign and numerical value within square brackets [21].

SMILES strings can often be written in multiple ways for the same molecule. However, there are algorithms designed to always generate the same SMILES string for a given molecule. These SMILES strings are said to be in canonical format. To illustrate this concept, multiple SMILES representations of the molecule Aspirin are shown in Table 2.1 along with the canonical SMILES string for the molecule. Canonical SMILES ensures uniqueness, which is important in many applications where SMILES strings are used – such as chemical databases.

SMILES representation	Canonical SMILES	Name
C(C)(Oc1c(C(O)=O)cccc1)=O	CC(=O)Oc1ccccc1C(=O)O	Aspirin
c1cccc(C(=O)O)c1OC(=O)C	CC(=O)Oc1ccccc1C(=O)O	Aspirin
c1(OC(C)=O)c(C(O)=O)cccc1	CC(=O)Oc1ccccc1C(=O)O	Aspirin
c1cccc(OC(=O)C)c1C(O)=O	CC(=O)Oc1ccccc1C(=O)O	Aspirin

Table 2.1: This table illustrates multiple SMILES representations of the molecule Aspirin, along with the canonical SMILES representation.

2.6 Cheminformatics Toolkit RDKit

RDKit is an open-source cheminformatics toolkit [29]. RDKit offers a range of tools for the manipulation and analysis of chemical information. With RDKit, it is possible to calculate a wide range of molecular descriptors and properties such as molecular weight. There are also features for computing molecular similarity. The core data structures and algorithms of RDKit are implemented in the programming language C++, but the toolkit provides wrappers that allow using it with languages such as Python, Java, and C#.

RDKit uses so-called *Mol* objects, which are representations of molecules that can be manipulated and analyzed within the framework [30]. The Mol object contains detailed information on atoms, bonds, and the molecule's structure. They also hold information about properties such as molecular weight and the number of atoms in the molecule.

The Mol objects can be created from SMILES string or other file types, and can then be edited, analyzed, and used for calculations. For example, RDKit enables the addition or removal of atoms and bonds, and conducting substructure searches to identify specific patterns. It is also possible to visualize the molecules in RDKit using the Mol objects. An example of such a visualization is shown in Figure 2.3, where the molecule Aspirin is depicted.

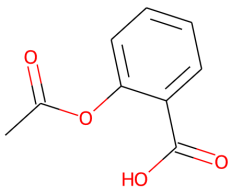


Figure 2.3: This figure shows a visualization of the molecule Aspirin, created with RDKit.

2.7 REINVENT - A Tool for Designing Molecules

REINVENT is an open-source generative AI framework for designing molecules [31]. The framework is built on recurrent neural networks and transformer architectures. The main idea of REINVENT is to train deep learning models that learn to produce valid molecules represented by SMILES strings. REINVENT 4, the latest version of REINVENT, supports multiple molecule generators. A generator is an algorithm that creates new molecules based on different constraints. LinkInvent, LibInvent, and Mol2Mol are three of the generators in REINVENT 4 [31].

Sampling mode is one of the different run modes in REINVENT. Sampling is utilized during inference time, after the training phase of the model, to generate molecules. During sampling in Mol2Mol, an input molecule is given to the model, and sampling techniques are used to generate a set of new molecules from this input molecule. The *n_smiles* parameter specifies the number of molecules generated for each input molecule. In Figure 2.4 the sampling process of a model with an input molecule is illustrated.

REINVENT supports two different sampling techniques: multinomial sampling and beam search. Multinomial sampling is a non-deterministic approach that generates sequences based on a probability distribution across possible elements, allowing for the fast creation of different combinations. This method has a parameter named *temperature*, that controls the level of randomness. A lower temperature will lead to more predictable outcomes, while a higher temperature increases diversity. Multinomial sampling can suffer from mode collapse, resulting in the generation of only a few unique sequences. Beam search on the other hand is a deterministic strategy. In contrast to multinomial sampling, which needs to save the probability of each possible token, beam search must also save the most probable subsequences. Beam search always generates unique sequences, but requires more computational resources [31].

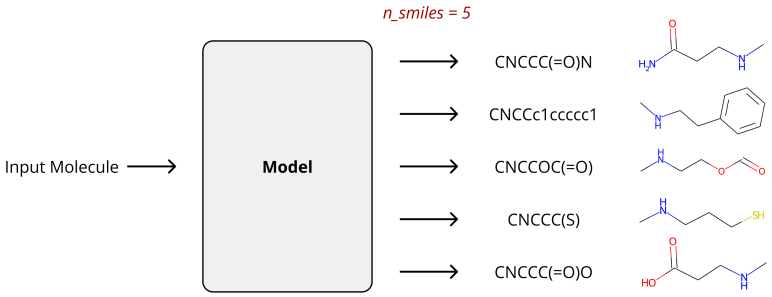


Figure 2.4: This figure provides an illustrative representation of the sampling process with an input molecule. For each input molecule being fed into the model, *n_smiles* new molecules are generated.

2.7.1 Mol2Mol

Mol2Mol is one of the generators in REINVENT. It is a transformer-based model used for molecular optimization, which means that the model can be used to find an optimized version of a given molecule. The model works based on molecular similarities. Given a restraint molecule and a similarity limit, Mol2Mol will generate similar molecules within the provided limit. The similarity between the input molecule and the generated molecule can be controlled, and the scaffold can be modified within the boundaries of the given similarity [31].

3

Methods

In this chapter, the methodology of the project is described. The methodology aims to find answers to the following research questions:

- Can a transformer model trained with masked modeling techniques generate counterfactual molecules?
- Can human feedback be integrated into the de novo drug design process to generate relevant molecular structures?
- How well does the transformer model manage complex substructure replacements when trained with masked modeling techniques?

The steps performed to answer these research questions are detailed in the following sections.

The machine learning model used in this project is based on the design of REINVENT's transformer-based Mol2Mol model. The Mol2Mol model is already integrated into the architecture of the REINVENT framework, which has several features for testing, evaluation, and reporting. Developing a model based on the same architecture as the Mol2Mol model made the implementation process efficient while allowing the utilization of the functionalities provided by the REINVENT framework.

All data used to train the models was retrieved from the publicly available ChEMBL database. The ChEMBL database is a large-scale chemical and bioactivity database managed by the European Bioinformatics Institute [32]. It is designed to store information about bioactive molecules with drug-like properties and consists of 2,4 million public compounds obtained from patents and publications.

3.1 Masked Modeling Setup

A masked modeling approach was used to explore the potential of using a transformer architecture to generate counterfactual molecules based on chemist feedback. The training set consisted of source-target pairs where the source is the masked SMILES and the target is the desired prediction.

The source is created by passing a SMILES string through a masking algorithm (described in Section 3.2). This algorithm creates masked versions of SMILES strings,

representing molecules where specific substructures are hidden. The way the source-target (input/output) pairs were set up is described in Section 3.1.1.

3.1.1 Source-Target Variations

Four distinct source-target variations were proposed, as illustrated in Table 3.1. These determine how the input (source) and the desired prediction (target) are structured for the learning process.

Variant	Source/Input	Target/Output
1	CCC(CO)NC**<SOM><MASK><EOM>**(CC)CO	CCC(CO)NC**<SOM>CNC<EOM>**(CC)CO
2	CCC(CO)NC**<SOM><MASK><EOM>**(CC)CO	**<SOM>CNC<EOM>**
3	CCC(CO)NC**<MASK>**(CC)CO	CCC(CO)NC**CNC**(CC)CO
4	CCC(CO)NC**<MASK>**(CC)CO	**CNC**

Table 3.1: This table illustrates the four different masking approaches suggested. For each variant, the input source structure and the corresponding target outputs are displayed. All variants have a special masking token in the source: <MASK>. Variants 1 and 2 also contain special <SOM> and <EOM> tokens in both the source and the target, marking the start and end of the masked sequences. Variants 3 and 4 differ in the target sequence: while variant 3 aims to generate the full SMILES string, variant 4 only generates the masked parts.

In variants 1 and 2, special tokens ('Start of Mask' (SOM) and 'End of Mask' (EOM)) indicate the start and end of the masked sections in the source molecule. The masked part is represented with a MASK token.

In variant 1, the target is the entire SMILES string, with the masked parts, delineated by SOM and EOM tokens. In variant 2, only the masked parts are used as targets, separated by the SOM and EOM tokens. This approach ensures that the substructure outside of the mask remains unchanged.

Variants 3 and 4 use no SOM and EOM tokens in the source or the target. In both variants, a MASK token is used to represent the masked section. In variant 3, the target is simply the original SMILES string. In variant 4, the target is the masked part. If there are multiple masked parts, they are separated by a separation token, '|', in the target.

3.2 Masking Algorithm

To mask the SMILES strings, the strings are converted into *Mol* representations in RDKit. In this format, each atom is assigned a specific index. The initial step in the masking process is to identify the atoms within the selected substructure using their indices and then setting their atomic number to 0.

In RDKit, an atomic number of 0 is represented by an asterisk, *, in the SMILES string, effectively clearing the atomic information. This method can be seen as the simplest form of masking. However, this approach alone is not sufficient to use for masking, as explained by the following example.

Consider the molecule shown in Figure 3.1, where the goal is to mask the substructure made up of the atoms with indices 0 and 1, connected with a double bond.

Masking only these atoms would output the following SMILES string: *=*NC(=O)CC(CC)c1ccccc1.

This approach would restrict the flexibility of what could be generated, as the newly generated molecule would also need to include a double bond. To make the generation more flexible, masking atom numbers 0 and 1 should also include the removal of the connecting double bond.

This issue is addressed in the algorithm by automatically masking all explicit bonds between any two masked atoms. This was implemented using string operations. Using the same example as above, it would now be: ***NC(=O)CC(CC)c1ccccc1. This adjustment allows the model to generate substructures more flexibly, not constrained by the presence of a certain bond between masked atoms

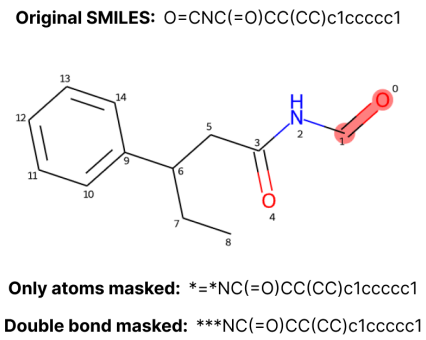


Figure 3.1: This figure shows a molecule where the substructure made up of the atoms with indices 0 and 1 should be masked, corresponding to the highlighted part in the figure. Below the molecule drawing, the SMILES string where only the atoms are masked is shown. The double bond has also been masked in the SMILES string furthest down.

Handling ring structures is another important part of masking. Consider cases where chemists want to remove a ring in a molecule, as illustrated in Figure 3.2. In these cases, it is essential to mask the ring information, represented by digits in the SMILES string. This again allows the model to be more flexible, making it possible to either generate a ring or another substructure. Without masking the ring information, the model would be constrained to generate a ring to produce a valid molecule. Presented in Figure 3.2 is one example of a SMILES string where ring information has been included in the mask, and one where the ring information is still there.

In certain cases, only the ring opening or ring closing should be masked. To illustrate why this is needed, another example is presented in Figure 3.3. In this example, three atoms are masked: one outside a ring and two inside it, as indicated by the '1' following one of the masked atoms. If masking is performed this way, the model is required to generate structures before and after the ring opening. However, if the ring opening is masked, the model only needs to generate strings in one place. Therefore, it can either generate only one atom and close the ring, or generate other atoms in front of it. Similar situations can happen with the closing ring information. So, if a digit follows a masked atom in the SMILES string, it will always be masked.

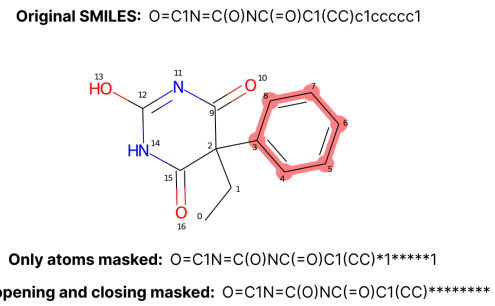


Figure 3.2: This figure shows a molecule where the substructure made up of the atoms with indices 3, 4, 5, 6, 7, and 8 should be masked, corresponding to the highlighted part in the figure. Below the molecule drawing, the SMILES string where only the atoms are masked is shown. The ring opening and closing have also been masked in the SMILES string furthest down.

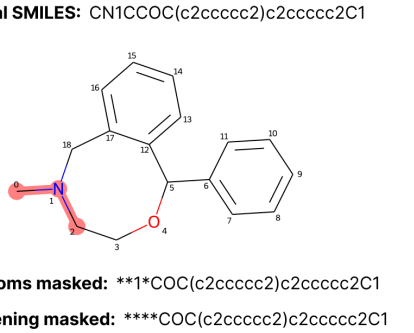


Figure 3.3: This figure shows a molecule where the substructure is made up of the atoms with indices 0, 1, and 2, corresponding to the highlighted part in the figure. Below the molecule drawing, the SMILES string where only the atoms are masked is shown. The ring opening has also been masked in the SMILES string furthest down.

The implementation of the masking algorithm consists of two main parts. Initially, the molecule is handled as a *Mol* in RDKit, and the atomic numbers for the selected atoms are set to zero. The second part of the implementation handles the molecule in the format of a SMILES string, using string operations to handle ring information, explicit bonds, and parentheses. Initially, the masked parts of the SMILES strings are represented by asterisks. Each sequence of asterisks is then replaced by one MASK token, as illustrated in Table 3.1.

3.2.1 Reconstructing Molecules

The same molecule can be represented by many different SMILES strings, as demonstrated in Table 2.1. The masked SMILES string and the original SMILES string must have the same representation; otherwise, they will not align.

The masking process begins by converting the SMILES string into a *Mol* representation using RDKit. In a later stage of the masking, the *Mol* object must be converted back to a SMILES string. RDKit outputs a canonical SMILES when converting from a *Mol* to SMILES, but this canonical form may be altered when atomic numbers

are set to zero during masking. Therefore, the SMILES string could change when converted back from the masked *Mol*.

To address this issue, a reconstruction algorithm was developed. The purpose of this algorithm is to take the masked source molecule, fill in the masked parts, and ensure that the source and target SMILES strings are formatted consistently. This ensures that the reconstructed molecule matches the format of the masked SMILES.

To illustrate the utility of the reconstruction algorithm, consider the case illustrated in Figure 3.4. In this example, the goal is to mask the marked substructure. The original SMILES string is shown at the top of the figure, and below are two masked versions of the same SMILES string. As seen, the masked SMILES string where the reconstruction algorithm has not been applied has another representation than the original SMILES string. However, the SMILES string furthest down has been passed through the reconstruction algorithm and now has the same representation as the original SMILES string.

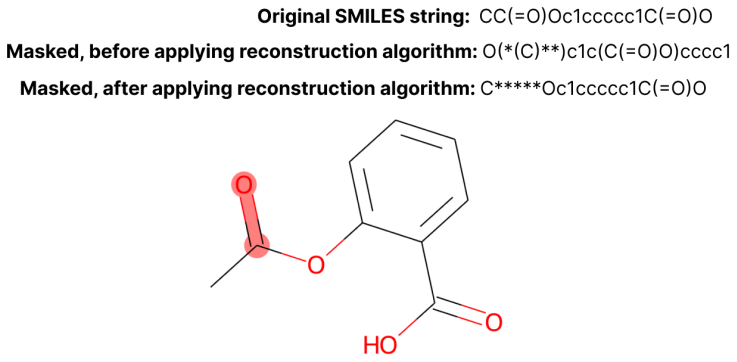


Figure 3.4: This figure shows an example of two versions of the same masked SMILES string. The same substructure is masked in both strings, however, their representation differs. After applying the reconstruction algorithm, the masked SMILES string has the same representation as the original SMILES string.

3.3 Masking Strategies

The selection of which atoms will be masked in a molecule is decided by the masking strategy. It is important that the masking is diverse, as selecting the masks in a restricted way could result in a limited model with less applicability and learning potential. Multiple approaches for selecting which substructures to mask in the data were developed, described in the following sections. The three masking strategies are illustrated in Figure 3.5.

3.3.1 Masking Rings and Functional Groups

This masking strategy masked a combination of two types of substructures: rings and functional groups. Each molecule in the dataset was masked six times, creating six different source-target pairs.

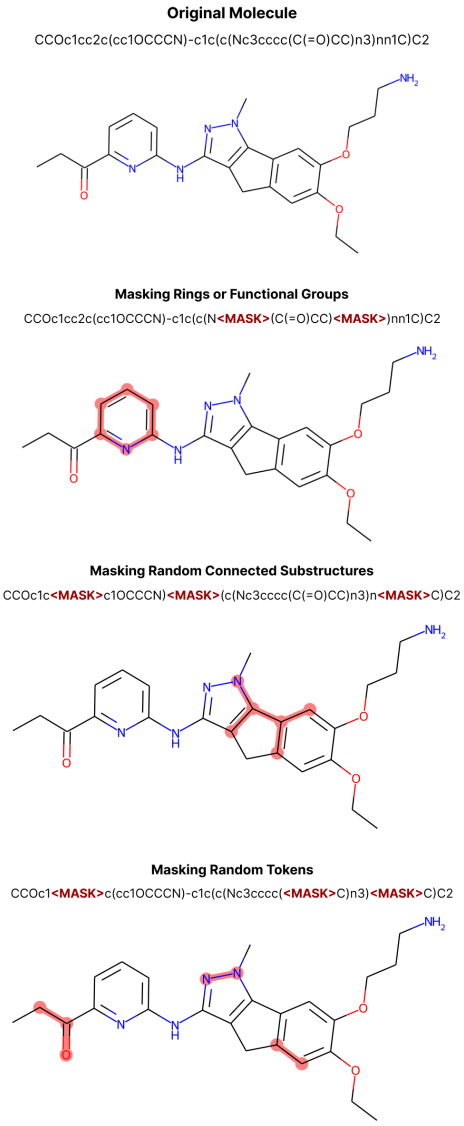


Figure 3.5: This figure shows an example of the different masking strategies presented in 3.3. At the top of the figure, the original molecule and its SMILES are shown. Below, an example from each masking strategy is shown, with the highlighted atoms indicating a mask.

Functional groups are parts of molecules that have distinctive chemical properties. Due to their distinctive properties, functional groups are often targeted for replacement by chemists. The functional groups were identified through an iterative algorithm [33]. An example of a functional group that can be selected as a mask is the hydroxyl group, as illustrated in Figure 3.6.

Since many functional groups consist of only one or a few atoms, they might limit the variability of masks. To achieve a broader range of masks, the masking strategy introduced an element of randomness. This was implemented by including a probabilistic mechanism that, with a specified probability, extended the masks of small functional groups by adding their neighboring atoms. With this approach, the size of the masked functional groups was diversified.

Random rings were also masked, as illustrated in Figure 3.7. Masking only complete rings might restrict the model to generating exclusively ring structures and no additional substructures when a chemist selects a ring. Therefore, in addition to masking entire rings, when a ring intersects with a functional group, there is a probability that both the ring and the functional group are combined into a single mask. This expanded mask selection is shown in Figure 3.8.

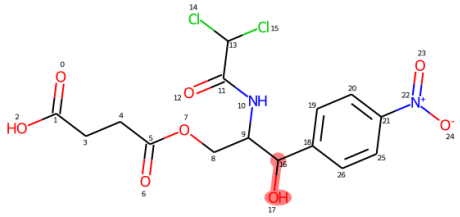


Figure 3.6: This figure shows an example of a molecule where a functional group has been selected to be masked. The functional group selected has been highlighted in red and consists of the atoms with index 16 and 17.

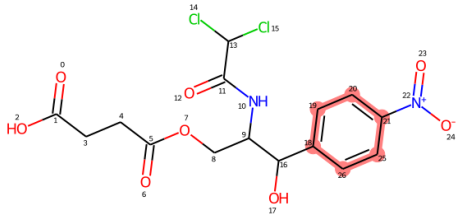


Figure 3.7: This figure shows an example of a molecule where a ring has been selected to be masked. The ring selected has been highlighted in red and consists of the atoms with indices 18, 19, 20, 21, 25, and 26.

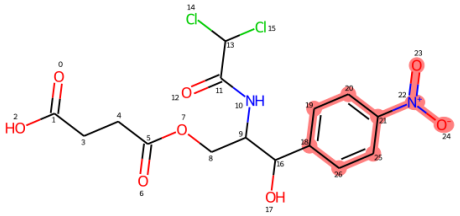


Figure 3.8: This figure shows an example of a molecule where a ring and an intersecting functional group have been selected to be masked. The mask is highlighted in red and consists of the atoms with indices 18, 19, 20, 21, 22, 23, 24, 25 and 26.

3.3.2 Masking Random Tokens

The selection of which atoms to mask in this approach was entirely based on the SMILES representation of the molecule and not on the molecular graph. Each SMILES string was masked by inserting one to three MASK symbols in a random position of the string. Each MASK symbol replaced one to four tokens. It is important to note that the masked tokens were not necessarily adjacent or connected in the molecular graph.

3.3.3 Masking Random Connected Substructures

In this masking strategy, randomly connected substructures were selected to be masked. Each mask had a randomly selected length, ranging between one and eight connected atoms. The process started by choosing a random atom in the molecule as the starting point. From this atom, the mask was expanded by including the nearest neighboring atoms using a breadth-first search (BFS) approach. In this approach, all direct neighbors of the starting atom are included before moving on to their neighbors, ensuring that the mask grows level by level across the molecule. This expansion continued, adding layers of adjacent atoms until the mask reached the desired length. Using this method ensured that the masked substructures were connected in the molecular graph.

3.4 Initial Models

The initial machine learning models developed to generate counterfactual molecules are described in this section. The development of the first exploratory transformer model in this project is described in Section 3.4.1.

Four separate machine learning models were developed to investigate how input data variations impact the model's performance in generating accurate and unique molecular structures. The details about these models are described in Section 3.4.2.

3.4.1 Exploratory Model

An exploratory transformer model was developed to assess the potential of using a transformer-based architecture in the given context. This initial model was trained on a small dataset to limit the training times. After splitting the dataset into a training set, a validation set, and a test set, and masking the molecules, the training set consisted of approximately 5700 data points.

Each molecule in the dataset was masked in six different ways following the masking strategy allocating four masks for functional groups and two for rings, as described in Section 3.3. When a molecule lacked four functional groups or two rings, all available entities of each category were masked.

The source-target variation employed in the data used for training and evaluating this exploratory model was variant 1 shown in Table 3.1. This variant uses SOM and EOM tokens both in the source and target sequences.

The model underwent training over 50 epochs with this training set. The model was saved every third epoch, and the model with the lowest validation loss was used for sampling. To evaluate the exploratory model, two rounds of experiments were conducted, with each round consisting of three separate sampling experiments. In each round, three sampling runs used multinomial sampling, and three sampling runs used beam search. For each sampling run, the model generated a specified number of new molecules (`n_smiles`), specifically 15, 50, and 100, from each molecule in

the test set. The test dataset comprised approximately 700 unique molecules, each masked and not previously encountered by the model.

3.4.2 Exploring Source-Target Variations

As mentioned in Section 3.1.1, the SMILES strings can be masked using multiple syntax variations. The different variants are described in Section 3.1.1 and illustrated in Table 3.1. Four unique models were trained, one for each source-target variation. By training individual models on these variations, the aim was to evaluate which specific source-target pairing yielded the most accurate and reliable model.

The dataset employed for training the variant models was consistent with that used for the initial exploratory model, using the same masking strategy. However, it was uniquely adapted for each model to align with the respective source-target variation under investigation. Each model was trained for 50 epochs. The models were saved every third epoch, and the model with the lowest validation loss was used for sampling.

Three sampling runs were performed for each model to evaluate the four variant models. The datasets used for sampling were consistent with the ones used to sample the exploratory model. All sampling runs used beam search. The first run generated 15 new molecules per input molecule (`n_smiles` = 15), the second generated 50 molecules per input molecule (`n_smiles` = 50), and the third generated 100 molecules per input molecule (`n_smiles` = 100).

3.5 Masking Configurations

Following the evaluation of source-target variations and the selection of one approach, further experiments were conducted to explore different masking configurations. During the experiments, one parameter was changed while the others were fixed to see how they would affect the models' behavior. An overview of the parameters can be seen in Table 3.2.

For these models, larger datasets were used. Their sizes can be seen in Tables 3.3 – 3.8. Each model was trained for 50 epochs. The model was saved every third epoch, and the model with the lowest validation loss was used for sampling.

Parameter Name	Explanation
Masking Strategy	The method used to select which parts of the molecule are masked (described in Section 3.3.).
Atoms Masked	The number of atoms covered by the mask.
Masking Frequency	How many times each molecule is masked.
n_random	The number of times each SMILES string is randomized (0 means canonical SMILES is used).
Dataset size	The number of unique source-target pairs used to train the model.

Table 3.2: This table summarizes parameters tested in the experiments.

For each model, sampling was performed using beam search. The test set contained 5000 molecules. These molecules had functional groups and rings masked, using the masking strategy described in Section 3.3.1. For each molecule in the test set, 100 molecules were generated (`n_smiles` = 100).

3.5.1 Masking Strategy

The three masking strategies are described in Section 3.3 and illustrated in Figure 3.5. To evaluate the impact of each masking strategy, three models were trained. All three models were trained using data sets with the same molecules, but the masking was performed according to each masking strategy.

The first model was trained on data masked by employing the masking strategy masking rings and functional groups, as described in Section 3.3.1. Each molecule was masked six times, four being functional groups and two being rings.

The second model was trained on data masked by selecting random tokens in the SMILES string for each molecule, as described in Section 3.3.2. The third model was trained on data masked by selecting randomly connected substructures in the molecule. The strategy is described in Section 3.3.3.

An overview of the different models can be seen in Table 3.3

Model Name	Masking Strategy	Atoms Masked	Masking Frequency	n_random	Dataset Size
rings_functional_groups	Rings and Functional Groups	1-8	6	0	550 000
random_tokens	Random Tokens	1-8	6	0	550 000
random_substructures	Random Substructures	1-8	6	0	550 000

Table 3.3: This table describes the models trained to explore the different masking strategies. Model `rings_functional_groups` was trained on a dataset prepared with masking rings and functional groups, `random_tokens` with masking random tokens, and `random_substructures` with masking randomly connected substructures.

3.5.2 Varying Mask Length

These models were trained using data with random substructures masked, utilizing the masking strategy described in Section 3.3.3. Four datasets were prepared, varying the number of atoms included in the random substructures being masked. The molecules in the four datasets had 1–3, 1–8, 1–15, and 1–25 atoms masked respectively. The details are shown in Table 3.4.

Model Name	Masking Strategy	Atoms Masked	Masking Frequency	n_random	Dataset Size
1-3_mask_length	Random substructures	1-3	6	0	550 000
1-8_mask_length	Random substructures	1-15	6	0	550 000
1-15_mask_length	Random substructures	1-15	6	0	550 000
1-25_mask_length	Random substructures	1-25	6	0	550 000

Table 3.4: This table describes the models trained to explore the impact of the number of atoms masked. Model `1-3_mask_length` was trained on a dataset with 1–3 atoms masked, model `1-8_mask_length` with 1–8, model `1-15_mask_length` with 1–15, and model `1-25_mask_length` with 1–25.

3.5.3 Masking Frequency with Fixed Dataset Size

To examine the effects of varying masking frequencies on model performance, three datasets were constructed, each differing in the number of times each molecule was masked. As described in Table 3.5, three models were trained, one for each dataset. The first model was trained on a dataset where each molecule was masked once; the

second model on a dataset where each molecule was masked ten times; and the third model on a dataset where each molecule was masked twenty times.

Model Name	Masking Strategy	Atoms Masked	Masking Frequency	n_random	Dataset Size
1_mask_times	Random substructures	1-8	1	0	550 000
10_mask_times	Random substructures	1-8	10	0	550 000
20_mask_times	Random substructures	1-8	20	0	550 000

Table 3.5: This table describes the models trained to assess the impact of varying the number of times each molecule is masked. All models were trained on datasets of approximately the same size, but each dataset differed by the frequency of molecule masking: `1_mask_times` had one masking per molecule, `10_mask_times` had ten maskings per molecule, and `20_mask_times` had twenty maskings per molecule.

3.5.4 Masking Frequency with Varied Dataset Sizes

An additional experiment was conducted to analyze the effect of the masking frequency. In the previous experiment, described in Section 3.5.3, all models were trained on datasets of the same size. Here, three models were trained on the same unique set of molecules, and since the masking frequency was changed, the models with higher masking frequency had larger datasets.

Model Name	Masking Strategy	Atoms Masked	Masking Frequency	n_random	Dataset Size
1_mask_times	Random substructures	1-8	1	0	50 000
10_mask_times	Random substructures	1-8	10	0	500 000
20_mask_times	Random substructures	1-8	20	0	1 000 000

Table 3.6: This table describes the models trained to assess the impact of varying the number of times each molecule is masked. Each dataset differed by the frequency of molecule masking: `1_mask_times` had one masking per molecule, `10_mask_times` had ten maskings per molecule, and `20_mask_times` had twenty maskings per molecule. As the masking frequency increases, so does the size of the datasets.

3.5.5 Randomization of SMILES with Fixed Dataset Size

Given that a single molecule can be represented by multiple SMILES strings, as demonstrated in Table 2.1, experiments with randomized SMILES strings were conducted to investigate the potential limitations of using only the canonical SMILES in representing the chemical space.

Four distinct datasets were created to explore this, each using a different number of randomized SMILES representations. Four corresponding models were trained using these datasets. Table 3.7 describes the parameters used in the experiments. The first dataset used canonical SMILES strings only. The second dataset used one randomized variant for each SMILES string. The third and fourth datasets included three and six randomized variants per SMILES string. Each dataset consisted of approximately 550,000 molecules. The data was masked by selecting random connected substructures in the molecule, as described in Section 3.3.3. Between one and eight connected atoms were randomly selected for masking in each molecule.

Model Name	Masking Strategy	Atoms Masked	Masking Frequency	n_random	Dataset Size
0_random	Random substructures	1-8	3	0	550 000
1_random	Random substructures	1-8	3	1	550 000
3_random	Random substructures	1-8	3	3	550 000
6_random	Random substructures	1-8	3	6	550 000

Table 3.7: This table describes the models trained to explore the impact of using randomized SMILES strings. Four models were trained: `0_random` used a dataset with canonical SMILES strings only, `1_random` with each SMILES string randomized once, `3_random` with each SMILES string randomized three times, and `6_random` with each SMILES string randomized six times.

3.5.6 Randomization of SMILES with Varied Dataset Sizes

Further experiments were conducted to investigate the impact of using randomized SMILES strings. A similar experimental setup to that described in Section 3.5.5 was employed to determine whether increasing the training set sizes with randomized SMILES would affect the results. Four models were trained on the same unique set of 167,000 molecules, although the number of randomized SMILES strings variation in each model's training set was increased. The models that used multiple randomized SMILES strings therefore had a larger dataset.

Model Name	Masking strategy	Atoms Masked	Masking Frequency	n_random	Dataset Size
0_random	Random substructures	1-8	3	0	167 000
1_random	Random substructures	1-8	3	1	167 000
3_random	Random substructures	1-8	3	3	500 000
6_random	Random substructures	1-8	3	6	1 000 000

Table 3.8: This table describes the models trained to explore the impact of using randomized SMILES strings. Each dataset differed by the number of times each SMILES string was randomized. `0_random` used a dataset with canonical SMILES strings only, `1_random` with each SMILES string randomized once, `3_random` with each SMILES string randomized three times, and `6_random` with each SMILES string randomized six times. As the number of randomizations increases to 3 and 6, so does the size of the datasets.

3.6 Case Studies

To further understand and illustrate the behavior of the different models, a series of case studies was conducted to test the models' applicability in realistic scenarios. The case studies focused on a selected set of molecules with different substructures masked.

Five relevant drug-like molecules were selected, chosen to represent a diverse range of structures. Each molecule was masked in multiple ways, targeting substructures such as rings, functional groups, and other features. A total of 21 cases were studied. The masks in these molecules were selected to replicate typical situations where a chemist would likely select to modify or replace parts of a molecule.

The sampling process involved generating 100 samples per input molecule. Each experiment was evaluated based on several metrics, and every set of generated molecules was also studied. This process helped ensure that the models generated not only valid molecules but also relevant and interesting ones.

4

Results

In this chapter, the results of the experiments and analyses from the project are presented. The following subsections provide the results and a discussion of their various aspects.

4.1 Initial Models

In this section, the outcomes of the initial modeling stages are presented. First, the results of the exploratory model, trained using a smaller dataset of approximately 5700 masked SMILES strings are presented. This model was trained using source-target 1, described in Table 3.1. Following this, the results from evaluating the source-target pair variation models are presented.

The models were sampled using beam search, and three different metrics are presented for these experiments. The 'Valid' metric highlights the model's ability to generate valid molecules by showing what percentage of the generated SMILES are valid. The second metric, 'Unique', computes the percentage of unique SMILES generated over all of the molecules in the test set. The third metric, 'Valid and Unique', shows how many of the molecules are both valid and unique. This metric is used to interpret the model's capability of generating valid unique molecules.

The specifics of the results, including observations, are discussed in the following subsections.

4.1.1 Exploratory Model

The results from the sampling runs conducted to evaluate the exploratory model are presented in Table 4.1 and Table 4.2. In Table 4.1, the results from the three sampling runs using multinomial sampling are shown, and in Table 4.2, the results from the three sampling runs using beam search are presented.

Table 4.1 shows that the percentage of valid SMILES generated is ranging from 47.8% for `n_smiles` = 15 to 41.1% for `n_smiles` = 100 when using multinomial sampling. The percentage of unique SMILES is highest at `n_smiles` = 15, at 54.5%. As `n_smiles` increases to 50, the uniqueness decreases significantly to 39.6%, and for `n_smiles` of 100, it slightly improves to 42.9%.

n_smiles	Valid (%)	Unique (%)	Valid and Unique (%)
15	47.8	54.5	17.4
50	47.9	39.6	10.5
100	41.8	42.9	10.5

Table 4.1: This table presents the results of the three **multinomial** sampling runs on a test set of approximately 700 molecules using the exploratory model. It presents three distinct sample sizes (15, 50, and 100) and evaluates them in terms of the percentage of valid SMILES, the percentage of unique SMILES, and the percentage of valid unique molecules generated by the exploratory model.

n_smiles	Valid (%)	Unique (%)	Valid and Unique (%)
15	36.6	99.8	36.5
50	31.2	99.8	31.1
100	28.4	99.8	28.3

Table 4.2: This table presents the results of the three **beam search** sampling runs on a test set of approximately 700 molecules using the exploratory model. It presents three distinct sample sizes (15, 50, and 100) and evaluates them in terms of the percentage of valid SMILES, the percentage of unique SMILES, and the percentage of valid unique molecules generated by the exploratory model.

The percentage of SMILES that are both valid and unique is highest at `n_smiles`=15, with 17.4%. The results indicate that, when using multinomial sampling, the exploratory model is more effective at producing valid and unique SMILES strings when generating fewer SMILES per input molecule.

Table 4.2 shows that the percentage of valid molecules is 36.6% when sampling 15 molecules with beam search, but that the validity decreases when more molecules are sampled. The percentage of unique molecules generated is 99.8% for all sampling runs using beam search, which is expected since beam search generally generates unique molecules.

Comparing beam search and multinomial sampling, there are distinct trade-offs. Beam search consistently generates a higher percentage of unique molecules across all `n_smiles`. However, multinomial sampling shows higher validity percentages, especially at lower `n_smiles`.

Generating 28.3 % valid and unique molecules when sampling 100 molecules with beam search is not an outstanding performance. However, given the limited size of the dataset and the task's complexity, this outcome indicates that the model is effectively learning, thereby showing that the method is viable.

4.1.2 Source-Target Variations

Three sampling runs were performed for each model, differing in how many molecules were sampled for each input molecule. The percentage of valid SMILES and unique molecules were compared for the models trained on the different source-target variations. The results are presented in Tables 4.3 – 4.5.

The results from the sampling experiments suggest that the performance metrics are relatively similar across all variants. Only variant 2 seemed to perform slightly worse than the other variants in regards to the validity of the generated molecules.

Variant	Valid (%)	Unique (%)	Valid and Unique (%)
1	36.6	99.8	36.5
2	38.6	94.7	35.0
3	28.5	89.4	23.5
4	42.1	96.5	39.3

Table 4.3: This table presents the results from sampling 15 molecules per input (`n_smiles` = 15) on a test set of approximately 5000 molecules, comparing four source-target variants.

Variant	Valid (%)	Unique (%)	Valid and Unique (%)
1	31.2	99.8	31.1
2	31.0	94.7	28.9
3	25.5	89.5	21.2
4	35.2	96.3	33.3

Table 4.4: This table presents the results from sampling 50 molecules per input (`n_smiles` = 50) on a test set of approximately 5000 molecules, comparing four source-target variants.

Variant	Valid (%)	Unique (%)	Valid and Unique (%)
1	28.4	99.8	28.3
2	27.9	93.9	26.1
3	23.9	89.7	20.1
4	31.5	95.2	29.6

Table 4.5: This table presents the results from sampling 100 molecules per input (`n_smiles` = 100) on a test set of approximately 5000 molecules, comparing four source-target variants.

Variant	Training time (minutes)
1	86
2	22
3	83
4	21

Table 4.6: This table presents the training times for the four models trained on different source-target variants.

However, a distinct difference could be observed in the training duration for the four different models as seen in Table 4.6. Training times were significantly shorter for variants 2 and 4, where the task was to generate the masked sequences and not reconstruct the entire SMILES string, For this reason, a decision was made to move forward with variant 4, which showed better performance than variant 2.

4.2 Masking Configurations

In this section, the results from the experiments with different masking configurations are presented. Sampling runs were performed on the different models to explore how varying mask parameters affected the performance.

To evaluate the results of these experiments additional metrics were added to the already explained metrics. *Average Difference in Heavy Atoms* compares the number of heavy atoms in the generated molecule with the original unmasked molecule. A positive difference in heavy atoms means that the model has expanded the input molecule and that the generated molecule has more heavy atoms than the input molecule. A negative difference in heavy atoms means that the model has shrunk the input molecule by generating a smaller molecule. This difference is then averaged over all generated molecules to compute the average difference.

The *Average Tanimoto Similarity* compares the generated molecule with the input molecule by calculating the Tanimoto similarity between the two molecules. This measures how similar the generated molecule is to the original molecule. Again, the results are averaged over all generated molecules.

For some of the experiments, two additional metrics were calculated to look more specifically at what was being generated. The metric "Ring → Ring" measures how often a ring was generated when a ring was in the selected substructure. The metric "No Ring → Ring" measures how often a ring was generated when a ring was not part of the masked substructure.

4.2.1 Masking Approach

In Table 4.7, the results from the sampling runs on the three different models described in Section 3.5.1 are shown.

Model	Valid and Unique (%)	Avg. Diff. in Heavy Atoms	Avg. Tanimoto Sim.
rings_functional_groups	70.1	1.6	0.65
random_tokens	51.4	-0.2	0.64
random_substructures	82.5	1.5	0.66

Table 4.7: This table presents the results from sampling 100 molecules per input (`n_smiles` = 100) on a test set of approximately 5000 molecules, comparing three models trained using various masking approaches, as described in Section 3.5.1.

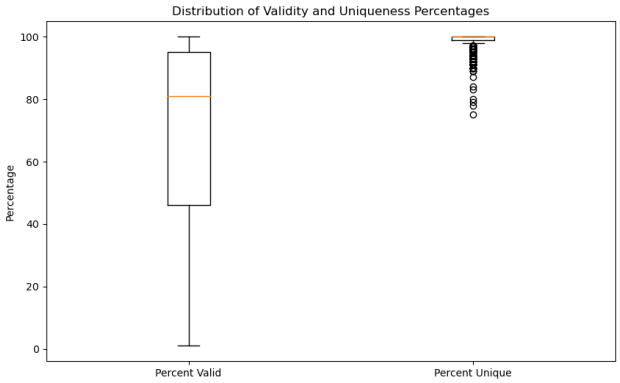


Figure 4.1: This boxplot shows the distribution of the percentage of valid and percentage of unique molecules generated by the `rings_functional_groups` model.

The model `rings_functional_groups`, trained on a dataset masking rings and functional groups, generates 70.1% valid and unique molecules. The distribution of the valid and the unique molecules can be seen in Figure 4.1. The difference in heavy atoms is 1.6, with an Average Tanimoto similarity of 0.65.

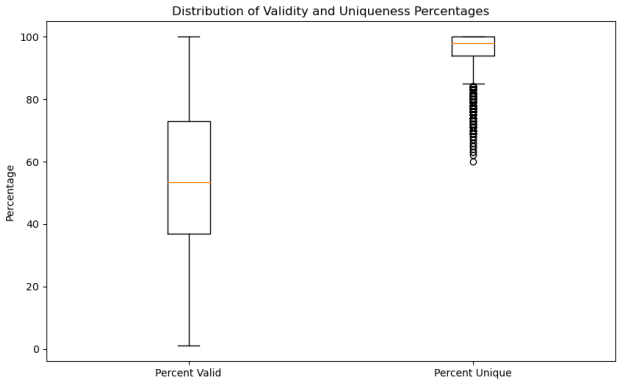


Figure 4.2: This boxplot shows the distribution of the percentage of valid and percentage of unique molecules generated by the `random_tokens` model.

The model `random_tokens` shows a lower performance, with only 51.4% of the generated molecules being both valid and unique. In Figure 4.2 the distribution of the valid and the unique molecules generated can be seen in a boxplot. The boxplot shows that the low valid and unique percentage of this model is mainly caused by the model generating fewer valid molecules. Interestingly, this model shows a negative average difference in heavy atoms (-0.2), indicating that it reduces the size of the generated molecules compared to the original molecule. The Tanimoto similarity on the other hand is similar to that of the first model, at 0.64.

The third model, `random_substructures`, has the highest percentage of valid and unique molecules (82.5%). The boxplot in Figure 4.3 shows the spread of the valid and unique molecules generated. The boxplot confirms that the `random_substructures` model not only has the highest percentage of valid and unique molecules but also that the entire distribution of valid molecules is shifted upwards compared to the other models.

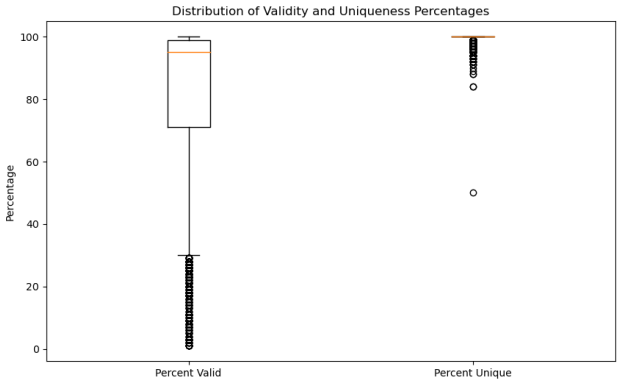


Figure 4.3: This boxplot shows the distribution of the percentage of valid and percentage of unique molecules generated by the `random_substructures` model.

The average difference in the number of heavy atoms is 1.6, which is comparable to the first model, and the Tanimoto similarity of this model, 0.66, is similar to both of the other models. In summary, the `random_substructures` model has the highest performance, whereas the `random_tokens` model shows significantly lower results.

4.2.2 Mask Length

In Table 4.8, the results for the experiments with models trained on varying mask lengths are presented. The experiment setup is described in Section 3.5.2.

Model	Valid and Unique	Avg. Diff. in Heavy Atoms	Avg. Tanimoto Sim.	Ring → Ring	No Ring → Ring
1-3_mask_length	44.8%	-0.89	0.66	7%	0.8%
1-8_mask_length	84.3%	1.62	0.66	71%	12%
1-15_mask_length	88.1%	3.00	0.66	83%	23%
1-25_mask_length	87.3%	3.38	0.65	84%	26%

Table 4.8: This table presents the results from sampling 100 molecules per input (`n_smiles` = 100) on a test set of approximately 5000 molecules, comparing three models trained using various lengths of masks, as described in Section 3.5.2.

It is clear from the table that the model using the shortest mask length, `1-3_mask_length`, generated significantly lower percentages of valid and unique molecules, compared to the other models. Beam search generally generates a high rate of unique molecules, so the low number suggests that masking shorter sequences may not provide enough variability or complexity to generate valid molecules.

The models trained on longer masked sequences performed better in this aspect. Notably, the model `1-15_mask_length` performed best, generating 88.1% valid and unique molecules. Longer masks seem to encourage the generation of valid and unique molecules, likely due to introducing more varied and complex features in the training phase.

The results also indicate a correlation between the average difference in heavy atoms and mask length. The model trained on the longest masks expands the molecules the most, with on average 3.38 heavy atoms, while the model trained on the shortest mask tends to decrease the size of the molecules. The average Tanimoto similarity

between the source and the generated molecules is nearly identical for all of the four models, at around 0.65.

The metrics "Ring → Ring" and "No Ring → Ring", show how the model performs for ring transformations. As mask length increases, the models tend to maintain or introduce ring structures more often. For example, the `1-25_mask_length` model shows the highest percentage of transformation from non-ring to ring structures (26%) and also preserves rings with the highest frequency (84%).

On the other hand, model `1-3_mask_length` introduces new rings less often, with only 0.8% of molecules without rings gaining new ring structures. This isn't surprising, as masking only 1-3 atoms will not include the masking of any ring structures. Models `1-8_mask_length` and `1-15_mask_length` demonstrate a considerable increase in the percentage of rings generated.

4.2.3 Mask Frequency per Molecule with Fixed Dataset Size

The results of the experiments with models trained on varying mask frequencies per molecule are presented in Table 4.9. The experiment setup is described in Section 3.5.3.

Model	Valid and Unique	Avg. Diff. in Heavy Atoms	Avg. Tanimoto Sim.	Ring → Ring	No Ring → Ring	Dataset size
1_mask_times	82.0%	1.53	0.66	67.6%	11.6%	550 000
10_mask_times	82.1%	1.58	0.67	67.9%	10.9%	550 000
20_mask_times	82.3%	1.48	0.67	67.7%	11.2%	550 000

Table 4.9: This table presents the results from sampling 100 molecules per input (`n_smiles` = 100) on a test set of approximately 5000 molecules, comparing three models trained using various mask frequencies per molecule with a fixed dataset size across all models, as described in Section 3.5.3.

These results indicate a strong similarity across all three models in terms of their ability to produce valid and unique molecules, as well as in their average Tanimoto similarity scores. Specifically, the models 1_mask_times, 10_mask_times, and 20_mask_times all generated a nearly identical percentage of valid and unique molecules, at around 82%. This similarity suggests that the frequency of masking per molecule does not significantly influence the overall validity and uniqueness of the generated molecules.

The average Tanimoto similarity scores are also very similar across the models, being between 0.66 and 0.67. This implies that increasing the number of times each molecule is masked does not largely affect the similarity to the reference molecules. The average difference in heavy atom weight is also similar among the models.

The percentage of generated molecules that maintain ring structures (Ring → Ring) is consistent across the models, at around 67-68%. The metric measuring if rings were generated when the masked substructure wasn't a ring (No Ring → Ring), is also similar for all models.

In summary, these results imply that the number of times a molecule is masked during training does not drastically influence the model's performance in these metrics.

4.2.4 Mask Frequency per Molecule with Varied Dataset Sizes

In Table 4.10, the results of the experiments with models trained on different mask frequencies per molecule with varied dataset sizes are presented. The experiment setup is described in Section 3.5.4.

The results show that the 20_mask_times was able to generate the most valid and unique molecules with 88.7%. The 10_mask_times model performs very similar to the 20_mask_times, while the 1_mask_times model performs noticeably worse with 69.0% valid and unique molecules generated. This indicates that expanding the dataset by masking the same molecule multiple times could be a useful strategy, however, when the dataset size is large, the mask frequency will not make a significant difference.

The average Tanimoto similarity and the average difference in heavy atoms are similar across the three models, the only distinguishable difference being that the 1_mask_times had a slightly smaller average difference in heavy atoms than the other two models. Interestingly, it can be seen in the ring metrics that the models that had higher mask frequency were more likely to generate rings. A potential explanation for this could be that the masking strategy has a bias towards masking rings, and expanding the dataset could also lead to increasing the bias.

4.2.5 Randomized SMILES with Fixed Dataset Size

Table 4.11 shows the results from sampling runs on the models trained with different numbers of randomized SMILES strings per molecule. The experiment setup is described in Section 3.5.5.

The data shows that all models trained with randomized SMILES strings have a high performance in generating valid and unique molecules, with percentages ranging from 89.7% to 91.8%. The model trained with one randomization has the highest percentage of valid and unique molecules, at 91.8%. The model with no randomization (only canonical SMILES) has a slightly lower performance in this metric, at 89.7%.

Regarding the average difference in heavy atoms, there is a noticeable increase for the model trained with 6 randomizations. This could suggest that more randomization leads to greater structural diversity in the generated molecules.

Model	Valid and Unique	Avg. Diff. in Heavy Atoms	Avg. Tanimoto Sim.	Ring → Ring	No Ring → Ring	Dataset size
1_mask_times	69.0%	2.49	0.64	61.5%	14.0%	50 000
10_mask_times	88.0%	3.09	0.66	82.7%	21.3%	500 000
20_mask_times	88.7%	3.08	0.66	84.6%	24.5%	1 000 000

Table 4.10: This table presents the results from sampling 100 molecules per input (`n_smiles` = 100) on a test set of approximately 5000 molecules, comparing three models trained using various mask frequencies per molecule but with a varied dataset size across the models, as described in Section 3.5.3.

Model	Valid and Unique	Avg. Diff. in Heavy Atoms	Avg. Tanimoto Sim.	Ring → Ring	No Ring → Ring	Dataset size
0_randomizations	89.7%	2.25	0.67	83.4%	22.8%	550 000
1_randomization	91.8%	2.39	0.67	79.3%	14.2%	550 000
3_randomizations	90.7%	2.29	0.67	78.2%	14.7%	550 000
6_randomizations	89.8%	3.22	0.67	78.5%	12.7%	550 000

Table 4.11: This table presents the results from experiments sampling 100 molecules per input (`n_smiles` = 100) on a test set of approximately 5000 molecules, comparing four models trained using different numbers of randomized SMILES strings per molecule, as described in Section 3.5.5.

On the other hand, the average Tanimoto similarity Tanimoto similarity for all models is consistent at 0.67. This suggests that while the structural details of the molecules vary, the overall similarity between the generated molecule and the input molecule remains relatively stable. The models maintain a certain level of resemblance to the reference molecules, regardless of the randomization degree.

The percentage of generated molecules that maintain ring structures (Ring → Ring) is relatively consistent for all models. However, in those cases where rings weren't masked but still generated (No Ring → Ring), a larger variation between the models is observed. Specifically, the model using only canonical SMILES (0_randomizations) has a higher percentage of rings generated in cases where other substructures were masked (22.8%).

4.2.6 Randomized SMILES with Varied Dataset Sizes

Table 4.12 shows the results from sampling runs on the models trained with different numbers of randomized SMILES strings using varied dataset sizes.

Model	Valid and Unique	Avg. Diff. in Heavy Atoms	Avg. Tanimoto Sim.	Ring → Ring	No Ring → Ring	Dataset size
0_randomizations	81.4%	2.68	0.66	77.4%	18.6%	167 000
1_randomization	85.9%	2.08	0.67	72.1%	11.0%	167 000
3_randomizations	90.5%	2.33	0.67	78.7%	14.2%	500 000
6_randomizations	91.7%	2.44	0.67	80.9%	15.6%	1 000 000

Table 4.12: This table presents the results from experiments sampling 100 molecules per input (`n_smiles` = 100) on a test set of approximately 5000 molecules, comparing four models trained using different numbers of randomized SMILES strings per molecule with varied dataset sizes, as described in Section 3.5.6.

In the results, it can be seen that the percentage of valid and unique molecules increases when the number of random SMILES strings and dataset size increases. The model trained on 1,000,000 datapoints using 6 random SMILES strings for each molecule has the highest valid and unique molecules with 91.7%, while the model trained on only canonical SMILES has the lowest percentage. The difference in valid and unique molecules generated is significantly less between the 6_randomization model and 3_randomization model than for example the 0_randomization and 1_randomization model. This indicates that randomization might be useful until a certain point, but using many random SMILES strings will not make a significant difference.

The results also suggest that randomization does not make a big difference in the average difference in heavy atoms or the average Tanimoto similarity. The percentage

of rings generated increases slightly when the dataset increases for the models using randomized SMILES strings. Notably, the model trained on canonical SMILES seems to generate more rings than the other models, especially compared to the 1_randomization model which is trained on a dataset of the same size.

4.3 Case Studies

In this section, three examples from the case study are presented. The model used to generate these molecules was `1-15_mask_length` (described in Section 3.5.2), as it showed good performance.

4.3.1 Case A

In Figure 4.4, one of the molecules in the case study is presented, with the masked atoms marked in red. The corresponding SMILES string without the mask is shown at the top of the figure, and the masked SMILES string is displayed at the bottom. In this case study the selected substructure to replace is a functional group consisting of 2 nitrogen atoms and a carbon atom.

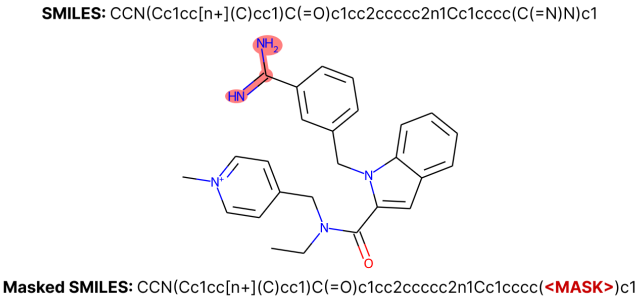
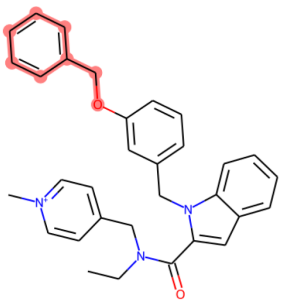


Figure 4.4: This figure shows a molecule that was a part of the conducted case study. The original SMILES string is shown above the image of the molecule. Below the image is the masked SMILES string, corresponding to the marked part in the molecular structure.

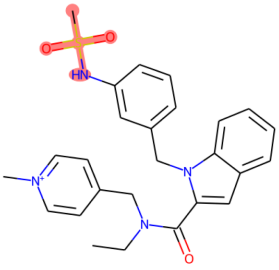
In Figures 4.5–4.7, three molecules generated from the masked SMILES string are presented. In the generated molecule in Figure 4.5, the model has decorated the molecule by slightly extending the chain and adding a ring at the end of it. In the second example seen in Figure 4.6 the masked atoms were replaced by a functional group centered around a sulfur atom.

The third example shown in Figure 4.7 highlights an example of a slightly larger replacement generated by the model. This chain-like replacement is nine atoms long. These three examples of the generated counterfactuals showcase the model's ability to generate diverse extensions to a molecule while keeping the original substructure intact.



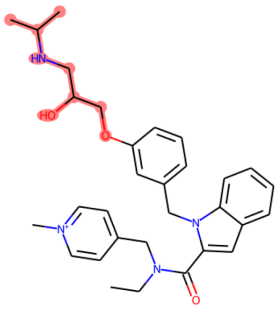
SMILES: CCN(Cc1cc[n+](C)cc1)C(=O)c1cc2ccccc2n1Cc1cccc(**OCc2ccccc2**)c1

Figure 4.5: This figure shows a molecule generated from the masked SMILES in Figure 4.4.



SMILES: CCN(Cc1cc[n+](C)cc1)C(=O)c1cc2ccccc2n1Cc1cccc(**NS(C)(=O)=O**)c1

Figure 4.6: This figure shows a molecule generated from the masked SMILES in Figure 4.4.



SMILES: CCN(Cc1cc[n+](C)cc1)C(=O)c1cc2ccccc2n1Cc1cccc(**OCC(O)CNC(C)C**)c1

Figure 4.7: This figure shows a molecule generated from the masked SMILES in Figure 4.4.

4.3.2 Case B

Figure 4.8 shows another molecule from the case study. The masked substructure is marked in red. In this case, a linker consisting of 3 atoms has been masked. The figure includes the original SMILES string above and the masked SMILES string below the molecule.

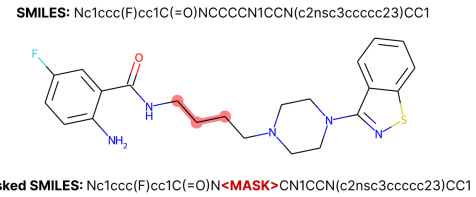


Figure 4.8: This figure shows a molecule that was a part of the conducted case study. The original SMILES string is shown above the image of the molecule. Below the image is the masked SMILES string, corresponding to the marked part in the molecular structure.

Three molecules generated from this masked input are presented in Figures 4.9–4.11. The first example shown in Figure 4.9 highlights the model's ability to grow the linker. In the original molecule, the selected linker was 3 atoms long but in the generated molecule, the model generated a longer linker consisting of 8 atoms. In the second example, illustrated in Figure 4.10, the model demonstrates its diverse capabilities by linking the two fragments through the generation of a ring. The final molecule which was generated for this case can be seen in Figure 4.11. The molecule in Figure 4.11 is similar to the first example where the model generated a linking chain, but in this case, the model also decorated the linker.

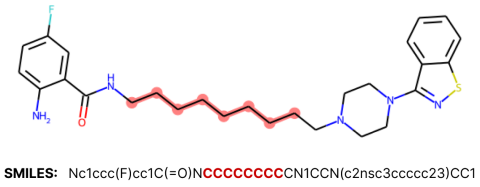


Figure 4.9: This figure shows a molecule generated from the masked SMILES in Figure 4.8.

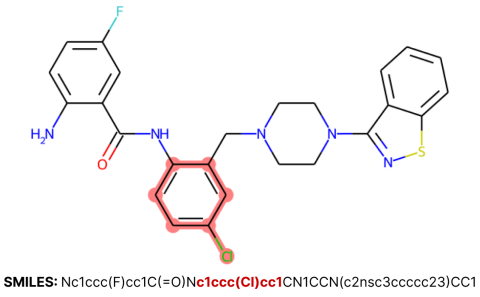
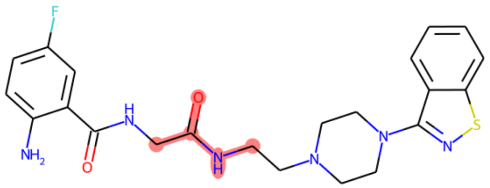


Figure 4.10: This figure shows a molecule generated from the masked SMILES in Figure 4.8.



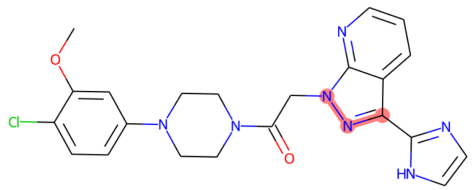
SMILES: Nc1ccc(F)cc1C(=O)N**CC(=O)NC**CN1CCN(c2nsc3ccccc23)CC1

Figure 4.11: This figure shows a molecule generated from the masked SMILES in Figure 4.8.

4.3.3 Case C

A third molecule from the case study is displayed in Figure 4.12, with the red marking showing the masked substructure. The original SMILES string is provided at the top of the figure, while the masked version is given at the bottom. Three atoms are masked, and as seen in the figure, the masked substructure results in one mask token in the SMILES string. This means that the model will generate in one place. The masking of this substructure breaks a ring, which can be seen as a more complex case than the two previous examples.

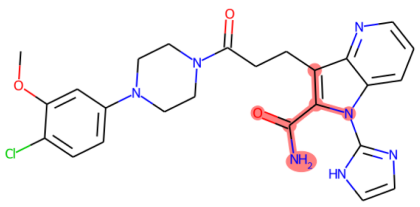
SMILES: COc1cc(N2CCN(C(=O)Cn3nc(-c4ncc[nH]4)c4cccnc34)CC2)ccc1Cl



Masked SMILES: COc1cc(N2CCN(C(=O)C**<MASK>**(-c4ncc[nH]4)c4cccnc34)CC2)ccc1Cl

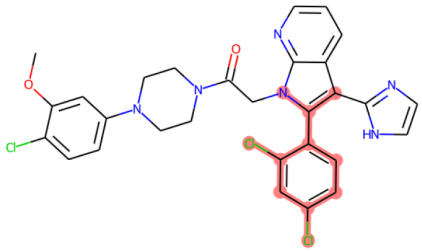
Figure 4.12: This figure shows a molecule that was a part of the conducted case study. The original SMILES string is shown above the image of the molecule. Below the image is the masked SMILES string, corresponding to the marked part in the molecular structure.

Figures 4.13–4.15 show three molecules generated from this masked SMILES string. As shown in the figures, all three generated molecules include ring structures. This is due to the syntax of the SMILES string. As the ring opening is masked but not the ring closing, a ring must be generated to produce a valid molecule. However, the model also generates structures branching from the ring as shown in Figures 4.13 and 4.14. The sizes of the generated rings also vary among the molecules.



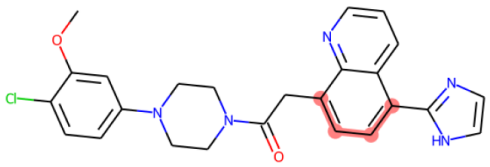
SMILES: COc1cc(N2CCN(C(=O)C**Cc3c(C(N)=O)n**(-c4ncc[nH]4)c4cccnc34)CC2)ccc1Cl

Figure 4.13: This figure shows a molecule generated from the masked SMILES in Figure 4.12.



SMILES: COc1cc(N2CCN(C(=O)C**n3c(-c4ccc(Cl)cc4Cl)c**(-c4ncc[nH]4)c4cccnc34)CC2)ccc1Cl

Figure 4.14: This figure shows a molecule generated from the masked SMILES in Figure 4.12.



SMILES: COc1cc(N2CCN(C(=O)C**c3ccc**(-c4ncc[nH]4)c4cccnc34)CC2)ccc1Cl

Figure 4.15: This figure shows a molecule generated from the masked SMILES in Figure 4.12.

5

Conclusion

This chapter concludes the thesis by discussing and summarizing the key findings from this project. The following sections discuss various strategies and parameters used in the experiments. The chapter also brings up the limitations of this study and potential directions for future work. It ends with a conclusive summary that links the results back to the research questions.

5.1 Discussion

This section discusses some of the methodological approaches and experimental setups used in the project. It also provides an analysis of the results, focusing on specific aspects of the experiments such as generation bias, mask length, masking strategies, mask frequency, and the use of randomized SMILES strings.

5.1.1 Masking Algorithm

The masking algorithm was specifically developed to meet the requirements of this project but holds potential for use in further research as well. While no experiments were conducted to systematically investigate the masking algorithm, it is important to note that its numerous design choices and details can significantly influence the model's performance. The algorithm could be modified to either be strict or allow for a more flexible generation, depending on the desired model behavior when rings, parts of rings, or branches are masked.

5.1.2 Tanimoto Similarity as Metric

The results showed that the Average Tanimoto Similarity between the generated molecules and the original molecules was very similar for all models. This similarity can be explained by the generation being substructure-constrained, meaning that large portions of the molecule are designed to remain unchanged. Tanimoto similarity evaluates the complete molecule similarity. This means that large molecules with small masks will always have a larger similarity. Given this aspect, the metric might not fully capture the diversity or novelty of the generated structures.

Another interesting metric considered was the Tanimoto similarity among the generated molecules themselves. However, because this similarity measure requires

pairwise comparisons, the computations were too time-consuming to be feasible. Even though it was not possible in this case, the metric could have given additional insights.

5.1.3 Generation Bias

One of the largest problems the developed model is facing is the bias in generating the replacements. If the original masked part is a non-ring structure, the model is much more likely to generate a non-ring replacement. The same applies in reverse, meaning that if a ring is selected only a few structures will be generated that do not contain a ring at that position. This suggests that the models are learning specific properties from the SMILES string that make them inclined to generate structures that are similar to the masked structures. The behavior indicates that the models are interpreting information encoded in the non-masked parts of the SMILES strings which reveal if the molecule contains a ring or not.

The observed bias has implications for the use case of the model. If, for example, the model `1-25_mask_length` is used, and a ring is selected to be replaced, there is a higher likelihood that the replacement structure will also be a ring. This bias could limit the model's exploratory capacity, as it is more prone to replicate familiar structures.

However, this bias also indicates that the models are learning to capture the molecular structure better as the mask length increases. This capability likely makes the model more able to generate valid molecules. As the mask length increases, the model appears to improve in generating valid and unique molecules.

Understanding the exact reason for the bias is challenging, as there are likely multiple factors contributing to the bias in generating rings A possible cause of the bias could be the ordering of ring information in the SMILES strings. The rings in SMILES strings are denoted by digits that indicate ring openings and closings. These digits are indexed starting from 1 in increasing order, and the model will likely learn this ordering. For example, consider a molecule with two rings indexed as 1 and 2. If the first ring is masked, the model will likely generate a new ring, since the ring with index 2 should not exist without a ring with index 1. To determine whether this sequential ordering contributes to the ring bias, it would be interesting to experiment by either completely randomizing the indices in the ring information or modifying the indices to ensure there are no gaps between them.

Managing this bias is important for specializing the models to suit specific goals and use cases. Depending on whether the goal is to increase reliability in generating known structures, or to focus on generating novel molecules, one might use different masking strategies. Finding a way to make the models unbiased in what they generate can be of significance for the future of these models.

5.1.4 Mask Length

The results show that the mask lengths have an impact on how the model performs. The `1-3_mask_length` model performs poorly, most likely because the masks are too short, and because of that are never able to capture rings or longer chains. This means that the model does not learn the concept of rings, and thereby performs poorly on the test set. The models containing longer masks all perform significantly better.

The results from the average difference in heavy atoms show that the mask length is important for how the model behaves and what it generates. Masking longer sequences leads to the model generating larger molecules. However, this seems to only hold up to a certain point. The model trained on masks of size 1-15 and the model trained on masks of size 1-25 has a similar difference in heavy atoms. A potential cause for this could be that the dataset contains molecules that consist of less than 25 atoms. Therefore, the distribution of the mask lengths in the `1-25_mask_length` model is not entirely uniform but rather shifted slightly towards the lower mask lengths.

An interesting aspect that has not been analyzed is the selection method for the randomly connected substructures that are masked in these experiments. The masks were chosen using a breadth-first search (BFS) approach, as explained in Section 3.3.3. An alternative method could be to select the masks using a depth-first search (DFS) approach instead. This would allow the atoms to be explored along one path in the molecule as far as possible before backtracking to explore a new path.

It could be interesting to investigate the impact that the way the masks are selected has on the generated molecules. Switching from a BFS to a DFS method for selecting masks in molecular experiments could significantly alter the structure of the masked substructures. It seems probable that the BFS method would mask rings more often than the DFS method, as BFS examines the molecules level by level, while DFS prioritizes depth.

The use of breadth-first search (BFS) to mask random components might explain why the models currently so often tend to generate rings. Since the masks affect what kind of structures the model generates, it is important to have a wide variety of structures masked to get a fair model. It would be interesting to explore whether using DFS leads to different outcomes and to consider combining DFS with BFS to potentially achieve a broader variety of masks.

5.1.5 Masking Strategy

Functional groups and rings are likely to be selected for replacement by chemists. This was the reason for trying to mask functional groups and rings specifically since that is what the model will need to learn to replace.

However, after trying randomized masking it was observed that randomized masking performed better than masking functional groups and rings in regards to validity and uniqueness. Masking random connected substructures leads to the model seeing

more diverse chemical structures and patterns than if only functional groups and rings were masked. It gives the model a better understanding of the molecules and the SMILES syntax which could explain the improved performance.

Masking functional groups specifically will on the other hand lead to the model seeing more functional groups, which could result in more unique and rare replacements. It could lead to the model having a better understanding of the different functional groups. While random masking will give the model a better understanding of the molecules, many of the masks will not be functional groups, which could lead to the model never seeing rare functional groups. Further experiments and analysis are needed to draw a conclusion on what strategy is best for different use cases, but a good compromise might be to use a mix of both approaches.

5.1.6 Mask Frequency

In the mask frequency experiments where the models were trained on a fixed dataset size, there was no significant difference in the performance of the models. This indicates that the mask frequency has no impact on the model performance and that if there is enough training data there is no gain in masking each molecule multiple times. In fact, it is probably better for the model to be trained on a larger set of unique molecules rather than having a higher masking frequency since the model will see more diverse cases and therefore learn a larger chemical space.

In the results presented in Table 4.10 where the models were trained with a varied dataset size the difference in performance between the models was larger. There was a clear difference between the 1_mask_times model and the better-performing 10_mask_times model. This shows that using a higher masking frequency is a good option when there is limited data to achieve better performance of the model. But as seen in Section 4.2.4 when the dataset becomes large the mask frequency will not make a noticeable difference. In this project the model was trained on drug-like molecules which have a lot of publicly available data, however, if a model would be trained on something more restricted, using a higher masking frequency is a useful solution.

5.1.7 Randomized SMILES

The use of randomized SMILES strings was initially tested to see if the model's generation bias would improve since canonical SMILES could be limiting and might be a cause of the bias. However, the models trained on varying numbers of randomized SMILES had the same sort of bias as the model trained on canonical SMILES. In these experiments, the different randomized version of the same SMILES string was simply included in the training set from the beginning. Another interesting experiment would be to provide the model with a different randomized version of the same SMILES string every epoch to see if this would affect the generation bias.

Models with randomized SMILES strings were also trained without restricting the dataset size. From the results in 4.2.6 it is clear that randomization is a good strategy to use to expand the dataset. Several models trained on randomized SMILES

have a valid & unique percentage of over 90%, which is better than what has been achieved during any other masking configurations. For example, the 6_randomizations model achieved 91.7% unique & valid molecules generated. The best model from the other masking configuration experiments achieved 88.1%, though many performed noticeably worse. This suggests that randomization benefits the models' performance.

5.2 Future Work

The results were evaluated using several metrics that helped understand the diversity of the molecules created and how well the models could generate valid molecules. However, the metrics might not cover every important aspect of potential drug-like molecules. Other important factors could be how the body absorbs and processes these molecules, or how easy they are to make. Future studies could look at other metrics that consider these aspects, as they would make these evaluations more useful for drug development.

It would also be possible to explore other metrics. For example, to evaluate the novelty of the generated molecules, it would be possible to look at function groups generated that were not in the training set.

The thesis also presents multiple ways to augment data, leading to possibilities to extend the training data. However, due to limitations in the time available for training, this study could not utilize datasets of larger size. Future research should consider extending the training duration and using more expansive datasets to enhance model performance and generalizability. Also, there is potential to explore various parameters in greater detail and extend the experiments to combine different strategies of masking.

Investigating the explainability of the models used is also important, as it could provide deeper insights into the decision-making processes of these models. These insights could be used both to understand the models' performance better and to help understand how they make predictions.

Lastly, it could be interesting to try different approaches to generate counterfactual molecules. Working with SMILES strings has proven to be complex. The mapping from a graph structure (a *mol* object) to a one-dimensional text representation is an intricate and difficult process comprising many steps. Handling the text representation requires advanced string manipulation and consideration of many complicated cases. One suggestion is therefore to try graph neural networks, as the molecules can be represented as graphs.

5.3 Conclusion

In conclusion, this thesis has shown that masked modeling for generating counterfactual molecular structures through the use of a transformer-based architecture has merit. However, in its current state, its biased generation hinders its utility. According to the results, specific masking strategies influence the generated molecules' properties. This demonstrates the importance of mask selection in this application.

This work shows the potential of applying masked learning to molecular generation and optimization. Generating substructure-constrained molecules by incorporating human feedback in the molecular design process shows promise in producing more relevant molecules.

The thesis also discusses some of the difficulties with the method and presents strategies and parameters to explore and optimize further. Expanding on the ideas presented in this report could encourage more efficient drug discovery processes, contributing to faster developments of pharmaceutical treatments. The insights gained from this study will hopefully inspire continued innovation and exploration in the field, driving forward the capabilities of AI in molecular design.

Bibliography

[1] C. A. Nicolaou and N. Brown, "Multi-objective optimization methods in drug design," *Drug Discovery Today: Technologies*, vol. 10, no. 3, e427–e435, 2013. DOI: 10.1016/j.ddtec.2013.02.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1740674913000085?via%3Dihub.

[2] I. Sundin, A. Voronov, H. Xiao, *et al.*, "Human-in-the-loop assisted de novo molecular design," *Journal of Cheminformatics*, vol. 14, no. 1, pp. 1–16, 2022. DOI: 10.1186/s13321-022-00667-8. [Online]. Available: https://jcheminf.biomedcentral.com/articles/10.1186/s13321-017-0235-x?.

[3] R. Bohacek, C. McMartin, and W. Guida, "The art and practice of structure-based drug design: A molecular modeling perspective.," *Med. Res. Rev.*, vol. 16, no. 1, pp. 3–50, 2004. DOI: 10.1002/(SICI)1098-1128(199601)16:1<3::AID-MED1>3.0.CO;2-6. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/8788213/.

[4] M. Wang *et al.*, "Deep learning approaches for de novo drug design: An overview.," *Current opinion in structural biology*, vol. 72, pp. 135–144, 2022. DOI: 10.1016/j.sbi.2021.10.001. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0959440X21001433.

[5] P. Schneider, W. P. Walters, A. T. Plowright, *et al.*, "Rethinking drug design in the artificial intelligence era," *Nature Reviews Drug Discovery*, vol. 19, no. 5, pp. 353–364, 2020. DOI: 10.1038/s41573-019-0050-3. [Online]. Available: https://www.nature.com/articles/s41573-019-0050-3.

[6] N. Brown, P. Ertl, R. Lewis, T. Luksch, D. Reker, and N. Schneider, "Artificial intelligence in chemistry and drug design," *Journal of Computer-Aided Molecular Design*, vol. 34, pp. 709–715, 2020. DOI: 10.1007/s10822-020-00317-x. [Online]. Available: https://link.springer.com/article/10.1007/s10822-020-00317-x#Sec3.

[7] S. Baron, "Explainable ai and causal understanding: Counterfactual approaches considered," *Minds and Machines*, vol. 33, pp. 1–31, 2023. DOI: 10.1007/s11023-023-09637-x. [Online]. Available: https://link.springer.com/article/10.1007/s11023-023-09637-x.

[8] A. Lamens and J. Bajorath, "Generation of molecular counterfactuals for explainable machine learning based on core-substituent recombination.," *Chemmedchem*, vol. 19, e202300586–e202300586, 2023. DOI: 10.1002/cmdc.202300586. [Online]. Available: https://chemistry-europe.onlinelibrary.wiley.com/doi/10.1002/cmdc.202300586.

[9] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. DOI: 10.48550/arXiv.1706.03762. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[10] J. He, K. Tian, S. Luo, *et al.*, "Masked molecule modeling: A new paradigm of molecular representation learning for chemistry understanding," 2022. DOI: 10.21203/rs.3.rs-1746019/v1. [Online]. Available: https://www.researchsquare.com/article/rs-1746019/v1.

[11] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, pp. 255–260, 2015. DOI: 10.1126/science.aaa8415. [Online]. Available: https://www.cs.cmu.edu/~tom/pubs/Science-ML-2015.pdf.

[12] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, "Natural language processing: An introduction," *Journal of the American Medical Informatics Association*, vol. 18, no. 5, pp. 544–551, 2011. DOI: 10.1136%2Famiajnl-2011-000464. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3168328/.

[13] E. D. Liddy, "Natural language processing," in *Encyclopedia of Library and Information*, Marcel Decker, Inc., 2001.

[14] D. Otter, J. Medina, and J. Kalita, "A survey of the usages of deep learning for natural language processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, pp. 1–21, 2020. DOI: 10.1109/TNNLS.2020.2979670. [Online]. Available: https://www.researchgate.net/publication/340820178_A_Survey_of_the_Usages_of_Deep_Learning_for_Natural_Language_Processing.

[15] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–1780, Dec. 1997. DOI: 10.1162/neco.1997.9.8.1735. [Online]. Available: https://www.researchgate.net/publication/13853244_Long_Short-term_Memory.

[16] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, Dec. 2014, pp. 3104–3112. DOI: 10.5555/2969033.2969173. [Online]. Available: https://dl.acm.org/doi/10.5555/2969033.2969173.

[17] K. Cho, B. van Merriënboer, C. Gulcehre, *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds., Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179. [Online]. Available: https://aclanthology.org/D14-1179.

[18] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer normalization*, 2016. DOI: 10.48550/arXiv.1607.06450. arXiv: 1607.06450 [stat.ML]. [Online]. Available: https://arxiv.org/pdf/1607.06450.pdf.

[19] T. Liu, M. Chen, M. Zhou, S. S. Du, E. Zhou, and T. Zhao, *Towards understanding the importance of shortcut connections in residual networks*, 2019. DOI: 10.48550/arXiv.1909.04653. arXiv: 1909.04653 [cs.LG]. [Online]. Available: https://arxiv.org/pdf/1909.04653.pdf.

[20] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen, "Molecular de-novo design through deep reinforcement learning," *Journal of cheminformatics*, vol. 9, no. 1, pp. 1–14, 2017. DOI: 0.1186/s13321-017-0235-x. [Online]. Available: https://jcheminf.biomedcentral.com/articles/10.1186/s13321-017-0235-x?.

[21] Daylight Checmical Information Systems, Inc., *Smiles tutorial*, 2022. [Online]. Available: https://www.daylight.com/dayhtml_tutorials/languages/smiles/index.html.

[22] Y. Li, J. Zhang, R. Zhao, *et al.*, "Highly efficient actively q-switched yb:lggg laser generating 3.26mj of pulse energy," *Optical Materials*, vol. 79, pp. 33–37, 2018. DOI: https://doi.org/10.1016/j.optmat.2018.03.022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925346718301496.

[23] P.-C. Kotsias, J. Arús-Pous, H. Chen, O. Engkvist, C. Tyrchan, and E. J. Bjerrum, "Direct steering of de novo molecular generation with descriptor conditional recurrent neural networks," *Nature Machine Intelligence*, vol. 2, no. 5, pp. 254–265, May 2020. DOI: 10.1038/s42256-020-0174-5. [Online]. Available: https://doi.org/10.1038/s42256-020-0174-5.

[24] M. Skalic, D. Sabbadin, B. Sattarov, S. Sciabola, and G. De Fabritiis, "From target to drug: Generative modeling for the multimodal structure-based ligand design," *Molecular Pharmaceutics*, vol. 16, no. 10, pp. 4282–4291, 2019. DOI: 10.1021/acs.molpharmaceut.9b00634. [Online]. Available: https://doi.org/10.1021/acs.molpharmaceut.9b00634.

[25] N. Brown, M. Fiscato, M. H. Segler, and A. C. Vaucher, "Guacamol: Benchmarking models for de novo molecular design," *Journal of Chemical Information and Modeling*, vol. 59, no. 3, pp. 1096–1108, Mar. 2019. DOI: 10.1021/acs.jcim.8b00839. [Online]. Available: https://doi.org/10.1021/acs.jcim.8b00839.

[26] D. Bajusz, A. Rácz, and K. Héberger, "Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations?" *Journal of Cheminformatics*, vol. 7, no. 20, 2015. DOI: 10.1186/s13321-015-0069-3. [Online]. Available: https://jcheminf.biomedcentral.com/articles/10.1186/s13321-015-0069-3.

[27] D. Weininger, "Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules," *Journal of chemical information and computer sciences*, vol. 28, no. 1, pp. 31–36, 1988. DOI: 10.1021/ci00057a005. [Online]. Available: https://pubs.acs.org/doi/10.1021/ci00057a005.

[28] Daylight Checmical Information Systems, Inc., *Smiles - a simplified chemical language*, 2022. [Online]. Available: https://www.daylight.com/dayhtml/doc/theory/theory.smiles.html.

[29] RDKit, *Rdkit: Open-source cheminformatics software*, 2024. [Online]. Available: https://rdkit.org/.
[30] RDKit, *Rdkit github*, 2024. [Online]. Available: https://github.com/rdkit.
[31] H. Loeffler *et al.*, "Reinvent4: Modern ai driven generative molecule design," *ChemRxiv*, 2023. DOI: 10.26434/chemrxiv-2023-xt65x. [Online]. Available: https://chemrxiv.org/engage/chemrxiv/article-details/65463cafc573f893f1cae33a.
[32] ChEMBL, *Chembl database*. [Online]. Available: https://www.ebi.ac.uk/chembl/.
[33] P. Ertl, "An algorithm to identify functional groups in organic molecules," *J. Chem. Inf.*, 2017. DOI: 10.1186/s13321-017-0225-z. [Online]. Available: https://jcheminf.biomedcentral.com/articles/10.1186/s13321-017-0225-z#citeas.