JOHANNES KEPLER
UNIVERSITY LINZ

Author
Mag.
Stephan Holzgruber

Submission
Institute for Machine Learning

Thesis Supervisor
Assoc.-Prof. Mag. Dr.
Günter Klambauer

Co-Supervisor
Philipp Seidl MSc.

January 2024

MolReactGen: Generating Molecules and Reaction Templates with a Transformer Decoder Model

Master Thesis
to obtain the academic degree of
Master of Science
in the Master's Program
Artificial Intelligence

JOHANNES KEPLER
UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Austria
www.jku.at
DVR 0093696

Kurzfassung

In der chemischen Forschung und den Life Sciences haben sich Künstliche Intelligenz Modelle als wesentliche Instrumente etabliert. Generative Modelle leisten einen wichtigen Beitrag bei der Erforschung und Entwicklung neuer Moleküle. Diese Technologien tragen zur Entdeckung neuer Medikamente bei und unterstützen die Entwicklung von Materialien, die für innovative und umweltfreundliche Energielösungen unerlässlich sind.

Ein signifikantes Beispiel für solche Modelle ist GuacaMol. Dieses Modell ist in der Lage, Moleküle mit definierten Eigenschaften zu generieren. Zudem stellt es Metriken für eine objektive Bewertung der generierten Moleküle vor. Eine zentrale Frage dieser Thesis ist, inwiefern ein Transformer-Decoder-Modell die Leistung des GuacaMol-Modells, insbesondere im Hinblick auf die Fréchet-ChemNet-Distanz, verbessern kann.

Ein weiterer wichtiger Aspekt ist der Einfluss der Datenaufbereitung und Tokenisierung auf die genannte Metrik. Hierbei wird die Hypothese untersucht, dass die Art und Weise, wie Daten aufbereitet und in das Modell eingespeist werden, relevante Auswirkungen auf das Ergebnis haben kann.

Des Weiteren wird die potenzielle Anpassung von GPT-2, einem vortrainierten Sprachmodell, für die Aufgabe der Molekülgenerierung erforscht. Die Frage ist, ob ein primär für menschliche Sprache entwickeltes Modell effektiv auf die Domäne der chemischen Modellierung übertragen werden kann.

Abschließend konzentriert sich die Arbeit darauf, ob die Erkenntnisse und entwickelten Modelle auf die anspruchsvolle Aufgabe der Generierung von Reaktionstemplates angewendet werden können. Dies würde einen weiteren Schritt in Richtung der Automatisierung und Effizienzsteigerung im chemischen Entdeckungsprozess darstellen.

Diese Arbeit zeigt, dass das Transformer-Decoder-Modell das GuacaMol-Modell hinsichtlich der Fréchet-ChemNet-Distanz übertrifft und auch bei der Generierung bekannter Reaktionstemplates erfolgreich ist.

Abstract

Artificial intelligence models have become indispensable in the fields of chemical research and life sciences. Generative models play a pivotal role in the innovation and development of novel molecules. These technologies are instrumental in the discovery of groundbreaking drugs and the creation of sustainable materials, which are pivotal for pioneering eco-friendly energy solutions.

A prime example of these models is GuacaMol. Distinguished by its proficiency in synthesizing molecules with specific attributes, GuacaMol also offers robust metrics for objective assessment of the molecules it produces. This thesis primarily investigates how the deployment of a transformer-decoder model could enhance GuacaMol's efficacy, particularly focusing on the Fréchet ChemNet Distance metric.

Moreover, this study delves into the impact of data pre-processing and tokenization on the aforementioned metric. It probes the hypothesis that the methodology of data pre-processing and its feeding into the model significantly influence the results.

In addition, this study explores the feasibility of adapting GPT-2, a pre-trained language model, for the task of molecule generation. It questions whether a model originally designed for human language processing can be effectively repurposed for chemical modeling.

Finally, this thesis assesses the applicability of the acquired insights and developed models for the complex task of generating reaction templates. This endeavor could mark a substantial leap towards automation and efficiency in the chemical discovery process.

The study indicates that the transformer decoder model surpasses the GuacaMol model in terms of the Fréchet ChemNet distance and is also successful in generating known reaction templates.

Contents

List of Figures

List of Tables

1. Introduction

In this thesis, we explore leveraging artificial intelligence (AI), particularly generative models, in the field of computational chemistry. The thesis is organized into four main sections: Introduction, Methodology, Results and Conclusion. In the Introduction, we contextualize our area of application within a broader spectrum of computational chemistry and AI. A review of related studies is presented to frame our study within existing literature. Following this, we articulate the research questions driving our work, which are designed to address some of the challenges posed by the domain. To set the stage for the subsequent sections, we provide a high-level overview of both the chemical domains under study, such as the generation of molecules and reaction templates, and the transformer architecture, a cornerstone in the realm of generative models.

In the Methodology section, we delineate the pipelines that guide our empirical studies. Here, we describe the datasets leveraged for our experiments and the pre-processing steps to make them suitable for machine learning models. This section outlines the training, generation, and evaluation procedures used in this study. The methodologies are framed within three distinct pipelines, each designed to address one or more of the research questions posited earlier.

The Results section is dedicated to synthesizing and discussing the data generated through our experiments. It presents an analysis of our findings and explains how they contribute to understanding the research questions. The interpretation of these results provides both scientific and practical insight.

In Conclusion, we take the opportunity to summarize the contributions of this research and identify potential avenues for future work. The overarching aim is not only to provide answers to our guiding questions, but also to inspire subsequent research endeavors at the intersection of AI and chemistry.

1.1. Application area

In the context of this thesis, “life sciences” and “chemistry” are related fields that intersect and complement each other. Life sciences encompass a broad range of scientific disciplines that study living organisms and their structures, functions, behaviors, and interactions. These include biology, biochemistry, genetics, and biotechnology. Life sciences provide valuable insights into the complexity of biological systems, from the molecular to the ecosystem level, and form the foundation for understanding human health, diseases, and the natural world.

On the other hand, chemistry focuses on the composition, properties, structure, and behavior of matter. It explores the interactions between atoms, molecules, and compounds. Chemistry is fundamental to many aspects of life sciences as it provides an understanding of how molecules interact and function within living organisms. It plays a crucial role in drug discovery, biochemistry, pharmacology, and various other areas within life sciences.

1.1.1. History and application of AI in life sciences

The application of AI in life sciences has reformed the way we approach and understand biological and biomedical systems [1, 2, 3]. The use of computational techniques in life sciences dates back to the mid-20th century, with the emergence of first-generation computers [4]. Early life science researchers relied on simple statistical methods to analyze biological data [5]. The development of neural networks in the late 1950s and the advent of genetic algorithms in the 1960s provided new tools for addressing biological problems. However, their application remained limited because of computational constraints [6, 7]. The advent of polymerase chain reaction (PCR) technology in the 1980s revolutionized the generation of biological data, laying the groundwork for high-throughput methods [8]. However, the data analysis remained challenging. It was not until the 1990s that machine learning (ML) [9] was applied in life sciences. Support vector machines (SVM) and decision trees became popular tools for the classification of biological sequences and structures [10]. The completion of the Human Genome Project in 2003 was a turning point in the integration of AI into life sciences [11]. The subsequent explosion of genomic data necessitated the development of ML techniques for processing and interpreting this information. During this period, ML played a significant role in gene annotation, understanding gene expression patterns, detecting genetic variations and mutations, and predicting disease susceptibility [12]. In the 2010s, deep learning techniques began to gain traction in life sciences, driven by advances in hardware, most notably graphic processing units (GPU), and the increasing availability of large datasets [13, 14]. Deep learning allowed for greater accuracy in tasks such as protein structure prediction [15, 16], non-coding RNA classification [17], and identification of disease-associated genetic variants [18, 19].

Complex models are now commonly used to analyze imaging data [20], sequence data [21], and electronic health records [22]. AI platforms also in aid drug discovery [23] and development, target identification [24], and clinical trials [25]. The advent of federated learning [26] has proven instrumental in harnessing the power of multi-institutional data while respecting privacy constraints. Generative models have been used to design novel therapeutic molecules [27]. The integration of explainable AI (xAI) approaches is another emerging trend that addresses interpretability issues in machine-learning models [28]. This is crucial in fields such as life sciences, where understanding the reasoning behind a prediction can be as important as the prediction itself.

One of the most important areas in which AI is applied in chemistry is drug discovery and design. The development of new drugs is both time consuming and costly. However, AI, particularly ML techniques such as deep learning, have significantly streamlined this process. Research is aimed at predicting the properties of new chemical entities and forecasting their behavior within biological systems. AI is being employed for in silico screening of vast chemical spaces for potential drug candidates, substantially reducing the time and cost compared to traditional high-throughput screening methods. Additionally, generative models are used to design novel chemical structures with desired therapeutic properties.

One of the main challenges in synthetic chemistry is predicting the outcomes of chemical reactions [29]. The use of AI in reaction prediction offers the potential to radically speed up the synthesis planning process. AI models trained on large datasets of known reactions can predict the products of a wide array of organic reactions. AI algorithms also optimize reaction conditions such as temperature, pressure, and catalysts, thereby improving the yield and selectivity of chemical reactions. AI has become an invaluable tool in materials science and is used to discover and design novel materials with tailored properties [30].

AI also helps solve the inverse problem, where, given a set of desired properties, the AI model can suggest potential chemical compositions and processing methods [31]. AI models can analyze complex chemical reactions in reverse, enabling chemists to identify starting materials and optimal pathways to efficiently synthesize compounds with desired attributes. These advancements not only streamline the synthesis process but also foster innovation in developing novel compounds and materials.

In summary, the journey of AI in life sciences has evolved from rudimentary applications to sophisticated algorithms that help decode complex biological systems. This has had a profound impact on our understanding of life at the molecular level, and has facilitated numerous breakthroughs in diagnostics, therapeutics, and personalized medicine. As data generation continues to grow and ML algorithms become more advanced, the possibilities for future applications of AI in life sciences seem immense.

1.1.2. Application of generative AI models

The use of generative AI models has offered a transformative perspective in the field of molecule generation, providing the ability to explore vast chemical spaces for novel structures [32]. However, despite their potential, certain challenges limit the effectiveness and applicability of these models.

Generating molecules

One of the primary challenges in molecule generation is to ensure that the generated molecules are chemically valid. The generated molecules should adhere to the principles of chemical stability and synthetic feasibility, and conform to the rules of chemistry. Addressing this challenge requires the proper design of generative models and inclusion of chemical rules.

The generated molecules can be biased towards the training data, producing molecules similar to those in the training set but lacking diversity. This bias could limit the exploration of novel chemical spaces and reduce the overall utility of generative models for molecule generation. Generative models face a tradeoff between novelty and similarity. Although the primary goal of these models is to generate novel molecular structures, it is critical that these molecules are not too far from known compounds to ensure their synthetic feasibility and biological relevance. Achieving a balance between novelty and similarity remains a challenge.

The choice of molecular representation significantly affects the performance of generative models. Currently, common representations include "Simplified Molecular Input Line Entry System" (SMILES) strings [33], molecular graphs [34], and 3D conformations [16]. Each representation has its strengths and limitations. For example, although SMILES strings offer a simple and compact representation, they occasionally result in invalid molecules. And while graph representations maintain chemical validity, they are computationally complex.

A key challenge in using AI models, including generative models for molecule generation, is their lack of interpretability. AI models might seem to act as "black boxes" and provide little insight into the decision-making processes. For generative models, the inability to understand why a certain molecule is generated hampers our ability to exploit these models effectively.

Evaluating the performance of generative models for molecule generation is a non-trivial task. Common metrics such as validity, uniqueness, and novelty provide a partial view of the model's performance but often fail to capture the real-world utility of the generated molecules. Establishing a comprehensive set of evaluation metrics that reflects both the chemical relevance and synthetic feasibility of the generated molecules is a challenge that needs to be addressed [35].

Distribution and goal-directed learning

Distribution and goal-directed learning represent two distinct paradigms in the field of generative models for molecule generation [36]. Distribution learning seeks to understand and replicate the underlying statistical properties of molecular data that govern the structure and behavior of molecules. The primary goal is to capture the complex relationships and dependencies between different molecular features, thereby allowing the generation of new molecules that adhere to the observed distribution. The methods employed in distribution learning are often probabilistic and unsupervised and focus on the entire data distribution.

Goal-directed learning is tailored to achieve specific objectives or to satisfy certain constraints. This might include designing molecules with desired pharmacological properties, optimizing stability, or adhering to specific synthetic pathways. The primary goal is not merely to replicate the observed distribution but also to guide the generation process towards pre-defined targets. Goal-directed learning uses guided and supervised techniques that incorporate specific feedback or constraints.

The choice between these approaches depends on the specific needs and goals of the research, and a hybrid approach can be employed to leverage the strengths of both paradigms. Understanding the main goals and differences between these learning strategies is essential for effective design and implementation of generative models in the rapidly evolving field of molecular science.

This study focuses on the distribution-learning paradigm of molecule and reaction template generation.

Generating reaction templates

Generative models for the generation of molecules have been explored extensively. While challenging, this task is relatively well-defined, as it focuses on exploring the space of individual molecules. Another critical aspect that requires attention is the generation of reaction templates [37], which presents challenges and complexities. The generation of reaction templates involves navigating an immensely large and complex space for chemical transformation. The sheer diversity of reactions, encompassing various bond formations, cleavages, rearrangements, and functional group modifications adds complexity to the generation process.

Molecule generation relies largely on unsupervised or weakly supervised approaches, allowing for more exploration and novelty. However, the generation of reaction templates requires contextual information and constraints. Reactions are influenced by factors such as the reaction conditions, catalysts, and reactant compatibility. Capturing and encoding these contextual aspects to ensure the generation of meaningful and synthetically feasible reaction templates are challenging.

Training generative models for molecules often benefits from large-scale databases containing known compounds and their properties. However, curated databases of well-defined reaction templates are relatively scarce, particularly when compared to the abundance of individual molecule data.

Reactions are fundamentally different from individual molecules because they involve multiple reactants, intermediates, and products. The sequential nature of reactions, with distinct steps and transformations including reactant ordering, bond breaking and forming, and stereochemical changes, poses challenges for generative models.

Evaluating the quality and correctness of generated reaction templates is a non-trivial task. Although metrics such as validity, novelty, and diversity are commonly used for molecule generation, the evaluation of reaction templates requires additional consideration. The generated templates should align with known chemical rules, be synthetically feasible, and exhibit utility for enabling the synthesis of novel molecules. The development of appropriate evaluation frameworks and benchmarks for generated reaction templates is an ongoing research challenge.

1.2. Objectives and scope

1.2.1. Motivation

There is a compelling case to invest effort and resources into advancing current research on generative AI for molecule and reaction template generation. These potential benefits are multifold and have implications across multiple disciplines, notably in the pharmaceutical, biotechnology, and materials sciences sectors.

The potential to accelerate drug discovery is a compelling reason for enhanced research in this field. Generative AI models can accelerate drug discovery by generating novel molecules and predicting potential synthetic routes, potentially saving millions of dollars and numerous years of research and development. Improved models can increase the likelihood of successful drug candidates and reduce the attrition rates in drug development pipelines.

There is a vast and largely unexplored chemical space containing molecules with beneficial properties. Traditional exploration methods tend to be slow, laborious, and expensive. Enhanced generative models can facilitate exploration of this chemical space in silico, leading to the discovery of novel molecules and reaction pathways.

The generation of new molecules and reaction templates could contribute significantly to the emerging field of personalized medicine. By tailoring molecules to individuals or groups of patients, health care outcomes can be improved. Generative AI models have the potential to contribute to the realization of truly personalized treatments.

Advancing generative AI models can provide environmental benefits. By predicting the most efficient synthetic routes, these models could reduce waste and minimize the use of harmful reagents, thereby contributing to greener chemical practices. Moreover, in silico generation and testing of molecules can reduce the need for physical experiments, save resources, and minimize the environmental impact.

The pursuit of improved generative AI models in chemistry represents a worthwhile endeavor that holds the promise of revolutionizing our approach to chemical research and application.

1.2.2. Baseline model

GuacaMol [36] was proposed by BenevolentAI as an evaluation framework designed to standardize the assessment of both classical and neural models for de novo molecular design. The GuacaMol framework provides a set of standardized benchmarks that can be used to measure various aspects of model performance. These benchmarks include the ability to generate novel molecules and their capacity for the exploration and exploitation of the chemical space.

GuacaMol´s Long Short-Term Memory (LSTM) model [38] is implemented in Python and is open-source, meaning that it is freely available for use and modification by the research community. A leaderboard showcasing the performance of various models on the GuacaMol benchmarks can be found on the BenevolentAI website.

We use GuacaMol as our baseline and a subset of its distribution-learning benchmarks to compare our models' performance with the models proposed in the GuacaMol paper. In addition to the distribution-learning task, GuacaMol includes goal-directed benchmarks for scoring the individual molecules. However, these benchmarks are beyond the scope of this study.

1.2.3. Research questions

The primary objective of this thesis is to explore the application of a transformer decoder architecture [39, 40] in the field of de novo molecular design, with a specific focus on comparing its performance with that of the GuacaMol framework. The study also investigates the impact of different tokenization approaches on model performance, the feasibility of using a pre-trained model, and the potential of the transformer decoder model to generate reaction templates. Specifically, we address the following research questions:

Performance comparison of the transformer decoder and GuacaMol — The first task is to compare the performance of the transformer decoder architecture with the GuacaMol framework. This study implements a transformer decoder architecture and compares its performance with the benchmarks featured in GuacaMol.

Effect of different tokenization approaches — The second task is to investigate the effect of different tokenization approaches on the performance of the transformer decoder model. Tokenization is a critical step in the pre-processing of data for ML models, and the choice of tokenization approach can significantly affect the model's performance.

Fine-tuning a pre-trained natural language model — The third task is to explore the feasibility of using a model pre-trained on natural language as a basis for fine-tuning a "molecule language" model. This approach, often referred to as transfer learning [41], has shown promise in various domains, and this study investigates its applicability in the field of de novo molecular design.

Generation of reaction templates — The final task is to investigate whether the transformer decoder model can also be used to generate reaction templates. Reaction templates are critical components of many chemical informatics systems, and their ability to automatically generate them could have significant implications in the field.

1.2.4. Related work

Many approaches have been developed and refined for the expansive landscape of generative models applied to chemistry domains. However, these methods often vary in several respects. The source and nature of data can significantly influence the performance and applicability of a generative model. Additionally, the architecture and design of generative models have demonstrated significant diversification. Different model classes have been employed in the context of chemistry, each with distinct strengths and limitations. The underlying objective of the generative model also varies. While some models may aim to generate molecules that are similar to a specific set of molecules, others look for novel molecular structures with desired properties. Finally, closely tied to the generation goals, the metrics used to evaluate the success of a model differ and can range from the accuracy in predicting molecular properties to the similarity of generated compounds with a given dataset.

There has been an influx of research papers, particularly in areas related to molecule generation and reaction prediction. However, a noticeable disparity seems to exist in the volume of papers dedicated to molecular generation compared to those focusing on reaction template generation. While the current literature landscape seems to show a clear dominance of molecular generation papers, there is a need to recognize and address the understated significance of reaction template generation. A balanced exploration of both domains is instrumental in bridging the gap between these two areas. The combination of high-quality reaction templates and molecular generation can streamline the process from molecular design to synthesis.

It is important to emphasize that the approaches discussed herein serve as exemplars and are by no means exhaustive. The ever-evolving nature of this field has led to the emergence of new methodologies and frameworks.

Related datasets

Generative models in the chemistry domain leverage various datasets for training purposes. Some of the more widely used datasets are as follows:

- ZINC [42] is a freely available database of commercially available compounds that can be used for virtual screening. It contains over 230 million purchasable compounds and is widely used in drug discovery projects and in deep learning models that generate molecular structures.
- The Molecular Sets (MOSES) dataset [43] was specifically designed to benchmark molecular generation models. It is derived from ZINC Clean Leads and provides a standardized benchmarking platform for molecular generation.
- ChEMBL [44] is a manually curated chemical database of bioactive molecules maintained by the European Bioinformatics Institute (EBI) [45]. It is a large-scale bioactivity database that contains binding affinity and functional activity data for numerous drugs and drug candidates. The ChEMBL dataset is a critical resource in drug discovery and cheminformatics. It contains over two million distinct compounds and their bioactivity data against more than 12,000 targets. The bioactivity data includes information on binding affinities and functional activities.

The foundation of this thesis is the GuacaMol dataset, derived from the ChEMBL database. The creators of GuacaMol post-processed the ChEMBL dataset to tailor it to the specific objectives pursued in their study. The detailed procedures involved in post-processing are described in the Appendix of the original GuacaMol publication.

The aforementioned datasets contain molecular structures. The generative models discussed in the next section demonstrate universal applicability that extends to various sequences of characters. Their adaptability has been demonstrated in their application to other domains such as protein sequences.

It is worth noting that the pre-processing, data augmentation, and feature extraction methods used can significantly influence model performance. For example, molecules can be represented in multiple ways, such as SMILES strings, molecular graphs, or fingerprints [46], and the choice of representation can affect the performance and capability of the model.

Related model architectures

Several architectures have been adopted and modified for generating molecules. Some of the most widely used architectures are listed in this section.

Recurrent neural networks (RNNs) were among the earliest applications of deep learning for molecular generation [47], especially their LSTM variant. The models were primarily trained on SMILES strings.

Both autoencoders [48] and variational autoencoders (VAEs) [49] were used for molecular generation. The idea was to encode a molecular representation into a latent space and decode it to generate new molecules.

Generative adversarial networks (GANs) were also applied for molecule generation [50]. The typical setup consists of a generator network that produces molecular structures, and a discriminator network that evaluates the authenticity of these structures. Adversarial training between these two forces the generator to produce increasingly realistic molecules.

Molecules can naturally be represented as graphs with atoms as nodes and bonds as edges. Graph neural networks (GNN) have become popular for modeling molecules, especially for property prediction [51]. However, they have also been adapted for molecular generation, enabling the direct generation of molecular graphs instead of sequences, such as SMILES.

Reinforcement learning (RL) techniques have been combined with the above architectures (such as RNNs and GNNs) to guide molecular generation towards specific desired properties [52]. The idea is to reward molecules that have the desired characteristics, enabling optimization of the generated molecules for certain targets.

Inspired by their success in natural language processing, transformer models, such as Bidirectional Encoder Representations from Transformers (BERT) [53] and Generative Pre-Trained Transformer (GPT) [40] variants, have been adopted for molecular tasks in the last few years. Transformers can be employed for both property prediction and generation by treating molecules as sequences.

Furthermore, significant emphasis has been placed within the research community on the fusion of multiple modalities in molecular representations. A prevalent approach encompasses the dual representation of molecules: primarily, employing the SMILES string representation to capture the molecular structure and concurrently providing a natural language narrative to describe its distinct characteristics [54]. This multi-faceted representation not only ensures a detailed understanding of the molecule but also offers richer data for generative models to harness.

It is important to note that although individual architectures have been developed chronologically, their application to molecular generation and design is often intertwined, and hybrid models that combine the features of multiple architectures are not uncommon.

Related benchmarking suites and metrics

Standardized benchmarking suites ensure that generative models can be evaluated and compared consistently. Several benchmark suites have been introduced to facilitate a fair and comprehensive evaluation of these models. In addition to GuacaMol, MOSES is another widely recognized benchmark suite.

MOSES is a benchmark suite that provides a standardized test bed for molecular generation models. It offers both distribution learning and goal-directed benchmarks. It includes metrics, such as drug-likeness, diversity, novelty, and synthesizability. As mentioned previously, the dataset is derived from the ZINC clean leads collection.

When evaluating the quality of generated molecules, it is essential to use multiple metrics together. Each metric provides insight into different aspects of the generation process, and a comprehensive evaluation requires consideration in combination. In addition, the relevance of specific metrics depends on the specific objectives and goals of the generative modeling task.

1.3. Life sciences background

1.3.1. Fundamentals of chemistry

The field of chemistry, a fundamental branch of physical science, holds a vital position within the framework of our understanding of the natural world. It provides a lexicon for interpreting and explaining the interaction, composition, and transformation of matter at the elementary level. Atomic theory, a core principle in chemistry, presents atoms as the smallest units of matter that retain the characteristics of an element. Atoms can bond to form molecules governed by principles such as the octet rule, which highlights the tendency of atoms to attain a stable configuration with eight electrons in their outermost shell. This dynamic interaction leads to the creation of a vast array of substances through chemical reactions, that is, the transformation of reactants into products. Chemistry further expands to include studies of various states of matter, along with the exploration of chemical bonds, periodicity, energetics, and kinetics. It provides insights that drive advancements across diverse fields, from medicine to materials science, environmental science, and nanotechnology.

1.3.2. Molecules

Molecules represent a fundamental concept in chemistry, and are critical for understanding the nature of various forms of matter. A molecule is defined as a group of two or more atoms bonded together and represents the smallest unit of a chemical compound that retains its unique chemical properties. They range from simple diatomic molecules, such as molecular oxygen (O_2) to complex macromolecules, such as proteins, which are composed of thousands of atoms.

Molecules are the primary actors in chemical reactions. They participate in collisions and interactions that break and form bonds, leading to the formation of new molecules. The rates at which these reactions occur can be studied using the principles of chemical kinetics.

The role of AI in the study of molecules has become increasingly important. ML models can predict molecular properties, simulate chemical reactions, and aid in the design of new molecules for pharmaceutical and material sciences. The ability to handle the vast and complex space of possible molecular structures makes AI a valuable tool in contemporary chemistry and other related disciplines.

The representation of molecules

The representation of molecules suitable as inputs for AI models constitutes a crucial challenge for the integration of chemistry with AI. Molecules are conventionally illustrated using chemical diagrams or molecular formulas. However, such representations are unsuitable for direct use in AI models because of their lack of numerical information, prompting the need for alternative approaches that capture the intrinsic characteristics of molecules in a machine-interpretable format.

Molecular descriptors and fingerprints are among the most widespread alternatives [55] for describing the structure and properties of a molecule. Molecular descriptors provide numerical or Boolean indications of molecular characteristics, such as size, shape, polarizability, and electrostatics. These features can range from simple scalar values such as molecular weights to multidimensional vectors. An advantage of molecular descriptors is their ability to provide quantitative insights into molecules, allowing for the correlation between structure and properties to be discerned.

By contrast, molecular fingerprints convert the chemical structure of a molecule into a binary bit string, where each bit represents the presence or absence of a particular substructure or pattern within the molecule. Different fingerprinting algorithms can capture various levels of molecular detail [56]. For example, molecular access system (MACCS) keys [57] provide a set of 166 pre-defined structural fragments, whereas extended connectivity fingerprints (ECFP) [58] represent atom neighborhoods with increasing diameters.

Another prominent approach is the SMILES notation [33], which is a textual representation method that transcribes the 2D structure of a molecule into a line of characters. This string can be directly processed by AI models or encoded further to yield more expressive representations. The SMILES encoding process is non-unique, meaning that the same molecule can be represented by multiple valid SMILES strings, thereby introducing an additional layer of complexity.

An alternative graphical approach that utilizes graph-based representations. where molecules are depicted as graphs with atoms as nodes and bonds as edges. This representation is advantageous for AI models based on graph theory, such as GNNs, which can analyze the relationships between nodes while preserving the local spatial information of the molecule.

An emerging method involves the use of 3D voxel grids to represent molecules [59]. This approach divides the 3D space around a molecule into a grid and assigns a value to each voxel based on the presence and type of atoms within that voxel. This representation is particularly useful for 3D convolutional neural networks, as it captures not only the molecular structure, but also its 3D spatial orientation and shape.

Each of these approaches has its strengths and limitations, and is more suitable for certain types of AI models or tasks than others. The challenge lies not only in representing a molecule in a format that an AI model can process but also in accurately capturing the complex characteristics and behaviors of molecules that are relevant to the task at hand. Balancing computational efficiency, representation complexity, and predictive performance is pivotal to the successful application of AI in chemistry.

The SMILES format

The textual representation system known as SMILES has been widely adopted to characterize and communicate molecular structures. SMILES is a simple yet powerful language that encodes molecular structures into linear strings of ASCII characters, thereby facilitating the exchange and manipulation of molecular information.

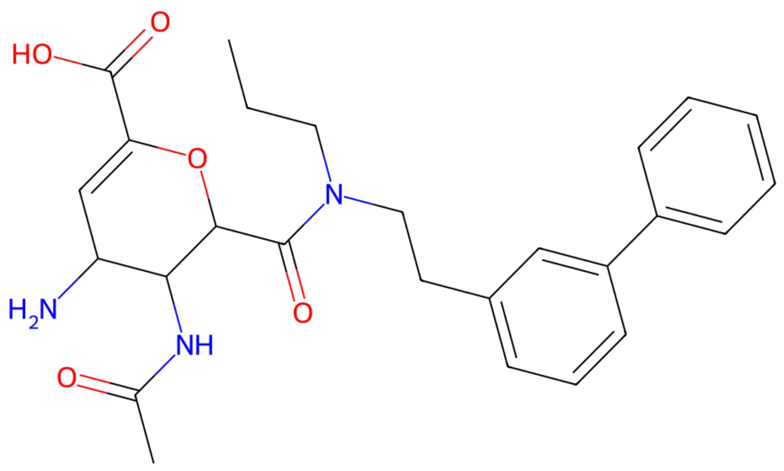

Figure 1.1.: Molecular structure of a sample SMILES, visualized with `RDkit` [60]:
`CCCN(CCc1cccc(-c2ccccc2)c1)C(=O)C1OC(C(=O)O)=CC(N)C1NC(C)=O`

A SMILES representation starts with atomic symbols to denote individual atoms (e.g., C for carbon and O for oxygen) and uses additional syntax to illustrate the connections and configurations of these atoms. A single bond is represented by simple concatenation (e.g., CO represents a molecule with a carbon atom bonded to an oxygen atom); double, triple and quadruple bonds are denoted by the symbols "=", "#" and "$", respectively. The ring structures are expressed using numerical identifiers. Stereochemistry, another critical aspect of molecular structures, is also encoded in SMILES strings through symbols such as "@" and "/".

A unique feature of SMILES is its ability to convey substantial molecular information in a compact and readily manipulable form. This feature is particularly valuable in the development and application of ML models for tasks such as molecular property prediction, drug discovery, and synthetic route planning. With proper encoding of the molecular structure, these models can understand and learn the underlying patterns and correlations in chemical space, thereby driving their predictive performance.

However, a challenge associated with SMILES is the existence of multiple valid SMILES strings for a single molecule, owing to its inherent flexibility in describing the same molecular structure in various ways. While offering a higher dimension of representation, this issue introduces complexity in handling and interpreting data. To overcome this, canonical SMILES, a unique representation of each molecule, can be generated using specific algorithms.

The advent of SMILES has significantly influenced the computational field of chemistry. It has not only streamlined data management in chemistry, but also opened up new avenues for the application of AI in the field, paving the way for advancements in chemical prediction, drug design, and beyond.

1.3.3. Chemical reactions

Chemical reactions transform a set of chemical substances into another. It involves the breaking and formation of chemical bonds and the rearrangement of atoms. Reactants are substances that initiate a chemical reaction. These molecules are present before the start of the reaction. Reactants interact with each other, often in a specific manner owing to their molecular structure, to undergo a chemical reaction. The chemical reaction can occur in various ways such as synthesis, decomposition, single replacement, double replacement, and combustion reactions. The type of reaction typically depends on the nature of the reactants and conditions under which the reaction occurs. The products are the compounds produced by the reaction, and their properties often differ from those of the reactants. The conditions of a chemical reaction are external factors that influence the reaction. These include the temperature, pressure, presence of catalysts, and reactant concentration. For instance, increasing the temperature frequently increases the reaction rate, because it provides the energy required to break the bonds in the reactants. Ligands are ions or molecules that bind to central atoms to form coordination complexes. They can influence chemical reactions by changing the reactivity of the central atom. The nature and arrangement of the ligands around the central atom can significantly affect the outcome of the reaction.

The representation of chemical reactions

Reaction templates [37] refer to generalized patterns or frameworks that capture the essential elements and characteristics of a particular class of reactions. These templates provide a concise representation of the fundamental steps involved in a reaction, including the reactants, products, and transformation of chemical bonds. By utilizing reaction templates, chemists can efficiently analyze and predict the outcomes of similar reactions, thereby aiding in the design and synthesis of new molecules.

Reaction templates are typically constructed based on common reaction patterns observed in experimental data and their theoretical understanding. These templates serve as guides for chemists, enabling them to identify the underlying mechanisms and pathways that govern specific types of reactions. For instance, a template for a nucleophilic substitution reaction may include the presence of a nucleophile, electrophile, or transfer of a leaving group.

One of the primary advantages of using reaction templates is their ability to facilitate knowledge transfer among chemists. By employing a standardized set of templates, researchers can communicate and share information about reactions more effectively. This helps in establishing a common language and understanding within the chemical community, promoting collaboration, and advancing the field as a whole.

Reaction templates are valuable tools for computer-aided synthesis planning and retrosynthesis analysis [61]. ML algorithms can be trained on reaction databases to automatically recognize and apply reaction templates. This allows for the prediction of viable reaction pathways and generation of synthetic routes for target molecules. The utilization of reaction templates in computational tools significantly enhances the efficiency and accuracy of chemical synthesis planning.

The SMARTS format

The "Smiles Arbitrary Target Specification" (SMARTS) format [62] is a widely used molecular pattern language that allows for the specification of reaction templates in a concise and flexible manner [63]. It is an extension of the SMILES notation, which is used to represent chemical structures.

For example, the SMARTS pattern `[CX3]=[OX1]` represents a carbonyl group with low specificity, and represents carboxylic acid, ester, ketone, aldehyde, carbonic acid/ester, anhydride, carbamic acid/ester, acyl halide, and amide [64].

The SMARTS format also enables chemists to define reaction templates by describing the structural features and connectivity patterns of molecules involved in a reaction. It incorporates logical operators, as well as special atomic and bond symbols. These operators enable the specification of complex reaction patterns that involve multiple reactants and products.

A sample reaction template is `[C;D1;H3:3]-[NH;D2;+0:4]-[CH2;D2;+0:1]-[c:2]>>Br-[CH2;D2;+0:1]-[c:2].[C;D1;H3:3]-[NH2;D1;+0:4]`.

The SMARTS format is widely used in cheminformatics and computational chemistry for tasks such as substructure searching, reaction prediction, and reaction enumeration. It provides a versatile and expressive language for defining reaction templates, allowing chemists and researchers to capture and analyze the structural characteristics of chemical reactions in a computationally accessible format.

1.4. Fundamentals of machine learning

1.4.1. Deep neural networks

Deep neural networks (DNN) [65], a subset of ML techniques [9], fall under the broader umbrella of AI. These networks are similar to multi-layer perceptrons (MLP) [66], a specific type of artificial neural network (ANN). They are structured as layers of interconnected nodes or neurons [67], each capable of processing and transmitting information to the subsequent layers. The term "deep" in deep learning signifies the depth of the layers within these networks. Although a universally accepted definition remains elusive, a deep learning model is typically characterized by more than a single layer and can extend to several hundred hidden layers. Each of these layers is typically comprised of more than 100 neurons. To facilitate the learning of intricate patterns, these models incorporate activation functions such as the Rectified Linear Unit (ReLU) [68] and Scaled Exponential Linear Unit (SELU) [69]. These functions introduce non-linearity into the model and enhance its learning capability. A distinguishing feature of deep learning models that distinguishes them from traditional ML algorithms is their ability to learn features autonomously. Traditional algorithms often require manual feature extraction, whereas deep learning models can automatically discern features that aid in accomplishing tasks such as classification. This capacity to learn directly from raw data underscores one of the principal advantages of deep neural networks.

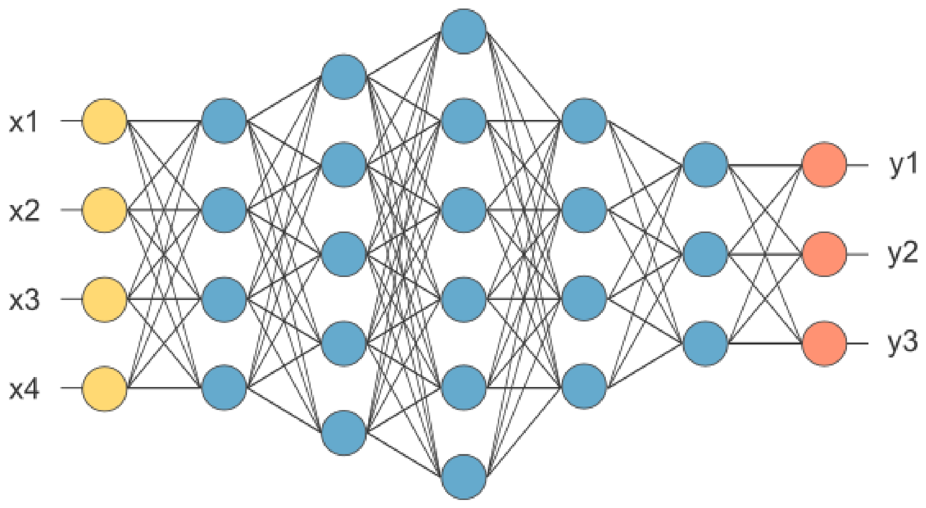

Figure 1.2.: Sample deep neural network: the yellow (left, x_i) nodes represent the input layer, the blue (middle) nodes represent the hidden layers and the red (right, y_i) nodes represent the output layer. Source: Kottas (2017) [70]

1.4.2. Learning

AI employs an array of training methods that facilitate tailored approaches to specific tasks. These paradigms form the backbone of AI, empowering it to learn, adapt, and address multi-faceted problems.

Supervised learning [71] is a common AI training method. It operates based on the principle of learning from labeled data by incorporating both input variables and the corresponding correct outputs. The task of the algorithm is to learn a mapping from the inputs to the outputs and make predictions based on this relationship. The algorithm's output is corrected whenever the predictions are incorrect, mirroring a teacher-student dynamic.

Supervised learning is typically used for classification and regression tasks, where the goal is to predict outcomes based on given data.

In contrast to supervised learning, *unsupervised learning* involves training AI algorithms with data that have no labels, which means that the correct output is unknown. The objective of this approach is to identify hidden patterns and structures inherent to the input data [72]. This is similar to how humans learn and comprehend large volumes of information by establishing patterns. Unsupervised learning is extensively used for clustering [73], where the task is to group subsets of entities based on their similarity, and anomaly detection, which involves identifying outliers or unusual data points in a dataset.

Semi-supervised learning is a hybrid of supervised and unsupervised learning methods. It uses a blend of a small amount of labeled data and a large volume of unlabeled data. The aim is to leverage labeled data to assist in the learning process and categorization of unlabeled data. This training method is beneficial when labeling data is costly or impractical and is quite similar to supervised learning in its implementation.

In *reinforcement learning* [74], an artificial agent interacts with its environment, for which the state is known, to achieve a goal. The agent can perform several actions within the environment and observe outcomes or feedback, which are termed rewards or penalties. The goal of the agent is to learn a policy, that is, a strategy for selecting actions that maximize the cumulative reward over time. Reinforcement learning operates on a trial-and-error basis by learning from experience in order to inform future decisions. This type of learning is widely used in real-world applications including gaming, robotics, and resource management.

1.4.3. Training paradigms

In the field of AI, particularly deep learning, model development typically falls into two broad categories: training models from scratch, and transfer learning. We use the terms transfer learning and fine-tuning pre-trained models interchangeably, although fine-tuning is sometimes described as a subset of transfer learning. Both approaches have unique characteristics, with distinct advantages and disadvantages that may significantly influence researchers' choices in the context of a particular research project or application.

Training models from scratch

Training models from scratch involves initializing a neural network with random weights and training it on a specific dataset. This approach requires a careful architectural design, hyperparameter tuning, and extensive training.

The first salient advantage of this approach is customization. When training a model from the ground up, the researcher has full autonomy over the selection of the architecture and the tuning of hyperparameters. This enables the creation of models that are finely tailored to specific tasks or domains, thus optimizing performance for particular applications. Control and specificity in model development can lead to innovations that may not be possible with pre-trained alternatives.

Training from scratch ensures that there is no prior bias in the model. Unlike models that are fine-tuned from existing pre-trained versions, where influences from previous data or tasks may persist, a model trained from scratch is guided solely by the task at hand. This focus can prevent inadvertent biases and confounding influences, providing a more unbiased understanding of the relationship between input features and the target variable.

However, this approach is challenging. Resource intensity is one of the most prohibitive factors that should be considered. Training a model from scratch can require substantial computational resources and time, often necessitating access to powerful hardware and extensive processing time. This not only increases the financial burden, but also limits the feasibility of the approach in constrained environments.

Furthermore, the effectiveness of training from scratch is closely related to the data requirements. A well-performing model typically requires a large amount of data. The procurement and preparation of such data can be both challenging and costly, particularly in specialized domains where examples may be scarce or expensive to produce. This dependence on large quantities of specific data can hinder the applicability of training from scratch in situations where data availability is limited.

Fine-tuning a pre-trained model

Fine-tuning refers to the process of taking a pre-trained model that has been trained on a general task and adapting it to a specific task by continuing training on a new dataset. Fine-tuning pre-trained models has emerged as an instrumental practice in ML, particularly in scenarios where there is a need to build a sophisticated model without the considerable resources that training from scratch might demand.

There are three primary strategies for fine-tuning. The *feature extraction* approach leverages the pre-trained model as a fixed feature extractor, in which the existing layers are used to transform the input data. Additional layers are then trained on the new task using these transformed features. This allows the new task to benefit from the generalized representations learned by the pre-trained model without altering its underlying structure.

The *full fine-tuning* method adjusts the entire network. By fine-tuning all the layers, it provides a higher degree of adaptation to the new task. Different learning rates can be employed for different layers, offering nuanced control over the adaptation process.

Rather than fine-tuning the entire network, the *partial fine-tuning* approach focuses on a specific subset of the layers. By keeping certain layers frozen and fine-tuning others, it is possible to balance the retention of general features with the acquisition of task-specific knowledge.

One of the most compelling advantages of fine-tuning pre-trained models is their efficiency. By capitalizing on existing architectures and weights, fine-tuning often accelerates the development process, mitigating the need for long and computationally expensive training cycles.

This approach can deliver a strong performance even with smaller datasets. Pre-trained models have already extracted generalized features from extensive data, and this knowledge can be leveraged for new tasks. Mitigation of the need for vast amounts of specialized data makes fine-tuning particularly beneficial in domains in which acquiring ample data may be challenging or costly.

Despite these advantages, the approach of fine-tuning pre-trained models also carries inherent challenges that warrant careful consideration.

An unavoidable concern when utilizing pre-trained models is the risk of inheriting biases from the original training data. These biases can subtly but significantly influence the behavior of the fine-tuned model, leading to skewed or unintended outcomes. Rigorous examination of the pre-trained model's background and meticulous evaluation of its potential biases are essential to prevent the inadvertent propagation of these biases.

The second challenge revolves around the limited flexibility in modifying the architecture. Major alterations to the structure of a pre-trained model may lead to incompatibilities with existing parameters. This constraint can limit the level of customization and adaptation that can be achieved, potentially hindering the optimal alignment with the specific requirements of the new task.

1.4.4. Loss functions

The concept of loss functions, sometimes referred to as cost functions or error functions, plays an instrumental role in training and optimizing machine-learning models [75]. They provide a quantitative measure of how far the predictions of an ML model deviate from actual outcomes. A loss function is used to evaluate how well an algorithm models a given dataset. If the predictions deviate significantly from the actual results, the loss function outputs a high value. Conversely, if they are similar, the loss function returns a low value. By minimizing this function, an ML model can learn the optimal parameters to predict outcomes with greater accuracy.

In mathematical terms, a loss function L is defined as $L(y, g(\boldsymbol{x}; \boldsymbol{w})))$ or $L(y, \hat{y}))$, where y is the "ground truth", and $\hat{y}$ is the prediction of model g with input $\boldsymbol{x}$ and parameters $\boldsymbol{w}$.

The landscape of loss functions is diverse and tailored to suit the various forms of data and ML algorithms. We discuss a selection of the most commonly used regression, classification and contrastive loss functions.

- Mean Squared Error (MSE): This is the average of the squared differences between the predicted and actual values. This function places more weight on large errors because of the squaring operation.
- Mean Absolute Error (MAE): This calculates the average of the absolute differences between the predicted and actual values. Unlike MSE, it treats all errors the same, regardless of their magnitude.
- Huber Loss: The Huber Loss offers a blend between the MSE and MAE. It calculates the squared difference between the predicted and actual values for small errors (typically for absolute differences less than a certain threshold δ, known as the Huber threshold), similar to the MSE, and the absolute differences for larger errors, similar to the MAE.
- Binary Cross-Entropy or Log Loss: This function is used in binary classification problems. It quantifies the dissimilarity between predicted probabilities and actual binary values.

- Categorical Cross-Entropy: Used for multiclass classification problems, it quantifies the difference between the predicted probability distribution and the actual distribution. We provide a formal definition here because we use categorical cross-entropy as the loss function in our study:

$$L(y, \hat{y}) = -\sum_{i=1}^{K} y_i \log \hat{y}_i \tag{1.1}$$

  where K is the number of categories or classes, y is the true distribution, represented as a one-hot encoded vector, where one of the components is one for the correct class and the rest are zero, and the vector $\hat{y}$ represents the predicted probabilities, where each component value is the predicted probability of the instance belonging to class i.

- Contrastive loss functions are primarily used in learning embeddings or representations, particularly in tasks such as similarity learning. They are designed to help the model learn to distinguish between similar and dissimilar instances. The main idea is to reduce the distance between similar pairs of instances, while pushing dissimilar pairs further apart. Examples of contrastive loss functions include Contrastive Language Image Pre-training (CLIP) [76], Contrastive Leave One Out Boost (CLOOB) [77] and Contrastive Learning and leave-One-Out-boost for Molecule Encoders (CLOOME) [78].

The choice of loss functions can significantly influence the performance of ML models. It is not merely about model accuracy, but also about the model's capacity to generalize well to unseen data. Different loss functions have different sensitivities to outliers in the data, which can affect the robustness of a model. Moreover, loss functions should be chosen in line with the specific characteristics of the problem to be solved, for instance, whether the cost of false positives and false negatives is asymmetrical.

1.4.5. Back-propagation and gradient descent

In the context of AI, specifically in the domain of neural networks, both gradient descent [79] and backpropagation [80] are instrumental concepts that facilitate model optimization. Optimization refers to the process of determining the most effective parameters that minimize the objective function, typically the loss function.

Backpropagation is used to compute the gradient of the loss function with respect to the weights in the network. This is because the calculation proceeds backward through the network, starting from the output layer and moving towards the input layer. The principle of backpropagation is fundamentally based on the chain rule of calculus, allowing partial derivatives to be effectively computed, even in complex, multi-layered networks.

Gradient descent is an optimization algorithm that aims to minimize the loss function. The gradient is used by gradient descent to update the weights in the network. The weights are adjusted in the opposite direction to the gradient, and the size of the adjustment is controlled by a parameter known as the learning rate. A smaller learning rate might cause the algorithm to converge slowly, whereas a larger rate could lead to overshooting the minimum. Therefore, setting a good learning rate is crucial for the effectiveness of back-propagation.

In a mathematical context, the gradient of a function is a multivariate derivative that provides the direction of steepest ascent at a particular point. A negative gradient is used to identify the direction of the steepest descent, which is the direction in which the function decreases the fastest.

To be applicable, the loss function must be differentiable, that is, it must have a derivative at every point in its domain. Differentiability ensures that the gradient can be calculated at any point during the optimization process. Furthermore, the loss function is often considered to be a convex function, which is a special type of function in which the line segment between any two points on the function is above or on the graph. Convex functions have a global minimum that can be effectively determined by gradient descent.

Both processes are repeated until the model's performance on the data is satisfactory or until a set number of iterations is completed. Convergence refers to the point at which further iterations do not reduce the loss function significantly. This implies that the algorithm determines a local or global minimum. The rate of convergence can depend on factors such as smoothness of the loss function, learning rate, and initialization of the parameters. However, despite the loss having plateaued and the performance having ceased to improve significantly, there may still be several options to improve the performance of the model, for example, with different weight initializations, learning rate schedulers, and different optimizers.

1.4.6. Maximum likelihood estimation

Maximum likelihood estimation (MLE) is a fundamental concept in statistical inference and ML [81]. It provides a method for estimating the parameters of a statistical model by maximizing the likelihood function. The likelihood function quantifies the probability of observed data for a given set of parameters. The parameters that maximize this likelihood are called the maximum likelihood estimates and are considered the most plausible parameters given the observed data.

Mathematically, given a set of observations $X = x_1, x_2, \ldots, x_n$ and a statistical model with parameters θ, the likelihood function $\mathcal{L}(\theta; X)$ is defined as the probability of observing data X given the parameters θ. The maximum likelihood estimate of θ, denoted by $\hat{\theta}$, is the value of θ that maximizes $\mathcal{L}(\theta; X)$.

MLE plays a pivotal role in various AI domains. In ML, MLE is used to estimate the parameters of models such as linear regression. In natural language processing (NLP), the model parameters are adjusted to maximize the likelihood of training data consisting of large amounts of text. This allows the model to learn the underlying patterns in the data and generate new text that is similar to the training data.

1.4.7. Statistical significance of results

In scientific research, obtaining empirical results is a critical part of the experimental process [82]. However, mere observation of a difference between two or more sample means is not sufficient to conclude that a real underlying difference exists in the population from which the samples were drawn. Random variability and sampling errors can lead to apparent differences, which are simply the result of chance.

To address this issue, statistical significance tests are performed. They provide a formal mechanism for evaluating whether the observed differences are likely to be genuine or whether they could have occurred by chance. By quantifying this likelihood, statistical tests enable researchers to make informed judgments regarding the validity and reliability of their results.

Statistical tests offer several benefits. They provide a measure of the probability that the observed difference occurred by chance, represented by a p-value. This offers a quantified measure of confidence in the findings. Statistical tests also allow for controlled and systematic comparisons between different models, algorithms, and experimental conditions, thereby providing a basis for conclusions grounded in statistical theory.

These are typical high-level steps in statistical tests:

- Define the hypotheses: Clearly state the null hypothesis H_0 (e.g., no difference between the means) and alternative hypothesis H_a (e.g., a significant difference between the means).
- Determine the significance level α: Establish a threshold value α (typically 0.05 or 0.01), below which the null hypothesis is rejected. This value represents the probability of rejecting the null hypothesis when it is true, essentially controlling for the Type I error rate.
- Choose the appropriate test: Select the statistical test that is suitable for the data type and distribution, such as a t-test for comparing means.
- Calculate the test statistic: Apply the chosen test to the data and obtain a test statistic and associated p-value.
- Interpret the results: Compare the p-value to the chosen significance level α and draw a conclusion on whether to reject or fail to reject the null hypothesis.
- Report the findings: The results, including α, p-value and sample size, convey statistical evidence for or against the hypothesis being tested.

In summary, statistical significance is pivotal for robust evaluation of experimental results in AI research. By going beyond mere comparisons of means and incorporating statistical tests, statistically sound conclusions can be drawn.

1.5. Transformer architecture

1.5.1. History

The following is a non-exhaustive list of key developments in the history of transformers [83]. The field continues to evolve rapidly, with new architectures and applications being developed regularly.

In 2017, Vaswani et al. introduced the transformer model in their seminal paper “Attention is All You Need” [39]. This model proposes a novel architecture that relies on self-attention mechanisms, eliminating the need for recurrence and convolutions.

Researchers at Google developed BERT in 2018 [53]. BERT has revolutionized the field of natural language processing by using a bidirectional transformer.

In 2019, OpenAI released GPT-2, a transformer decoder-based large language model (LLM) that showcased the power of unsupervised learning [40]. It was trained to predict the next word in a sentence, and demonstrated a remarkable ability to generate coherent and contextually relevant sentences.

In 2020, OpenAI released GPT-3 [84], an even larger version of GPT-2. With 175 billion parameters, it demonstrated an unprecedented ability to generate human-like text and showed a surprising amount of “knowledge” and “understanding” despite being trained only to predict the next word in a sentence.

In 2020, Google extended the application of transformers to computer vision tasks using a Vision Transformer (ViT) [85]. They demonstrated that transformers pre-trained on large-scale image datasets can outperform convolutional networks for image classification tasks.

One year later, OpenAI introduced CLIP (Contrastive Language-Image Pretraining), a model that connects vision and language using transformers [86]. It can understand images using natural language, and has spurred the development of multimodal models.

Introduced in 2023, GPT-4 marks the evolution of OpenAI’s transformer-based language model series [87]. By leveraging vast training data, it significantly improves its predecessor, GPT-3, in terms of understanding the context and generating relevant responses.

ChatGPT [88], an application of the GPT model, has seen remarkable refinement in its conversation capabilities in each version. The latest release, based on the GPT-4 architecture, is adept at handling complex conversational contexts and providing a more natural, coherent, and contextually accurate interaction.

The transformer model represents a significant milestone in the evolution of deep-learning models. Their innovative architecture and self-attention mechanism have enabled the development of models that can “understand” and generate human language.

In our study, we selected the GPT-2 model variant with a reduced hidden dimension for our experiments, as it empirically maintains an optimal balance between computational demands and the capacity to effectively learn the distribution of the training data.

1.5.2. Architecture fundamentals

This section explores the fundamental concepts underlying the transformer architecture and its key components. Transformers have become common, and because our model is very close to the original, we omit an exhaustive background description of the model architecture. Instead, we select and describe one of its main contributions, the scaled dot-product attention, and refer to the seminal paper [39], its update in 2023 [89], and several online guides [90].

We use the terms “word” and “sentence” to ease our understanding of how a transformer model processes human language. In more abstract terms, the transformer model processes a sequence of tokens. A token typically refers to the smallest input unit that a model can process. In NLP, a token can be as small as a character or as large as a word, depending on the model’s design and the language it is processing. For models such as GPT-2, a token is often a sub-word unit, which means that a word can be split into multiple tokens based on common sub-word patterns in the training data.

In this context, a sentence generally refers to a sequence of tokens that represent a complete thought, traditionally ending with punctuation, such as a period, question mark, or exclamation point. However, it is important to note that transformer models do not inherently understand the concept of a sentence, as humans do. Instead, they process inputs and generate outputs based on patterns learned from the data irrespective of the sentence boundaries. Therefore, in these models, a sentence is merely a token sequence.

The transformer model is based on the encoder-decoder structure, where the encoder maps an input sequence to a continuous representation that captures the semantic information of the input and the decoder generates an output sequence from this representation. The heart of the transformer model is the self-attention mechanism, also known as scaled dot-product attention. This mechanism allows the model to weigh the relevance of different words in a sentence, while processing a particular word.

The formula for the scaled dot-product attention [39] is

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1.2}$$

where Q denotes the matrix of queries, K the matrix of keys, and V the matrix of values. The dot product of each query and key indicates the similarity between them. A larger dot product indicates that the query and key are more similar. Thus, the corresponding value should be given more attention. The dot product is then divided by the square root of d_k which is the dimension of queries and keys. This ensures that the dot products do not become too large, which in turn helps to stabilize the gradients during training.

Researchers have subsequently amended the original transformer model to optimize it for specific use cases or for efficiency reasons. Among the initial modifications were the equalization of the key and value matrices and the order of the network layers, particularly the transfer of the normalization layer before the self-attention layer [53].

Because the transformer model lacks recurrence and convolutions, it does not have a notion of the order of words in a sentence. To address this issue, the model uses fixed or learned positional encoding, which injects information regarding the position of words in a sentence into the model. This allows the model to consider the order of words when making predictions.

The transformer model consists of stacks of encoders and decoders. Each encoder in the stack has two sublayers: a self-attention layer and feed-forward neural network. Each decoder also has two similar sub-layers, along with a third sublayer that performs multi-head attention over the output of the encoder. This architecture allows the model to generate complex representations of the input sequences and produces accurate output sequences.

1.5.3. Tokenization

Tokenization is a crucial step in data pre-processing, particularly in NLP [91]. This involves breaking down data structures such as text into smaller, more manageable units known as tokens. In the context of text data, these tokens are often individual words or parts of words.

The tokenization process has several purposes. First, it simplifies the data, making it easier for the models to interpret and learn. Second, it allows for the extraction of meaningful features from the data, which can then be used to train the model. Finally, tokenization can help to reduce the dimensionality of the data, thereby making the model training process more efficient.

Tokenization plays a pivotal role in the model training. The tokens generated through this process serve as the inputs for the model. These tokens are typically converted into numerical representations such as vectors, which can be processed by the model. This conversion is typically achieved using techniques such as one-hot encoding [92] or word embeddings [93].

Furthermore, the quality of tokenization significantly affects the performance of the model. Effective tokenization can lead to more accurate and meaningful feature extraction, which, in turn, can lead to better model performance. Conversely, poor tokenization can result in the loss of important information, which can negatively impact the model's ability to learn from the data.

Anatomy of a tokenizer

These are the key components of the tokenizer and their role in the tokenization process.

Normalization is the first step in the tokenization pipeline, which aims to transform the input text into a standardized format that makes it "cleaner" and easier to process. Common operations in normalization include the following.

- Stripping whitespace: Removing leading and trailing spaces to ensure consistent formatting.
- Removing accented characters: Replacing accented characters with their unaccented counterparts to avoid discrepancies caused by different character representations.
- Lowercasing all text: Converting all text to lowercase to eliminate case sensitivity during tokenization.

Proper normalization ensures that the tokenization process starts with consistent and uniform text.

Pre-tokenization is the act of breaking down input text into smaller units, often referred to as pre-tokens. The pre-tokenization step prepares the text for the subsequent tokenization model. A common approach is to split the text into individual "words" or units that are likely to be part of the final tokens.

For example, in English, pre-tokenization may involve splitting text based on space, punctuation marks, and other delimiters. However, the final tokens may be parts of those "words," depending on the tokenization model used.

The *tokenization model* or *tokenization algorithm* is a critical component of the tokenizer pipeline. This is applied to the pre-tokens generated in the pre-tokenization step. The tokenization model is responsible for dividing the pre-tokens into final tokens, which are used as inputs for NLP models.

The tokenization model must be trained on a dataset representative of the language and domain for which it is used. Various tokenization algorithms exist, such as Byte Pair Encoding (BPE) [94], WordPiece [95], and Unigram [96], which are frequently used in state-of-the-art NLP models.

After tokenization, *post-processing* is performed to apply additional transformations to the encoded tokens. A common use is the addition of special tokens to a token sequence. Special tokens include padding, beginning-of-sequence (BOS), end-of-sequence (EOS), or domain-specific tokens. Post-processing allows the tokenizer to include essential information that helps NLP models understand the context and boundaries of the text.

The *decoding* step is the final phase of the tokenization process, in which the numerical representation of the tokens obtained after tokenization is converted back into a human-readable text using the tokenizer's vocabulary. During decoding, additional special tokens are removed to retrieve the original sequence of the tokens.

If the tokenization model utilizes sub-tokens to represent parts of a word (e.g., using the "##" symbol in WordPiece), custom decoding logic may be required to handle such sub-tokens appropriately.

Tokenization models

Lookup tables, which are also known as *dictionary-based* or *word level* tokenizers, are the simplest tokenization models. They segment text into tokens based on a pre-defined dictionary or word vocabulary. Lookup tables are simple, straightforward, and relatively efficient because they do not require any computationally heavy lifting. Their primary limitation is dealing with out-of-vocabulary (OOV) words. If a word is not present in the vocabulary, the tokenizer cannot handle it effectively, which leads to a loss of information. Furthermore, they do not account for morphological variations in words, limiting their effectiveness.

Byte Pair Encoding (BPE) begins with the vocabulary of individual characters, and iteratively merges the most frequent pair of symbols to create a new symbol until a pre-defined vocabulary size is reached. This approach allows BPE to handle rare and complex words by breaking them down into smaller and more common subwords. The BPE vocabulary must be calculated beforehand, which can be a computational burden for large datasets. BPE can be applied to any text-based dataset, and there are specific pre-trained implementations for SMILES [97].

Byte-level BPE is a specific type of BPE. It operates at the byte level and can handle text data irrespective of the language. This resolves a significant issue in tokenization: the need to predefine vocabulary or to deal with out-of-vocabulary words. Byte-level BPE can segment any string of text into a sequence of known subwords or, in the worst-case scenario, into individual bytes, thereby ensuring that all the input text can be processed.

In this manner, large language models can handle rare words, which is a typical challenge when dealing with diverse and extensive language datasets. However, byte-level BPE is not without its limitations. The first pertains to the efficiency. Owing to the byte-level operation, a substantial number of tokens may be required to represent a single word, particularly for languages with large character sets. This can make the processing less efficient compared with a higher-level tokenization approach.

Plain BPE operates based on a fixed list of common words and sub-words. If it encounters a word that is not in its vocabulary, it does not have a reliable way of handling it. Byte-level BPE, on the other hand, has a solution for any out-of-vocabulary word.

For GPT-2, byte-level BPE was chosen as the tokenization strategy, owing to its robustness and universality. GPT-2 was designed as a general-purpose language model capable of handling a diverse array of text data, making the wide-ranging capabilities of byte-level BPE ideal. Despite potential efficiency trade-offs, the ability to never encounter an out-of-vocabulary word is a key factor driving this choice.

WordPiece is similar to BPE but has a slight difference in the merging process. Instead of always merging the most frequent pair, WordPiece selects the merge that maximizes the likelihood of the training data, given the language model. This makes WordPiece more flexible in deciding which sub-words to form. It has been widely used in models such as BERT, demonstrating its effectiveness.

Unlike BPE and WordPiece, the *Unigram* tokenization model starts with a large vocabulary and iteratively removes tokens. It uses a unigram language model and removes tokens based on the likelihood of the training data. This approach allows Unigram to handle a wide range of languages.

1.5.4. Generative models

The generative capabilities of transformer models are a significant factor in their widespread adoption, particularly for text generation. This section delves into the autoregressive text generation aspect [98] of transformer decoder models, with a specific focus on GPT-2 as an example of this architecture.

Generative models are a class of models that learn the probability distribution of their input data and subsequently generate new data samples via stochastic sampling from this distribution. This process is inherently probabilistic, that is, each generated sample is a random draw from a learned distribution. This allows the model to generate a wide variety of samples, each of which is a plausible example. This is one of the key strengths of generative models, because it allows them to generate diverse and realistic samples.

Autoregressive text generation is a process in which a model generates a sequence token by token, using previously generated tokens as the context for generating the next. Token masking is a crucial strategy for this method. The model is trained such that it can only see the previous tokens and the current token being processed; the future tokens are masked. This mimics the scenario during prediction, where the future tokens are unknown. In transformer models, this is typically achieved using the decoder part of the architecture. The decoder uses an input token and a continuously updated internal state that encapsulates information from the previously generated tokens to generate the next token in the sequence.

GPT-2, developed by OpenAI [99], is a prime example of an autoregressive transformer decoder model. Unlike the original transformer model, which uses both an encoder and decoder, GPT-2 only uses the decoder part of the architecture. This is because GPT-2 is designed for tasks that require text generation, such as language modeling, in which the goal is to predict the next word in a sentence, given the previous words.

1.5.5. Modifying pre-trained tokenizers

Pre-trained models are generally accompanied by pre-trained tokenizers, which are responsible for transforming raw text into a format that is understandable by the model.

The tokenizer is vital because it maps textual input into numerical tokens that correspond to embeddings in the model. Alterations in tokenization methods and vocabulary can significantly affect a model's performance and adaptability. The necessity to modify pre-trained tokenizers can arise when the task targets different languages that are not included in the pre-training data or when domain-specific terminology is not covered by the original tokenizer. In addition, optimization considerations can lead to adaptation of the tokenizer to cater to specific computational or efficiency constraints.

Various methods, including vocabulary expansion and adaptation, can be used to modify tokenizers. *Vocabulary expansion* involves adding new tokens to the vocabulary of a pre-trained model. This addition can help adapt the model to new fields without the need for retraining, allowing for better understanding and capture of nuances. However, an increased vocabulary size may lead to a longer tokenization process and require more memory. Additionally, the newly added words might not align perfectly with the existing word embeddings, which could degrade performance.

Vocabulary adaptation is a modification of existing vocabulary to suit a specific task or domain. Efficiency and performance can be improved by tailoring the vocabulary to a particular task. Furthermore, by removing unnecessary tokens, the tokenization process can be sped up, thereby reducing computational overhead. However, overspecializing the vocabulary might lead to a loss in the model's ability to generalize across various tasks, and incorrect adaptation could cause incompatibility with pre-trained embeddings, resulting in suboptimal performance.

1.5.6. Sampling strategies

The characteristics of the generated data depend on the learned probability distribution as well as the sampling strategy used to draw samples from this distribution. In this section, we delve deeper into some of the most widely used sampling strategies for generative models. These strategies play a crucial role in determining the quality and diversity of samples generated by these models. The strategies discussed include greedy search, beam search [100], multinomial sampling, beam search multinomial sampling, top-k sampling [101], and top-p (nucleus) sampling [102].

A *greedy search* is the simplest sampling strategy. This involves selecting the next most probable item at each step of the generation process. Although this approach is computationally efficient, it often leads to suboptimal results because it does not consider the overall sequence probability, and can become stuck in a locally optimal solution and produce uniform text.

Beam search is a sophisticated strategy that maintains a "beam" of the most promising sequences at each step. The width of the beam, known as the beam width, determines the number of sequences maintained at each step. Although beam search can yield better results than greedy search, it is more computationally intensive and can still be stuck in locally optimal solutions.

Multinomial sampling, also known as random sampling, involves selecting the next item based on its probability distribution. This strategy introduces randomness into the generation process, leading to more diverse results. However, this can also lead to less coherent sequences as low-probability items are selected.

Beam search multinomial sampling combines the strengths of beam search and multinomial sampling. It maintains a beam of the most promising sequences, such as a beam search, but selects the next item based on its probability distribution, as in multinomial sampling. This strategy can lead to more diverse and higher quality results than either strategy alone.

Top-k sampling creates a balance between diversity and quality. This involves selecting the next item from the top k most probable items. This approach can lead to more diverse results than greedy search or beam search while still maintaining a high level of quality. *Top-p sampling*, also known as nucleus sampling, is a variant of top-k sampling. Instead of selecting from the top k most probable items, it selects from the smallest set of items whose cumulative probability exceeds a threshold p.

Some strategies prioritize quality, others prioritize diversity, and some strike a balance between the two. Understanding these strategies and their trade-offs is essential for effective use and development of generative models. Some of these strategies, such as multinomial sampling, have hyperparameters that can influence sampling results. For example, the "temperature" adjusts the probability distribution used for sampling. A high temperature results in a more uniform distribution (increasing diversity), whereas a low temperature results in a more peaky distribution (increasing determinism).

These sampling strategies are not the only examples. Other variants include contrastive search [103] and diverse beam search decoding [100]. More recent models, such as ChatGPT, involve additional model classes such as reinforcement learning (RL) to improve sampling.

1.5.7. Evaluation

Evaluating generative models is a non-trivial task, given the absence of an immediate loss function after training the model. Model evaluation requires intricate methods that allow us to determine how well the model has learned to generate new instances that convincingly emulate the distributions found in the training data.

The realm of text generation has seen substantial progress, and the advent of metrics has been designed to quantify the quality of the generated text. A critical requirement for these metrics is their ability to align with human assessments of text quality in order to provide meaningful interpretations of a model's performance.

A well-established example is the Bilingual Evaluation Understudy (BLEU) score, originally developed to assess machine-translation systems [104]. It measures the overlap of n-grams between the generated and reference texts, rewarding the models to match additional and longer n-grams. However, it is important to note that while the BLEU score offers valuable insights, it might not fully capture nuances such as semantic and stylistic elements in the generated text. Thus, it should not serve as the sole metric for text quality.

In contrast to text, the domain of generating molecules and reaction templates presents unique challenges when assessing the quality of generated results. Initial metrics, such as validity (that is, whether the generated molecular structure is chemically valid), are only sufficient for basic tasks, but can be easily manipulated by models to achieve superficially high scores without genuinely learning the underlying molecular distributions [35].

A more promising metric is the Fréchet ChemNet Distance (FCD) [105], which is analogous to the Fréchet Inception Distance (FID) used in image generation tasks [106]. The FCD measures the Fréchet distance [107], also known as the Wasserstein-2 distance [108], between the activations of a pre-trained ChemNet [109] for real and generated molecules, providing a more comprehensive evaluation of the model's ability to mimic the distribution of the training data. Using ChemNet, a deep learning model trained on a large set of chemical structures, the FCD measures the similarity between the distributions of generated and real-world molecules, making it a more nuanced and holistic measure of model performance. Nevertheless, the FCD is based on a pre-trained model, and its effectiveness depends on how well this pre-trained model captures the essential features of the molecular structures. This makes it susceptible to limitations of the underlying ChemNet model. Several studies [36, 43] have sought to establish this metric as part of a robust framework for assessing the quality of generated molecules.

The formula for calculating the FCD is as follows:

$$d^2\left((\boldsymbol{m}, \boldsymbol{C}), (\boldsymbol{m}_w, \boldsymbol{C}_w)\right) = \|\boldsymbol{m} - \boldsymbol{m}_w\|_2^2 + \operatorname{Tr}\left(\boldsymbol{C} + \boldsymbol{C}_w - 2\left(\boldsymbol{C}\boldsymbol{C}_w\right)^{1/2}\right) \tag{1.3}$$

Here, $d\,(.,.)$ is the Fréchet distance between the Gaussian distribution $p_w(.)$ with mean $\boldsymbol{m}_w$ and covariance $\boldsymbol{C}_w$ obtained from the GuacaMol molecules, and the Gaussian distribution $p(.)$ with mean $\boldsymbol{m}$ and covariance $\boldsymbol{C}$ representing the generated molecules. The two distributions are the activation values of the penultimate layer of the underlying ChemNet model, and are assumed to follow a multidimensional Gaussian distribution.

Reaction templates, which are used to predict chemical reactions, pose an even more complex evaluation challenge. There appears to be a lack of consensus on a universal metric that is suitable for this task. In this study, we propose that a generative model for reaction templates can be considered successful if it generates known templates which the model has not been trained on.

Nevertheless, it is important to recognize that these metrics can only provide a partial view of a model's capabilities. A comprehensive evaluation should consider a combination of various measures, coupled with human assessment where feasible, to ensure that the generated outputs are both novel and of high quality, and can be synthesized in a wet lab.

Select challenges in evaluating generated molecules

Similar to every evaluation metric, the evaluation metrics for generated molecules encounter challenges and pitfalls. In this section, we will focus on the intrinsic challenges presented by molecule generation metrics, specifically referencing the "copy problem" outlined by Renz et al. [35].

This study highlights a particularly striking issue in the evaluation of generative molecular models. In their study, they described the "copy problem," in which newly generated molecules were produced by inserting a single carbon atom into random positions in molecules taken directly from the training set. This seemingly simplistic model, although not generating chemically significant molecules, outperformed many intricate models on several distribution-learning benchmarks. The heart of this issue is that traditional metrics are potentially rewarding trivial modifications to known molecules rather than true novelty or chemical significance.

While most metrics were deceived by the “copy problem,” it is worth noting the performance of the FCD in this scenario. The FCD reported a value of 0.871 (with 0.0 the theoretical best value) for carbon-added molecules, which is considerably worse than the FCD values of various state-of-the-art models. This shows the sensitivity of the FCD to such naive approaches, thus proving to be a potentially more robust metric against trivial alterations. Nevertheless, even though the FCD showed resistance to the “copy problem,” it is crucial to recognize that no single metric is entirely foolproof, and that a combination of metrics, human expertise, and domain-specific knowledge remains indispensable.

Human expertise is essential not only for gauging the novelty or potential efficacy of a molecule but also for evaluating its synthetic accessibility. Regardless of how promising it looks on paper, a molecule is of limited utility if it cannot be practically synthesized. Hence, synthetic accessibility is a cornerstone challenge that can affect the utility of generated molecules. That is, even the most refined evaluation metrics are rendered moot if the molecules they approve cannot be translated into real-world applications. Further complicating is the fact that rigorous assessment of synthesizability often necessitates experimental validation in a wet lab, a process that is both costly and time-consuming.

Gao et al. provides a beacon of hope, demonstrating that molecules generated using distribution-learning methods exhibit a synthesizability frequency comparable to that of the molecules in their training set [110]. This similarity in the synthesizability rates underscores the potential of generative models to produce real-world molecules.

Enhanced approaches involving the application of post-hoc filtering using auxiliary models designed to gauge synthesizability appear promising. By leveraging these models, one can cherry-pick compounds that not only exhibit advantageous theoretical metrics, but also have a high probability of successful synthesis.

2. Methodology

To explore the research questions outlined in Section 1.2.3, this section focuses on the design and implementation of three distinct pipelines, each with its own unique purpose and methodology for addressing specific research questions. Each pipeline includes data collection, data tokenization, model configuration, model training, generation, and evaluation. Each of these components plays a vital role in the overall functioning of the pipeline and contributes to the achievement of research objectives.

The primary aim of pipeline 1 is to compare the performance of the transformer decoder architecture with the benchmarks featured in GuacaMol. This comparison is crucial for understanding the effectiveness and efficiency of the transformer decoder model. Additionally, pipeline 1 focuses on analyzing the impact of different tokenization approaches on the performance of the model. This analysis helps to identify the most effective tokenization approach and its influence on the overall performance of the model.

Pipeline 2 is designed with the objective of investigating the feasibility of using a pre-trained natural language model as a foundation for fine-tuning a "molecule language" model. This investigation is crucial in understanding the potential of pre-trained models in enhancing the performance and accuracy of molecule generation models.

The goal of pipeline 3 is to determine whether the transformer decoder model can be used to generate reaction templates. This exploration is important to understand the versatility and applicability of the transformer decoder model in various contexts.

2.1. Pipeline 1 – Molecules from scratch

In this section, we present a pipeline designed to benchmark the performance of a GPT-2 transformer decoder model against GuacaMol. Figure 2.1 shows an overview of pipeline 1.

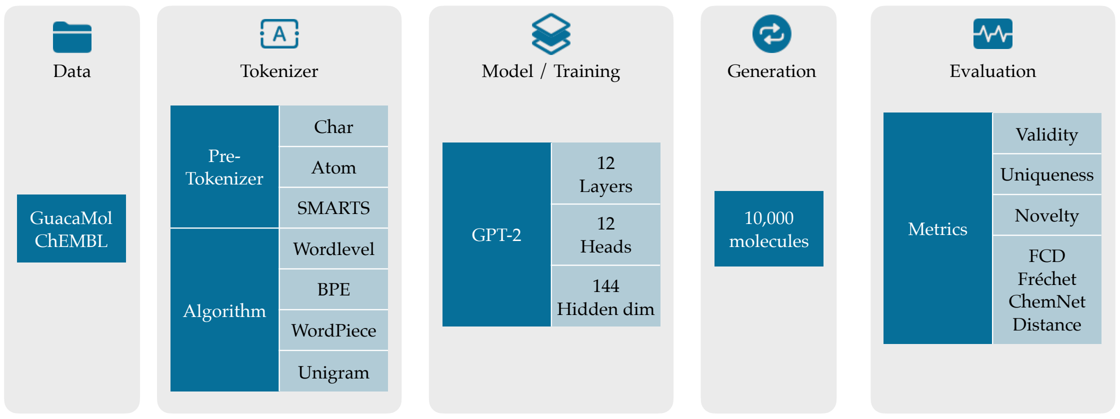

Figure 2.1.: Overview of pipeline 1

2.1.1. Data and tokenization

The GuacaMol dataset is based on the ChEMBL database, a large-scale bioactivity database for drug discovery that represents molecules in SMILES format. Table 2.1 and Figure 2.2 provide summary information regarding the number of molecules and their length distribution.

Data Split	Count	Percentage
Training	1,273,103	80%
Validation	79,567	15%
Test	238,705	5%
Total	**1,591,375**	**100%**

Table 2.1.: Number of molecules and their splits in the GuacaMol dataset.

The role of tokenizers in LLMs is a major area of investigation in this study. The fundamental objective of our research is to assess the impact of varying tokenization methodologies on these language models. To address this comprehensively, we utilize three different pre-tokenization options coupled with four distinctive tokenizer algorithms.

This method allows us to analyze the differential impact of these permutations on the performance of the LLM. However, it is essential to clarify that not all combinations of pre-tokenization options and tokenizer algorithms are feasible. Table 2.3 provides an overview of the valid pairs used in this study. The results obtained from these permissible

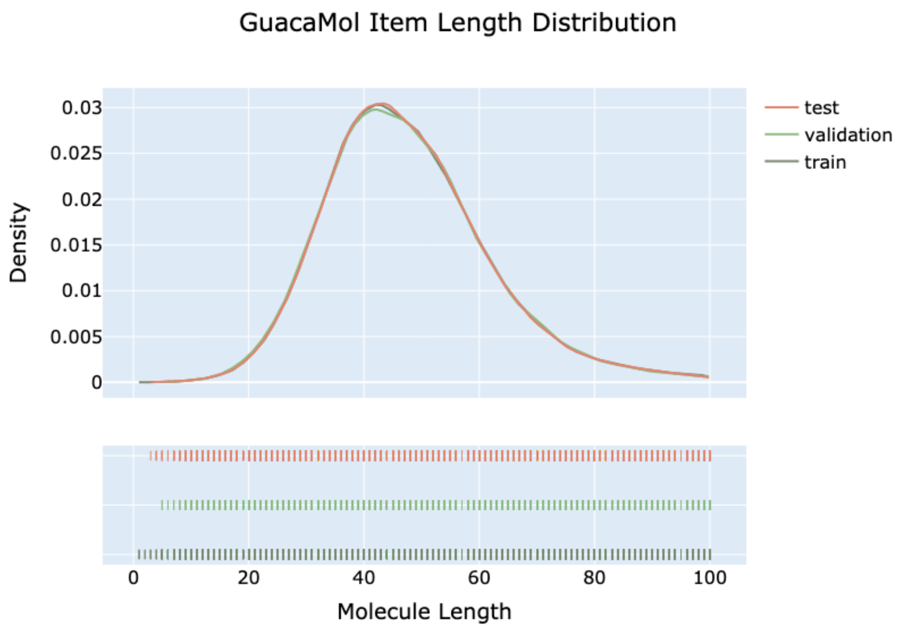

Figure 2.2.: Length distribution of SMILES strings for each data split. All datasets share the same average length of approximately 48 characters.

combinations offer a more profound understanding of the interplay between tokenizer choices and the efficacy of LLMs.

Pre-tokenizers

The *Char* or *Wordlevel* pre-tokenizer is a simple approach that splits the input text character-wise. Each character is treated as an individual token, resulting in a highly granular vocabulary.

The *Atom* pre-tokenizer utilizes regular expressions (RegEx) to identify chemical symbols with more than one character, such as Cl, and translates them into a single token. This pre-tokenizer is specifically tailored for processing chemical data, where atomic symbols play a crucial role in representing molecular structures. By treating chemical symbols as individual tokens, the Atom pre-tokenizer helps capture the essential information encoded by these symbols.

The *SMARTS* pre-tokenizer is an extension of the SMILES RegEx, as proposed by Schwaller et al. [111], and is designed to handle chemical structures represented using SMARTS notation. SMARTS notation allows for more complex representations of molecular patterns, including substructures enclosed in square brackets. The SMARTS pre-tokenizer treats all these chemical structures as single tokens, thereby preserving their semantic integrity.

Section A.1.1 in the Appendix lists the RegEx patterns for the Atom and SMARTS pre-tokenizers, and Table 2.2 presents the resulting tokens.

Pre-tokenizer	Resulting tokens
Char	O [C l +] O
Atom	O [Cl +] O
SMARTS	O [Cl+] O

Table 2.2.: The pre-tokenizer options and their resulting sample tokens

After the pre-tokenization process, we apply the tokenization algorithms described in Section 1.5.3.

Our primary objective was to combine all three pre-tokenizer options with each of the four tokenization algorithms (WordLevel, BPE, WordPiece, and Unigram). For these algorithms, we aimed to experiment with three distinct vocabulary sizes of 44, 88, and 176 tokens. It is important to note that our definition of vocabulary size does not encompass special tokens such as BOS and EOS. Unfortunately, because of a potential issue encountered with the Hugging Face tokenizers library [112], we were restricted to using the tokenization algorithms exclusively with the Char pre-tokenizer. Furthermore, the minimum vocabulary size for the WordPiece algorithm was 76, and considering its proximity to the vocabulary size of 88, we chose to omit this particular experiment. The resulting successful tokenizer configurations are presented in Table 2.3.

Pre-tokenizer	Tokenization Model			
	WordLevel	BPE	WordPiece	Unigram
Char	38	44, 88, 176	88, 176	44, 88, 176
Atom	50	n.a. due to implementation issues [112]		
SMARTS	106			

Table 2.3.: Valid combinations of pre-tokenizers, tokenization models, and vocabulary sizes

2.1.2. Model

Our research employs OpenAI's GPT-2 transformer decoder model, which has been recognized for its performance in various NLP tasks. This model is a variant of the transformer architecture that has gained immense popularity in the field of NLP and generative tasks. Unlike the full transformer architecture, which includes both encoder and decoder components, we exclusively use the decoder part for our task. GPT-2 comes in multiple "sizes" that differ in the number of layers, the number of heads, and the hidden dimension.

To make even the "small" GPT-2 model more suitable for our specific task and resource constraints, we employ several parameter reduction techniques. We maintain the original 12 layers and 12 heads of the transformer decoder. Although increasing the number of layers and heads could potentially enhance the performance of the model, it also increases the computational cost. By adhering to the original configuration, we strike a balance between the performance and computational efficiency. The hidden dimension in the transformer decoder directly affects the expressiveness and memory requirements of the model. To achieve a good trade-off between the model performance and computational resources, we reduce the hidden dimension from 768 to 144. This empirical decision is based on experiments to determine an optimal setting that maintains satisfactory performance while reducing the memory footprint of the model. It is also a multiple of 12 (heads), as required by the architecture, and 8, as recommended by NVIDIA for the GPU architecture [113]. The maximum sequence length affects the memory consumption of the model during training and inference. To accommodate all molecule tokens without splitting the input sequence, we set the sequence length to 128. This choice ensures that the information of the entire molecule is fed into the model without compromising the context provided to the decoder.

By employing the aforementioned parameter-reduction techniques, our model is tailored for efficient training and inference, while maintaining its capacity to generate meaningful and contextually coherent sequences. The final model obtained after these adjustments has approximately three million parameters, making it a computationally feasible and effective solution for our specific generative task.

2.1.3. Training

The training process follows the standard gradient descent algorithm (forward propagation, loss calculation, backpropagation, and optimization) applied to many deep learning models. We choose AdamW as the optimizer and learning rate scheduler, with warm-up during the initial stages of training and cosine annealing during training. The following section presents the chosen parameters in more detail.

Parameter	Options	Selection
Sequence length	–	128
Number of layers	–	12
Number of heads	–	12
Hidden dimension	–	144
Dropout	–	0.1
Number of epochs	–	50 with early stopping
Batch size	32, 64, 128	64
Gradient accumulation steps	–	4
Floating point precision	FP32, FP16	FP16
Loss function	–	Cross entropy without label weights
Label smoothing	0.0 (None), 0.1	0.0 (None)
Learning rate	0.0005, 0.0025	0.0025
Learning rate scheduler	–	Cosine annealing with warmup, warmup ratio 0.1
Gradient clipping	–	1.0
Optimizer	–	AdamW weight decay = 0.1 beta 1 = 0.9 beta 2 = 0.95

Table 2.4.: Training (hyper-) parameters

It should be noted that the choice of vocabulary size is not merely a decision about the linguistic scope of the model but also influences the statistical characteristics of the model's output probability distribution. As we increase the vocabulary size, the probability mass has more tokens to be distributed over, often leading to a reduction in the probability assigned to the most likely token. This change generally increases the cross-entropy loss.

When coupled with a specific learning rate, the increased loss value can significantly affect weight updates during the backpropagation step. Consequently, the optimal learning rate for one tokenizer may not be directly transferred to another tokenizer with a different vocabulary size. Therefore, when reporting the "best" learning rates, it is crucial to specify that these rates are optimal only for tokenizers with comparable vocabulary sizes.

2.1.4. Generation

Once the transformer decoder model has been trained for molecular generation, it can be used to generate molecules. This process is performed in an autoregressive manner, that is, the model uses its predictions as inputs for subsequent predictions.

The generation process begins with prompting the model with a BOS token. The model then generates a subsequent token based on the initial input. The generated token is concatenated to the BOS token and fed back into the model as the next input. This process is repeated until an EOS token is generated or the model's maximum sequence length is reached.

Each sequence generated by the model represents a potential molecule. However, not all the sequences correspond to valid molecules. To test the validity of the generated sequence, we attempt to parse it using the chemo-informatics library `RDKit` [60]. If `RDKit` can successfully parse the sequence, we consider the molecule to be valid.

The generation process continues until either a set number of valid molecules has been generated or a condition is met, which indicates unsuccessful generation. This condition is satisfied when the model generates a configurable number of invalid molecules in a row. This serves as a safeguard against the model from becoming stuck in a state in which it continually generates invalid molecules.

Table 2.5 lists the hyperparameters of the generation process.

Parameter	Options	Selection
Generation method	—	Multinomial sampling
Beams	1, 5	1
Length penalty	0.0, 1.0	1.0
Repetition penalty [114]	1.0, 1.2	1.0

Table 2.5.: Hyperparameters for molecule generation

To provide initial feedback on the generation performance of the model, we calculate some preliminary metrics during generation. These include uniqueness to assess the proportion of generated molecules that are distinct from each other and novelty to measure the proportion of generated molecules that are not found in the training set. These metrics provide valuable insights into the ability of the model to generate diverse and novel molecules. The evaluation section discusses these metrics in detail.

2.1.5. Evaluation

In this section, we delve into the established metrics that we apply based on the definitions provided in the GuacaMol study.

Validity is defined as the fraction of valid molecules among all generated molecules. Specifically, a molecule is considered valid if it can be parsed by `RDKit`. This implies that the generated structure adheres to the basic rules of chemistry, thereby ensuring that the generated molecules are chemically meaningful. Caution with validity is that a model can potentially learn to game this metric by merely producing chemically stable but uninteresting or overly simplistic molecules. Therefore, although validity is an essential first step, it alone is insufficient to gauge the overall performance of a model.

Uniqueness is calculated as the fraction of unique molecules among the valid molecules. A molecule is deemed unique if it has not been previously generated in the same generation run. This metric checks the capacity of the model to produce diverse molecular structures rather than repeatedly generating the same structures. However, uniqueness does not determine whether these unique molecules are useful. A model can generate a high percentage of unique, trivial, or known molecules, thereby illustrating the need for more comprehensive metrics.

Novelty is defined as the fraction of novel molecules among the unique molecules. A molecule is considered novel if it is not found in the training set. This measure pushes the model beyond reproducing known molecules from training data, thereby encouraging the discovery of potentially new molecular structures. The challenge of the novelty metric lies in ensuring that the novel molecules are not just different, but also chemically interesting and potentially useful. This necessitates additional considerations, such as synthetic feasibility.

The *FCD* (see Section 1.5.7) is calculated using the generated valid molecules and all molecules in the training set as the reference set. Guacamol post-processes the FCD as $\exp(-0.2 \times \text{FCD})$ and we report it as $\text{FCD}_{\text{GuacaMol}}$.

Table 2.6 summarizes the metrics used to evaluate the generated molecules.

Metric	**Formula**	**Target value**	**Comment**
Validity	$\frac{\text{molecules}_{\text{valid}}}{\text{molecules}_{\text{generated}}}$	$\uparrow 1.0$	Valid $\widehat{=}$ generated molecule can be parsed by rdkit
Uniqueness	$\frac{\text{molecules}_{\text{unique}}}{\text{molecules}_{\text{valid}}}$	$\uparrow 1.0$	Unique $\widehat{=}$ valid molecule generated only once
Novelty	$\frac{\text{molecules}_{\text{novel}}}{\text{molecules}_{\text{unique}}}$	$\uparrow 1.0$	Novel $\widehat{=}$ unique molecule not in training set
FCD	See Section 1.5.7	$\downarrow 0.0$	The similarity between two sets of molecules, in this case the GuacaMol training set and the generated *valid* molecules
$\text{FCD}_{\text{GuacaMol}}$	$\exp(-0.2 \times \text{FCD})$	$\uparrow 1.0$	

Table 2.6.: Metrics to evaluate the generated molecules

Determining the “best” tokenizer

To determine the optimal tokenizer, we must first address a fundamental issue. Directly comparing the validation loss of the models as a representation of model performance is not feasible, primarily because diverse tokenizers yield varying vocabulary sizes. As mentioned before, these discrepancies in vocabulary sizes have a direct influence on the model’s output, specifically the probability distributions, which in turn impact the scale of the cross-entropy loss. Consequently, we choose the FCD as our representative metric for model performance. Our approach begins by training a model for each tokenizer, ensuring a consistent random seed. Subsequently, 10,000 molecules are generated for each model, again maintaining a fixed random seed. The performance assessment of these molecules hinges on the FCD, serving as both a proxy and pivotal metric for model performance. The tokenizer exhibiting the superior FCD is preliminarily designated as our initial “best” tokenizer. To reinforce our findings and augment the likelihood of genuinely pinpointing the most effective tokenizer, we proceed with a statistical test process.

Following our preliminary identification of the initial “best” tokenizer, we continue our experiments to substantiate our findings. We train five distinct models employing this initially selected “best” tokenizer, but this time introduce variability with a random seed. This methodology yields five models. By leveraging each of these models, we generate five sets of 10,000 molecules, while maintaining a consistent seed during the generation process. This exercise provides us with five FCD values along with other metrics, enabling us to compute both the mean and sample standard deviation of those values. To validate whether our preliminary “best” tokenizer truly excels, our first hypothesis test involves assessing the normal distribution of the FCD values using a Shapiro-Wilk test. Subsequently, we conduct a one-sample single-sided Student’s t-test to determine which tokenizers are outperformed (based on FCD) by our initial “best” tokenizer. Within this context, “outperforming” is defined as indicating statistical significance that allows us to reject the null hypothesis of identical means. All tokenizers for which we cannot reject the null hypothesis become our “candidate” tokenizers, and we initiate five training and generation steps with each of those tokenizers. This allows for undertaking either a two-sample, single-sided Student’s t-test or, if the standard deviations diverge by more than a factor of two, Welch’s t-test. Ultimately, this testing process allows us to pinpoint the optimal tokenizer or, in instances where the differences are not statistically significant, to identify a set of top-performing tokenizers.

Comparing our result with the GuacaMol paper

To juxtapose our findings with those presented in the GuacaMol paper, we adopt the optimal model and tokenizer, respectively, as determined by our detailed selection procedure. Using this chosen model and tokenizer, we generate five sets of 10,000 molecules, diverging from our previous approach by introducing a random seed. This modification aims to amplify the variability within the generation process, subsequently enabling us to compute both the mean and standard deviation of the FCD. Using these statistical insights, we conduct a one-sample, single-sided Student’s t-test. This final analytical step facilitates a comprehensive evaluation, allowing us to compare the performance of our model with that of the GuacaMol model and provide a conclusive statement on the relative performance of our model.

All statistical tests were performed with a significance level of $\alpha = 0.05$. Table 2.7 summarizes the processes outlined above.

	Step 1	**Step 2**	**Step 3**
Main task	Determine the initial best tokenizer	Determine the final (set of) best tokenizer(s)	Compare performance with GuacaMol
Training iterations	11, one for each tokenizer	$5 \times n$, with n being the number of candidate tokenizers	
Training input	11 pre-trained tokenizers	n candidate tokenizers	n.a.
Training seed	Fixed	Random	
Training result			
Generation input	11 models	5 models for each of the n candidate tokenizers	The best model (based on validation loss) of the final best tokenizer (based on FCD)
Generation seed	Fixed	Fixed	Random
Generation result	11 molecule sets	$5 \times n$ molecule sets	5 molecule sets
Statistical test	n.a.	One or two sample t-test for lower FCD	One sample t-test for lower FCD
Output	The initial best tokenizer	The final (set of) best tokenizer(s)	Comparison with GuacaMol

Table 2.7.: Summary of the evaluation process

Differences between our evaluation and the GuacaMol evaluation

When comparing the evaluation process of the GuacaMol source code with our work, several differences are worth noting [115].

First, while GuacaMol utilizes the Python package `FCD` to calculate the FCD, based on `TensorFlow/Keras`, we use the `PyTorch`-based `fcd-torch` package. To ensure consistent results, we conducted several comparisons between the two packages and found that the largest absolute difference in reported FCD metrics was 0.001. We consider this difference sufficiently small, allowing us to declare equivalent results.

Second, GuacaMol generates new molecules for each metric, whereas in our study, we use the same set of molecules for all the metrics. This approach provides a consistent basis for comparisons across different metrics.

Finally, for the calculation of the FCD, GuacaMol samples 10,000 molecules from the training data. However, in our study, we use the entire training dataset to calculate the activation values, which are then used to determine the FCD. Because of the computational intensity of this process, the activations are calculated once and reused for subsequent calculations.

2.2. Pipeline 2 – Molecules from fine-tuning a pre-trained model

The setup of pipeline 2 bears a significant resemblance to that of pipeline 1, with key differences in the following areas: Unlike pipeline 1, where the models are trained from scratch, pipeline 2 leverages the power of a pre-trained model. By fine-tuning a pre-trained model, we can take advantage of previously learned features and patterns, potentially reducing the training time and improving performance. Pipeline 2 also involves alterations to a pre-trained tokenizer. This modification allows for better alignment with the specific requirements of the task at hand, ensuring that the tokenizer is well-suited to the unique characteristics of the data. Finally, the dimensions of the pre-trained model are adopted, allowing for the seamless integration of the pre-trained components into this task. Figure 2.3 shows a summary of pipeline 2.

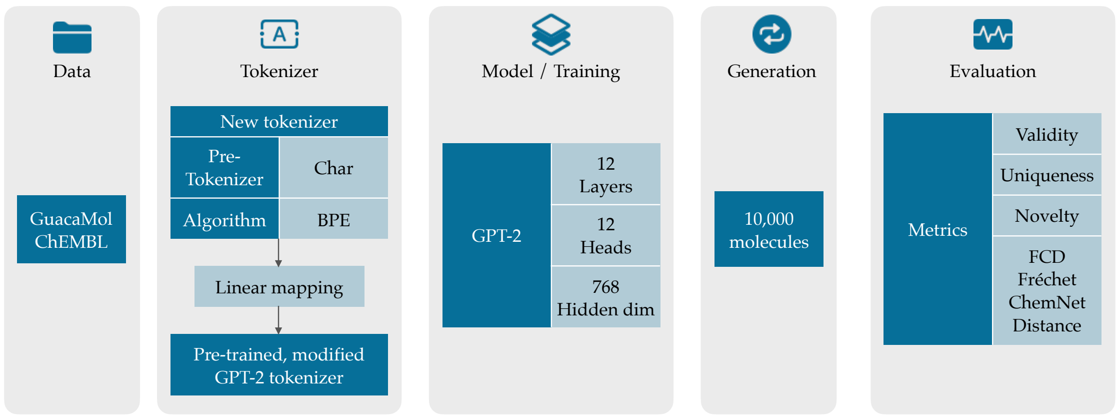

Figure 2.3.: Overview of pipeline 2

2.2.1. Data and tokenization

As described in Section 1.5.5, there are multiple methods for modifying the tokenizer of pre-trained LLMs. In the context of our research, we decided to pair the Char pre-tokenizer with the BPE tokenization algorithm. Our rationale for this selection is twofold.

Initially, our choice aligns with the methodology employed by the developers of the GPT-2 language model, thereby enabling consistency with an established model. Furthermore, this choice facilitates a streamlined implementation process. It is worth noting that this type of adaptation of the tokenizer is not an endorsed use-case within the given implementation framework, with no dedicated API calls to accommodate such modifications.

This necessitated a unique method to implement our approach, which involved manually reading the configuration file of the pre-trained tokenizer, applying the necessary changes, and then saving the adjusted configuration back to a configuration file. In our work, we observe that the file format for the tokenizer configuration may differ based on the chosen tokenization algorithm. This variability reinforces our preference for BPE.

Modifying the pre-trained GPT-2 tokenizer

In our approach to adapt the GPT-2 tokenizer to better suit the nature of our dataset, a sequence of steps is executed. Initially, we build a tokenizer with a Char pre-tokenizer and BPE tokenization model from the training data. We then examine the frequency of the tokens in the resulting vocabulary. Notably, this vocabulary encapsulates SMILES strings with the BOS and EOS tokens. Contrary to the customary GPT-2 " | <endoftext> | " token, in this context, the pre-trained GPT-2 tokenizer perceives the BOS and EOS tokens as conventional entries.

Given the relatively diminutive size of our original vocabulary, which contains 176 tokens, a transition is necessary to make it compatible with the considerably larger GPT-2 vocabulary of 50,527 tokens. This translation is performed by determining the approximate linear mapping "New token index $= -0.035 \times$ Original token index $+ 45,227$" based on the token frequency in our original vocabulary. The Carbon representation, denoted as "C", emerges as the predominant token in our dataset. In the modified GPT-2 vocabulary, a token ID of 256 is assigned. It is crucial to note that we abstain from utilizing token IDs ranging from zero to 255. This decision stems from the inherent special connotations of these IDs within the framework of a byte-level BPE tokenizer.

At the other end of the spectrum, the least recurrent token in our dataset is "b" with a frequency of 1. After considering potential integer rounding issues, nuances of nearly identical token frequencies, and leaving approximately 10% of leeway at the very end of the distribution tail, this token is allocated an ID of 45,230.

An essential follow-up step is the validation of the newly integrated tokens. We must ascertain that the tokens introduced into the GPT-2 tokenizer are not duplicates of existing tokens from the unaltered GPT-2 vocabulary. To ensure consistency and prevent ambiguity during tokenization, duplicate tokens are prefixed to distinguish them.

Following this, a tokenizer configuration file encapsulating the modifications is generated and saved in a file. This configuration file is employed to initialize the ultimate tokenizer utilized during the fine-tuning phase of the pre-trained GPT-2 model.

Table 2.8 shows sample changes to the GPT-2 tokenizer as outlined above.

Task	**GPT-2**				**MolReactGen**	
	Original			Adapted	Trained from scratch	
	Index	Token		Token	Count	Frequency index
Map tokens	256	Ġt	$\longrightarrow$	ˆ (BOS)	1,273,104	1
	257	Ġa	$\longrightarrow$	_ (EOS)	1,273,104	2
	23,384	Ġkidnapped	$\longrightarrow$	C	618,339	3
	41,302	Ġrites	$\longrightarrow$	CC(C)	111,097	65
	45,230	Ġhopped	$\longrightarrow$	b	1	176
Ensure unique token mapping	34	C	$\longrightarrow$	§§§-34	n.a.	
	61	ˆ (BOS)	$\longrightarrow$	§§§-61		
	2601	Cl	$\longrightarrow$	§§§-2601		

Table 2.8.: Sample adaptions of the pre-trained GPT-2 tokenizer. Tokens of the tokenizer trained from scratch are mapped into the pre-trained GPT-2 tokenizer based on their frequency in the GuacaMol training dataset. Tokens that are already present in the pre-trained GPT-2 are replaced to ensure a unique token mapping. The character "Ġ" is a characteristic of GPT-2's Byte-level BPE tokenizer.

2.2.2. Model

In this study, our approach is based on the GPT-2 language model. Since its inception, there have been multiple versions of this model, each expanding its predecessor in terms of its dimensions. This evolutionary progression has led to a nomenclature in which the initial GPT-2 model, once the flagship, is now colloquially termed the "small" GPT-2 model. This model comprises approximately 119 million parameters, which is 41 times larger than the streamlined model of pipeline 1.

There is a remarkable similarity between the model used in our fine-tuning process and that employed in pipeline 1. The discernible differences between the two are in their hidden dimension and sequence length. Although the model for pipeline 1 boasts a hidden dimension of 144, our selected model exhibits a more expansive hidden dimension of 768. It is imperative to note that this decision is not arbitrary; the pre-trained weights of our model are specifically tailored for this 768-dimensional space. Thus, to leverage the potential of these pre-trained weights to their fullest, we align our fine-tuning task with this hidden dimension.

Furthermore, to dissect and understand the contributions and effects of the fine-tuning process and extended hidden dimension, we perform an auxiliary experiment. In this experiment, we train a model with 768 hidden dimension *from scratch*. The rationale behind this is to delineate whether any enhancements observed in pipeline 2 are attributable to the fine-tuning process itself or to the augmented hidden dimension, deliberately isolating potential combinatorial effects.

Table 2.9 lists the parameters that are different from those of pipeline 1.

Parameter	**Value**	**Comment**
Sequence length	$128 \rightarrow 1024$	Resembles the pre-trained model
Hidden dimension	$144 \rightarrow 768$	Resembles the pre-trained model
Number of epochs	$50 \rightarrow 10$	Empirical choice for the fine-tuning task
Batch size	$64 \rightarrow 32$	Reduced because of GPU memory limitations
Learning rate	$0.0025 \rightarrow 0.0005$	Resembles the pre-trained model

Table 2.9.: Pipeline 2 (hyper-) parameter changes

2.3. Pipeline 3 – Reaction templates from scratch

Drawing parallels to pipeline 1, pipeline 3 is adapted to the specific requirements for the generation of reaction templates. These are the differences and adaptations: Reaction templates are represented as SMARTS strings, necessitating the incorporation of a dataset suitable for this purpose. The selection process for the tokenizer results in a different tokenizer being used. In line with the distinct dataset and tokenizer, the model parameters also require slight modifications. Although the core process of the generation process remains consistent with that of pipeline 1, the subsequent steps in evaluating the produced reaction templates show notable changes. These amendments primarily stem from the necessity of gauging the correctness and applicability of the generated reaction templates. Consequently, the evaluation metrics are revamped to provide a comprehensive assessment of their quality. Figure 2.4 shows an overview of pipeline 3.

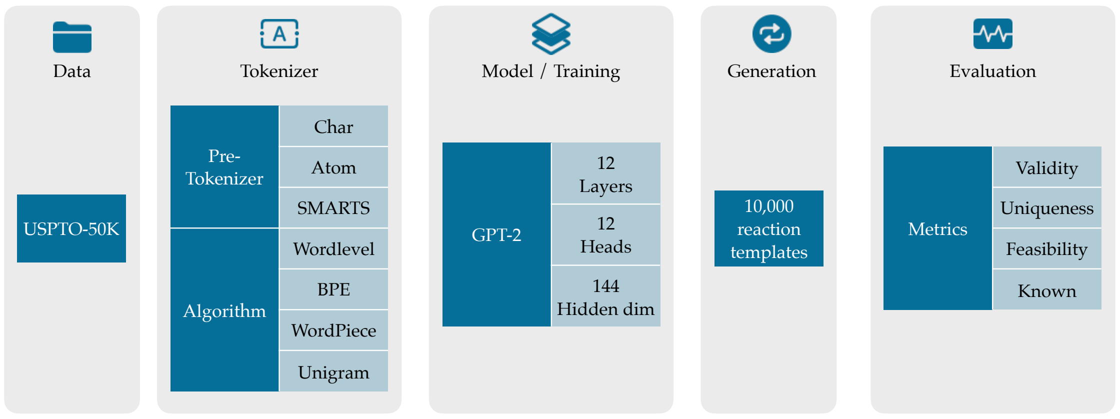

Figure 2.4.: Overview of pipeline 3

2.3.1. Data and tokenization

Our search for the dataset acquisition results in the USPTO-50K dataset. This dataset originates from patent and trademark texts published by the United States Patents and Trademark Office (USPTO) and encompasses 50,016 reactions. These reactions provide detailed information on the reactants, products, and most crucially, the reaction templates. Several authors have pre-processed the patents provided by the USPTO, starting by parsing the actual text and providing it in a machine-readable form to amend it with chemical reaction information. Table 2.10 provides insights into the traces of the data.

The reaction templates can be used in multiple reactions. Although this has advantages for practical applications, it introduces redundancies that must be taken into account during data preparation. We go through a pre-processing phase, in which duplicate reaction template entries are identified and purged. This filtration exercise trims the dataset from its original 50,016 entries to 12,626 unique reaction templates. These templates are segregated to ensure that there is no overlap among the training, validation, and test sets. This pre-processing step results in a much smaller dataset with approximately 25% of its original

Description	Change/Result
Extraction of chemical reactions from US patents published between 1976 and Sep 2016 via text mining	Chemical reactions in machine readable format [116]
Reactants and Products in SMILES format 50,000 random reactions from US patents [117]	Combination of several datasets
Retrosynthesis prediction [118], improving stereochemistry with rdchiral [119], and using code from [120] to cleanse data	Improved data quality
"Retrosim" data amendment with pre-computed reaction templates and reactants [121]	Applicable reaction templates and their reactants

Table 2.10.: Major steps in precedent data pre-processing of the USPTO-50K dataset

size and a decrease in the relative size of the training set. Table 2.11 presents the data split in detail and Figure 2.5 shows the length distribution of the reaction templates.

Data Split	Original Count	After pre-processing	Percentage
Training	40,008	7,877	62%
Validation	5,001	2,413	19%
Test	5,007	2,336	19%
Total	**50,016**	**12,626**	**100%**

Table 2.11.: Number of reaction templates and their splits in the USPTO-50K dataset. The reaction templates are non-unique and non-disjunct across splits. After removing double entries and disjuncting the datasets, we are left with approximately 25% of the original data.

The SMARTS format offers unique challenges and characteristics when juxtaposed with the commonly used SMILES format. First, data on SMARTS are relatively scarce. Further compounding the challenge is the average string length, which is approximately three times longer than that of SMILES strings in the GuacaMol dataset. Moreover, there is a notable presence of strings surpassing 800 characters in length.

SMARTS syntax is more intricate than SMILES syntax. One such complexity is the introduction of "atom mappings." These mappings play a pivotal role in maintaining a consistent correspondence between the reactants and products during template generation. Ensuring the accuracy and coherence of these maps is imperative to achieve reliable results. Empirically, the USPTO-50K dataset is perceived as a challenging format, owing to its characteristics and complexity of the SMARTS syntax.

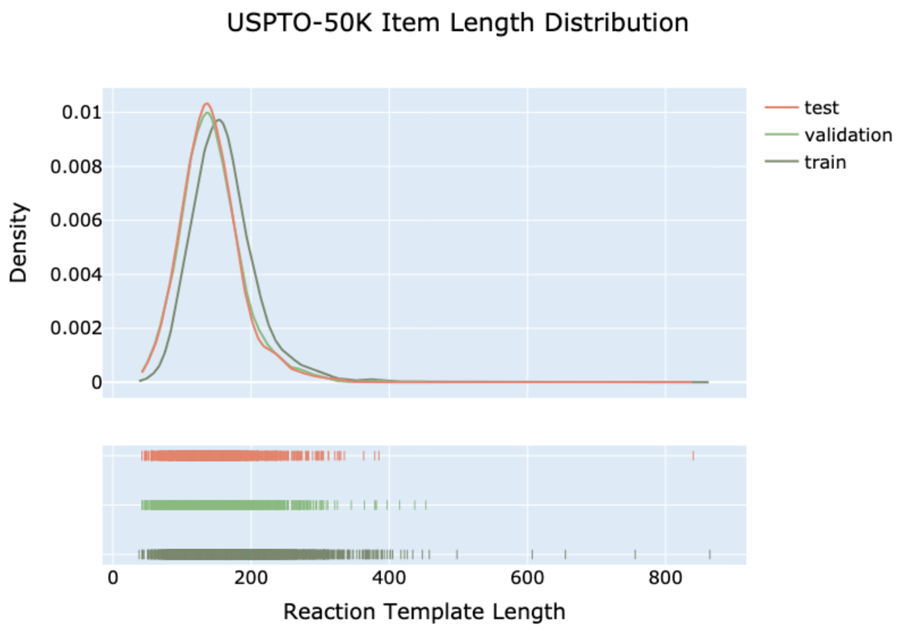

Figure 2.5.: Length distribution of SMARTS strings for each data split. The average lengths are approximately 161 (train set), 144 (validation set) and 143 (test set) characters.

Given the uniqueness of the dataset and the intricacies of the reaction template syntax, it is imperative to reconsider the tokenization methods from pipeline 1. We repeat the selection process discussed in Section 2.1.5 to determine the optimal combination of the pre-tokenizer and tokenization model.

2.3.2. Model

Retaining the architectural choices from pipeline 1, we persist with a hidden dimension of 144. The demands of the dataset lead to specific modifications in terms of the model parameters. Given their potential lengths, it is necessary to change the sequence length of the model. Accommodating a character-wise splitting, a step that inherently elongates the token sequence length, we enhance the model's sequence length to 896. A notable repercussion of tweaking the sequence length is the resultant expansion of the model size, particularly when adopting a transformer architecture. The attention parameters within such architectures exhibit a quadratic growth pattern in relation to sequence length. Consequently, by extending the sequence length, the number of trainable parameters increases. This expansion increases the computational requirements. However, considering the relatively limited number of reaction templates in our dataset, the training runtimes remain comparable to those of pipeline 1.

Table 2.12 shows the parameters that are different from those of pipeline 1:

Parameter	**Value**	**Comment**
Sequence length	128 $\rightarrow$ 896	To fit the maximum SMARTS length
Batch size	64 $\rightarrow$ 128	
Learning rate	0.0025 $\rightarrow$ 0.0005	Reduced to stabilize training

Table 2.12.: Pipeline 3 (hyper-) parameter changes

2.3.3. Generation

At its core, the generation process for producing sequences of tokens closely aligns with the procedure described in pipeline 1. This mechanism offers a blueprint for adapting to the specificities of reaction templates. A significant divergence lies in our task of generating *valid* reaction templates. This leads to the question of what constitutes a valid reaction template. To address this issue, we draw inspiration from the definition of molecular validity. After stripping the generated reaction template from its atom mappings, we obtain one or more reactants and products. We then separate the reactants and products and subject them to individual scrutiny as distinct molecules. A reaction template is considered valid only when each constituent molecule is deemed valid.

It is important to highlight that our established definition of validity focuses primarily on syntactic accuracy. However, a crucial dimension – chemical feasibility – is not encapsulated by this definition. The nuances of evaluating chemical feasibility are discussed in the subsequent section.

2.3.4. Evaluation

Fundamentally, the validity and uniqueness metrics derive their foundation from the measures set forth in pipeline 1. However, the criterion for "valid" has been nuanced, as detailed in the preceding section. Validity seeks to capture the proportion of generated reaction templates that qualify as being valid. A reaction template is valid if, upon exempting atom mapping, its reactants and products can be parsed by `RDKit`.

A reaction template is deemed unique if it has not been previously encountered in the same generation run. Uniqueness is defined as the ratio of unique to valid reaction templates.

Although syntactical correctness is paramount, the ultimate litmus test for a reaction template lies in its feasibility. The feasibility metric focuses on the fraction of unique reaction templates that qualify as being feasible. The feasibility of a reaction template is endorsed if we can trace a product that yields any known reactants in a retrosynthetic step.

More specifically, the following steps are performed to determine the *feasibility* of a generated valid reaction template:

- Search for *similar* reaction templates in the validation and test sets, that is, reaction templates the model has not been trained on.
- In this context, a reaction template is considered to be similar if both the sorted reactants and products (without atom mapping) of the generated reaction template are equal to at least one reaction template in the validation or test sets.
- We then enlist the products of these similar reactions, as found in the validation or test sets.
- If the generated reaction template can be applied to at least one of these products, it is deemed feasible.
- Applying a reaction template to a product refers to simulating a chemical reaction that determines its reactants. The reactants are determined using `RDChiral`, a wrapper for `RDKit` which improves the handling of stereochemistry [122].

Table 2.13 presents an example of the aforementioned processing steps.

Finally, a reaction template is designated as *known* if it finds a match within the validation or test set. This direct string comparison between the generated reaction template and that from the validation or test set, if affirmative, is a robust indicator. Not only does it attest to the template's chemical feasibility, as evidenced by wet lab validations, but it also underscores the model's ability to generate chemically relevant templates.

Table 2.14 presents an overview of the evaluation metrics employed.

Step	#	Sample Result(s)
Generated reaction template	–	`[C:2]-[CH;D3;+0:1](-[C;D1;H3:3])-[NH;D2;+0:4]-[c:5]>>O=[C;H0;D3;+0:1](-[C:2])-[C;D1;H3:3].[NH2;D1;+0:4]-[c:5]`
Similar reaction templates	#1	`[C;D1;H3:2]-[CH;D3;+0:1](-[C;D1;H3:3])-[NH;D2;+0:4]-[c:5]>>O=[C;H0;D3;+0:1](-[C;D1;H3:2])-[C;D1;H3:3].[NH2;D1;+0:4]-[c:5]`
	#2	`[C:2]-[CH;D3;+0:1](-[C;D1;H3:3])-[NH;D2;+0:4]-[c:5]>>O=[C;H0;D3;+0:1](-[C:2])-[C;D1;H3:3].[NH2;D1;+0:4]-[c:5]`
Products corresponding to those reaction templates	#1	`CC(C)Nc1ccc2[nH]ncc2c1`
	#2	`COC(=O)CCC(C)Nc1ccccc1N`
Reactants when applying the generated reaction template to those products	#1	`CC(C)=O.Nc1ccc2[nH]ncc2c1`
	#2	`COC(=O)CCC(C)=O.Nc1ccccc1N`
Result / Interpretation	–	Feasible generated reaction template

Table 2.13.: Sample results to determine the feasibility of the generated reaction template. We begin by searching for similar reaction templates in the USPTO-50K validation and test sets. In this example, we find two similar reaction templates and their corresponding products. We then apply the generated reaction template to these products and determine whether this results in valid reactants. If this is true, the generated reaction template is deemed feasible.

Metric	Formula	Target value	Comment
Validity	$\frac{\text{reaction\_templates}_{\text{valid}}}{\text{reaction\_templates}_{\text{generated}}}$	$\uparrow 1.0$	Valid $\widehat{=}$ reactant(s) *and* product(s) without atom mapping comprise valid molecules
Uniqueness	$\frac{\text{reaction\_templates}_{\text{unique}}}{\text{reaction\_templates}_{\text{valid}}}$	$\uparrow 1.0$	Unique $\widehat{=}$ valid reaction template generated only once
Feasibility	$\frac{\text{reaction\_templates}_{\text{feasible}}}{\text{reaction\_templates}_{\text{unique}}}$	$\uparrow 1.0$	Please refer to the description above for this multi-step process
Known	n.a.	>0	The number of unique reaction templates that are *not* in the training set, but in either validation or test set or both
$\text{Known}_{\text{val}}$			Known reaction templates in the validation set
$\text{Known}_{\text{test}}$			Known reaction templates in the test set

Table 2.14.: Metrics to evaluate the generated reaction templates

3. Results

This section presents the experimental results. We categorize the results based on their respective generation targets, specifically focusing on molecules and reaction templates. At the end of this discussion, we have addressed and provided answers to the primary research questions posed at the outset of this study.

3.1. Molecules

It is important to establish a metric to determine the optimal tokenizer for molecule generation. Typically, this metric is derived from the validation set and serves as a cornerstone for hyperparameter decision-making. However, the diversity in tokenization methodologies introduces discrepancies in vocabulary sizes, which in turn pertain to distinct classes within the cross-entropy loss computation. Given this backdrop, a direct comparison of the validation losses across tokenizers becomes unfeasible.

Instead of relying solely on the validation loss, we embrace the following approach: We employ the entire generation pipeline and identify the FCD as the salient metric to discern the superior tokenization technique, as discussed in Section 2.1.5. To maximize the probability of selecting the best tokenizer within our computational constraints, we concentrate on tokenizers for which we cannot reject the null hypothesis of the same mean when compared to the initial best-performing tokenizer.

Based on our empirical findings, we conclude that a combination of the Char pre-tokenizer paired with the Wordpiece tokenization algorithm, boasting a vocabulary comprising 176 tokens, is the most effective. Table 3.1 shows the details of the tokenizer evaluation results.

For the reasons explained in Section 2.2.1, we use the Char pre-tokenizer with the BPE tokenization algorithm and a vocabulary size of 176 to fine-tune the pre-trained GPT-2 model.

Pre-tokenizer	Tokenization model	Vocabulary size	FCD
Char	Wordlevel	38	0.226
	BPE	44	0.236
		88	0.223 $^{\pm 0.006}$
		176	0.220 $^{\pm 0.004}$
	Wordpiece	88	0.243
		176	0.219 $^{\pm 0.006}$
	Unigram	44	0.230
		88	0.222 $^{\pm 0.006}$
		176	0.232
Atom	Wordlevel	50	0.239
SMARTS	Wordlevel	106	0.241

Table 3.1.: FCD values compared for different tokenizers. Numbers represent single values or, if available, the mean and standard deviation (superscript) across five training and generation runs, reflecting step 1 and 2 of Table 2.7. The darker green background represents the tokenizer with the best mean, the lighter green background denotes tokenizers for which the null hypothesis – same mean – cannot be rejected.

After the tokenizer selection process, the final model is identified based on its validation loss. Using this model, we generate 10,000 molecules in five generation runs and apply the metrics listed in Table 2.6. The comprehensive results of these experiments, juxtaposed with the findings from the GuacaMol paper, are detailed in Table 3.2.

It is important to note the two options for representing the FCD value. The inaugural description from the original study sets the benchmark where a lower FCD value is more desirable, converging ideally to zero. In contrast, the GuacaMol interpretation renders the FCD as $\exp(-0.2 \times \text{FCD})$, where a higher value indicates superior performance, ideally peaking at one. In our analysis, the latter representation is denoted as $\text{FCD}_{\text{GuacaMol}}$. For ease of comprehension and consistency, we reverse-engineer $\text{FCD}_{\text{GuacaMol}}$ values using the term $\text{FCD} = -5 \times \log \text{FCD}_{\text{GuacaMol}}$.

Upon examination of the performance metrics presented in the table, it is evident that the transformer decoder model has a statistically significant superior performance with respect to the FCD, compared to the GuacaMol model. This underscores the potential advantage of the transformer decoder architecture in our application.

Model	**Metrics**				
	Validity	Uniqueness	Novelty	FCD	$\text{FCD}_{\text{GuacaMol}}$
GuacaMol	0.959	1.000	0.994	0.455	0.913
MolGPT	0.981	0.998	1.0	0.488	0.907
MolReactGen *from scratch*	$0.976^{\pm 0.001}$	$0.999^{\pm 0.000}$	$0.935^{\pm 0.002}$	$0.219^{\pm 0.006}$	$0.957^{\pm 0.001}$

Table 3.2.: Molecule generation results, comparing our model with GuacaMol *only*. MolGPT numbers listed for additional context. Numbers represent single values or, if available, the mean and standard deviation (superscript) across five runs with the tokenizer "Char Wordpiece 176". Numbers in green denote the (statistically significant) best metric. The FCD metric is not stated in the GuacaMol and MolGPT papers, we calculate it here as $-5 \times \log \text{FCD}_{\text{GuacaMol}}$
.

Beyond the primary scope of our research question, we identify a generative model that employs a transformer decoder analogous to ours. In this context, the MolGPT study is particularly pertinent [123]. This study utilizes a GPT-inspired model in tandem with the GuacaMol dataset. Notably, it harnesses comparable metrics, allowing for a juxtaposition between the outcomes from the GuacaMol study, the MolGPT model, and our own. As presented in Table 3.2, our findings not only stand their ground but also exhibit very good performance when benchmarked against MolGPT, particularly for the FCD metric. The performance in terms of novelty is lower; however, this can be effectively addressed by generating a larger number of molecules. This underscores the efficacy of the proposed molecular generation approach.

3.1.1. Comparison with the pre-trained model

Table 3.3 compares the performance of the fine-tuned model with that of the model trained from scratch. The fine-tuned model exhibits better performance when juxtaposed with the model trained from scratch. This is manifested not only in terms of FCD, but also in molecular validity. However, it is important to highlight a significant shortcoming: the novelty score of the pre-trained model is considerably lower. A plausible explanation for this observation is the inherent inclination of the pre-trained model to generate molecules that are already present in the training dataset. Although these molecules are inherently valid, given their origin, their lack of novelty is a tradeoff to consider.

To optimize the balance between validity and novelty, we conduct auxiliary experiments by adjusting the temperature during the molecule-generation phase. Temperature is a hyperparameter used during the sampling phase. It adjusts the probability distribution

Model	Metrics			
	Validity	Uniqueness	Novelty	FCD
MolReactGen *from scratch*	0.976 $^{\pm 0.001}$	0.999 $^{\pm 0.000}$	0.935 $^{\pm 0.002}$	0.219 $^{\pm 0.006}$
MolReactGen *fine-tuned*	0.990 $^{\pm 0.001}$	0.999 $^{\pm 0.000}$	0.797 $^{\pm 0.003}$	0.209 $^{\pm 0.006}$
MolReactGen *from scratch* Hidden dim 768	0.966 $^{\pm 0.002}$	0.998 $^{\pm 0.000}$	0.761 $^{\pm 0.006}$	0.227 $^{\pm 0.003}$

Table 3.3.: Molecule generation results, comparing the model trained *from scratch* with the *fine-tuned* model *only*. Numbers represent the mean and standard deviation (superscript) across five runs. We also show the results of a model trained *from scratch* with hidden dimension 768 (same as the fine-tuned model) to indicate that the FCD performance improvement stems from fine-tuning and not from the increased hidden dimension.

over the model outputs by dividing the logits by the temperature. This can influence the diversity of samples generated by the model. For t=1 (default temperature), the logits remain unchanged and the model produces outputs based on the original probabilities. For t > 1, the logits are scaled down, which makes the distribution "flatter." This implies that all possible outputs are more likely. Consequently, the output of the model becomes more diverse. For t < 1, the logits are scaled up, which exaggerates their difference. The highest logits values become even more dominant, and the lower values are suppressed. This makes the model more deterministic and biased towards the most probable outputs.

As hypothesized, an increase in the temperature enhances the novelty of the generated molecules. However, this alteration comes at cost. Even slight increments in temperature, such as at t=1.2, lead to rapid deterioration in the FCD. This delicate interplay between temperature, novelty, and FCD underscores the challenges of selecting the generation hyperparameters and highlights areas for future research.

In our research, we confront a critical question: is the performance of a fine-tuned model primarily due to its pre-training or to its increased hidden dimension? To disentangle these contributing factors, we design an experiment in which we train a model with increased hidden dimension from scratch. Specifically, we employ the parameters of the best-performing model trained from scratch while adjusting its tokenizer and hidden dimension to match those of the pre-trained model. Our empirical results indicate that the performance of this newly trained model is inferior compared to both the original model trained from scratch with its initial hidden dimension and the model that underwent fine-tuning. Consequently, we infer that the improved performance of the fine-tuned model is largely attributable to its pre-training phase rather than to its increased hidden dimension.

3.2. Reaction templates

After an additional tokenizer selection process described in Section 2.1.5, we choose an alternative tokenizer for the reaction templates. This entails integrating the SMARTS pre-tokenizer with a straightforward lookup table (Wordlevel), leading to an expanded vocabulary of 947 tokens. Table 3.4 presents the tokenizer results in detail.

Pre-tokenizer	**Tokenization model**	**Vocabulary size**	**Known**
Char	Wordlevel	47	705.2 $^{\pm 21.2}$
	BPE	47	698.4 $^{\pm 38.4}$
		88	688.0 $^{\pm 35.7}$
		176	618.0
	Wordpiece	94	587.0
		176	557.0
	Unigram	88	621.0
		176	616.0
Atom	Wordlevel	86	730.8 $^{\pm 52.7}$
SMARTS	Wordlevel	947	735.2 $^{\pm 27.0}$

Table 3.4.: "Known" values compared for different tokenizers. Numbers represent single values or, if available, the mean and standard deviation (superscript) across five training and generation runs, reflecting step 1 and 2 of Table 2.7. The darker green background represents the tokenizer with the best mean, the lighter green background denotes tokenizers for which the null hypothesis – same mean – cannot be rejected.

Proceeding with model selection, we generate 10,000 reaction templates. The associated metrics are listed in Table 3.5. We anticipate that validity and uniqueness will register lower scores in comparison to their counterparts in the less-demanding molecule generation tasks.

Model	Metrics					
	Validity	Uniqueness	Feasibility	Known from set		
				either val or test	val	test
MolReactGen *from scratch*	0.804 $^{\pm 0.022}$	0.795 $^{\pm 0.008}$	0.110 $^{\pm 0.004}$	735.2 $^{\pm 27.0}$	489.6 $^{\pm 13.6}$	511.0 $^{\pm 22.6}$

Table 3.5.: Reaction template generation results. Numbers represent the mean and standard deviation (superscript) across five training and generation runs with the tokenizer "SMARTS Wordlevel 947".

3.3. Discussion

Considering the research questions expounded in Section 1.2.3, our analysis encompasses a quartet of metrics for molecule generation: validity, uniqueness, novelty, and FCD. Of these, the FCD stands out as the paramount metric in our evaluation.

Although the tokenizer influences the generation metrics, its effect is limited. The tokenizer decoder model appears to be able to learn the data distribution, irrespective of the tokenization strategy.

Validity serves as a robust yardstick, reflecting the model's proficiency in comprehending the data distribution. Concerning uniqueness and novelty, values falling below one simply suggest a need for the generation of an augmented set of molecules. However, it is important to note that values below approximately 0.9 indicate a subpar model, reflecting its inability to generate diverse or novel molecules.

The fine-tuned model has the best FCD value among the comparisons. At first glance, this would make this model our preferred choice. However, this is not without caution. The fine-tuned model is laden with a parameter count that is an order of magnitude higher, translating to longer inference times and steeper computational costs. When juxtaposed with its diminished novelty score, this factor diminishes its appeal somewhat.

Balancing these considerations, a model trained from scratch emerges as a more harmonized alternative for molecule generation. The results from the model, which incorporates a hidden dimension of 768, suggest that augmenting the hidden dimension may actually be unfavorable.

The model architecture also exhibits the capability of generating more intricate structures, such as reaction templates. This seems significant, given the comparably smaller size of the dataset available for the training set dedicated to reaction templates. Central to our research question is the metric used to assess the number of known reaction templates. We posit that a figure exceeding zero indicates success in this venture. Our results unambiguously affirm this hypothesis, showing that our approach can generate known reaction templates.

Reflecting on the research questions, we want to answer them as follows:

- Our study shows that using the transformer decoder architecture is effective for generating molecules, improving the key metric of the FCD compared to the GuacaMol model. Furthermore, the performance of our model appears to be similar to or better than that of MolGPT, which also utilizes a transformer decoder architecture.
- The effects of the different tokenization techniques are less significant than the model architecture choice, although the tokenizer has a statistically significant performance impact, particularly for the reaction template generation task.
- We are able to fine-tune a pre-trained LLM trained on human text for molecule generation, showing good results. However, this approach requires additional computational resources.
- Despite using only a small amount of training data and the complex syntax of reaction templates, our model can effectively generate reaction templates, demonstrating its capability in handling complex chemical data.

4. Conclusion

4.1. Summary

To advance the frontier of generative models within the chemical domain, we focused our research on the GuacaMol dataset and its associated metrics, and employed them as a benchmark for molecular generation. The methodology we employed involved encoding the SMILES of the molecule using a range of pre-tokenizers and tokenization models. With these encoded data, we embarked on the training of a GPT-2 transformer decoder model from scratch and subsequently juxtaposed its performance against the baseline set by GuacaMol.

In the progression of our research, we developed an approach that facilitates the translation of a molecule's vocabulary into GPT-2's intrinsic vocabulary. Leveraging this, we fine-tuned the pre-trained GPT-2 model. A critical aspect of our study was to draw a comparative analysis between the outcomes derived from training a model from scratch and refining an already-trained model.

To extend our scope further, we utilized the USPTO-50K dataset to train our model on reaction templates. We were able to demonstrate the proficiency of the model in generating chemically feasible reaction templates, including reaction templates of the validation and test sets, which are known to be chemically feasible from wet lab experiments. This not only shows the adaptability of the model, but also underlines its potential for supporting novel chemical insights.

4.2. Future work

Considering the insights gathered in this thesis, several avenues can be explored in the future to further refine and improve the methods and results. The following sections outline potential steps and recommendations for ongoing research in the domain of AI applied to generative models in the chemical field.

4.2.1. Data and tokenization enhancements

Future efforts could involve the integration of additional datasets to ensure the diversity and robustness of the model. Specifically, datasets such as MOSES are pivotal for molecule generation, whereas USPTO-Full can be used for reaction template generation. Given the expansive nature of the USPTO-Full dataset, we postulate that its incorporation might yield a superior model for reaction template generation.

Enhancing the training data using augmentation strategies similar to the SMILES augmentation method proposed by Bjerrum [124] may be beneficial. However, it is noteworthy that the advantages of such augmentation might be more pronounced for datasets of a smaller scale, that is, smaller than GuacaMol.

In addition to the molecular representation methodologies used in this study, alternatives such as SELFIES [125] and DeepSMILES [126] can be evaluated to determine their efficacy in the context of the model.

Once the Hugging Face tokenizers issue [112] is resolved, we can revisit the missing tokenizer combinations, that is, using the tokenization algorithms BPE, WordPiece, and Unigram with the Atom and SMARTS pre-tokenizers.

For the implementation of pipeline 2, considering additional token mapping approaches, such as random mapping, may provide additional insights into the performance of fine-tuning a pre-trained model.

4.2.2. Model innovations

Future research could explore larger or contemporary (transformer decoder) models to ascertain whether they offer a performance boost or novel insight.

There is a compelling case for pre-trained LLMs, specifically on datasets encompassing molecules and reaction templates. Although some models, such as Meta's Galactica [127], have been trained on USPTO data, their current unavailability necessitates a new approach.

4.2.3. Training and generation enhancements

Checkpoint averaging can be employed to develop a more stable and optimized model [128].

Drawing from Liu [129], ensemble methodologies can be integrated into the training process to improve model reliability and accuracy.

The models can potentially benefit from a more exhaustive hyperparameter search space, paving the way for refined tuning and optimization.

Although existing metrics offer significant insight, introducing additional metrics and studying their interplay is crucial. For instance, understanding the sensitivity of optimizing a unique metric, such as enhancing novelty by modulating the temperature during generation, can offer nuanced perspectives.

Transitioning from solely distribution-directed learning, focus can shift towards goal-directed learning. This encompasses the generation of molecules or reaction templates that adhere to specific predefined characteristics.

In conclusion, the roadmap for future work is vast and versatile with promising advancements and innovations in generative models for chemical research.

Bibliography

[1] A. Filipa de Almeida, Rui Moreira, and Tiago Rodrigues. "Synthetic Organic Chemistry Driven by Artificial Intelligence". In: *Nature Reviews Chemistry* 3.10 (10 Oct. 2019), pp. 589–604. ISSN: 2397-3358. DOI: 10.1038/s41570-019-0124-0 (cit. on p. 2).

[2] Zachary J. Baum et al. "Artificial Intelligence in Chemistry: Current Trends and Future Directions". In: *Journal of Chemical Information and Modeling* 61.7 (July 26, 2021), pp. 3197–3212. ISSN: 1549-9596, 1549-960X. DOI: 10.1021/acs.jcim.1c00619 (cit. on p. 2).

[3] Konstantina Athanasopoulou et al. "Artificial Intelligence: The Milestone in Modern Biomedical Research". In: *BioMedInformatics* 2.4 (Dec. 17, 2022), pp. 727–744. ISSN: 2673-7426. DOI: 10.3390/biomedinformatics2040049 (cit. on p. 2).

[4] Pat Langley. "The Computational Support of Scientific Discovery". In: *International Journal of Human-Computer Studies* 53.3 (Sept. 2000), pp. 393–410. ISSN: 10715819. DOI: 10.1006/ijhc.2000.0396 (cit. on p. 2).

[5] Edwin J. De Beer. "The Place of Statistical Methods in Biological and Chemical Experimentation". In: *Annals of the New York Academy of Sciences* 52.6 (Mar. 1950), pp. 791–791. ISSN: 00778923, 17496632. DOI: 10.1111/j.1749-6632.1950.tb53971.x (cit. on p. 2).

[6] Hugh M. Cartwright. "Artificial Neural Networks in Biology and Chemistry—The Evolution of a New Analytical Tool". In: *Artificial Neural Networks*. Ed. by David J. Livingstone. Red. by John M. Walker. Vol. 458. Totowa, NJ: Humana Press, 2008, pp. 1–13. ISBN: 978-1-58829-718-1 978-1-60327-101-1. DOI: 10.1007/978-1-60327-101-1_1 (cit. on p. 2).

[7] AkshatKumar Nigam et al. *Augmenting Genetic Algorithms with Deep Neural Networks for Exploring the Chemical Space*. Jan. 15, 2020. DOI: 10.48550/arXiv.1909.11655. arXiv: 1909.11655 [physics]. preprint (cit. on p. 2).

[8] Henry A. Erlich, David Gelfand, and John J. Sninsky. “Recent Advances in the Polymerase Chain Reaction”. In: *Science* 252.5013 (June 21, 1991), pp. 1643–1651. ISSN: 0036-8075, 1095-9203. DOI: 10.1126/science.2047872 (cit. on p. 2).

[9] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Series in Computer Science. New York: McGraw-Hill, 1997. 414 pp. ISBN: 978-0-07-042807-2 (cit. on pp. 2, 18).

[10] Mukund Deshpande and George Karypis. “Evaluation of Techniques for Classifying Biological Sequences”. In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Ming-Syan Chen, Philip S. Yu, and Bing Liu. Red. by G. Goos, J. Hartmanis, and J. Van Leeuwen. Vol. 2336. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 417–431. ISBN: 978-3-540-43704-8 978-3-540-47887-4. DOI: 10.1007/3-540-47887-6_41 (cit. on p. 2).

[11] Marten H. Hofker, Jingyuan Fu, and Cisca Wijmenga. “The Genome Revolution and Its Role in Understanding Complex Diseases”. In: *Biochimica et Biophysica Acta (BBA) - Molecular Basis of Disease* 1842.10 (Oct. 2014), pp. 1889–1895. ISSN: 09254439. DOI: 10.1016/j.bbadis.2014.05.002 (cit. on p. 2).

[12] Maxwell W. Libbrecht and William Stafford Noble. “Machine Learning Applications in Genetics and Genomics”. In: *Nature Reviews Genetics* 16.6 (June 2015), pp. 321–332. ISSN: 1471-0056, 1471-0064. DOI: 10.1038/nrg3920 (cit. on p. 3).

[13] Garrett B. Goh, Nathan O. Hodas, and Abhinav Vishnu. “Deep Learning for Computational Chemistry”. In: *Journal of Computational Chemistry* 38.16 (June 15, 2017), pp. 1291–1307. ISSN: 01928651. DOI: 10.1002/jcc.24764 (cit. on p. 3).

[14] Daniele Ravi et al. “Deep Learning for Health Informatics”. In: *IEEE Journal of Biomedical and Health Informatics* 21.1 (Jan. 2017), pp. 4–21. ISSN: 2168-2194, 2168-2208. DOI: 10.1109/JBHI.2016.2636665 (cit. on p. 3).

[15] John Jumper et al. “Applying and Improving AlphaFold at CASP14”. In: *Proteins: Structure, Function, and Bioinformatics* 89.12 (Dec. 2021), pp. 1711–1721. ISSN: 0887-3585, 1097-0134. DOI: 10.1002/prot.26257 (cit. on p. 3).

[16] Mihaly Varadi et al. “AlphaFold Protein Structure Database: Massively Expanding the Structural Coverage of Protein-Sequence Space with High-Accuracy Models”. In: *Nucleic Acids Research* 50.D1 (Jan. 7, 2022), pp. D439–D444. ISSN: 0305-1048, 1362-4962. DOI: 10.1093/nar/gkab1061 (cit. on pp. 3, 4).

[17] Tuvshinbayar Chantsalnyam et al. “ncRDeep: Non-coding RNA Classification with Convolutional Neural Network”. In: *Computational Biology and Chemistry* 88 (Oct. 2020), p. 107364. ISSN: 14769271. DOI: 10.1016/j.compbiolchem.2020.107364 (cit. on p. 3).

[18] Jonathan Frazer et al. "Disease Variant Prediction with Deep Generative Models of Evolutionary Data". In: *Nature* 599.7883 (Nov. 4, 2021), pp. 91–95. ISSN: 0028-0836, 1476-4687. DOI: `10.1038/s41586-021-04043-8` (cit. on p. 3).

[19] Jun Cheng et al. "Accurate Proteome-Wide Missense Variant Effect Prediction with AlphaMissense". In: *Science* (Sept. 19, 2023). DOI: `10.1126/science.adg7492` (cit. on p. 3).

[20] Geert Litjens et al. "A Survey on Deep Learning in Medical Image Analysis". In: *Medical Image Analysis* 42 (Dec. 2017), pp. 60–88. ISSN: 13618415. DOI: `10.1016/j.media.2017.07.005` (cit. on p. 3).

[21] James Zou et al. "A Primer on Deep Learning in Genomics". In: *Nature Genetics* 51.1 (Jan. 2019), pp. 12–18. ISSN: 1061-4036, 1546-1718. DOI: `10.1038/s41588-018-0295-5` (cit. on p. 3).

[22] Jose Roberto Ayala Solares et al. "Deep Learning for Electronic Health Records: A Comparative Review of Multiple Deep Neural Architectures". In: *Journal of Biomedical Informatics* 101 (Jan. 2020), p. 103337. ISSN: 15320464. DOI: `10.1016/j.jbi.2019.103337` (cit. on p. 3).

[23] Hongming Chen et al. "The Rise of Deep Learning in Drug Discovery". In: *Drug Discovery Today* 23.6 (June 2018), pp. 1241–1250. ISSN: 13596446. DOI: `10.1016/j.drudis.2018.01.039` (cit. on p. 3).

[24] Hakime Öztürk, Arzucan Özgür, and Elif Ozkirimli. "DeepDTA: Deep Drug–Target Binding Affinity Prediction". In: *Bioinformatics* 34.17 (Sept. 1, 2018), pp. i821–i829. ISSN: 1367-4803, 1367-4811. DOI: `10.1093/bioinformatics/bty593` (cit. on p. 3).

[25] Alvin Rajkomar et al. "Scalable and Accurate Deep Learning with Electronic Health Records". In: *npj Digital Medicine* 1.1 (May 8, 2018), p. 18. ISSN: 2398-6352. DOI: `10.1038/s41746-018-0029-1` (cit. on p. 3).

[26] Qiang Yang et al. "Federated Machine Learning: Concept and Applications". In: *ACM Transactions on Intelligent Systems and Technology* 10.2 (Mar. 31, 2019), pp. 1–19. ISSN: 2157-6904, 2157-6912. DOI: `10.1145/3298981` (cit. on p. 3).

[27] Daniel Merk et al. "*De Novo* Design of Bioactive Small Molecules by Artificial Intelligence". In: *Molecular Informatics* 37.1-2 (Jan. 2018), p. 1700153. ISSN: 1868-1743, 1868-1751. DOI: `10.1002/minf.201700153` (cit. on p. 3).

[28] Michael Ridley. "Explainable Artificial Intelligence (XAI)". In: *Information Technology and Libraries* 41.2 (June 15, 2022). ISSN: 2163-5226, 0730-9295. DOI: `10.6017/ital.v41i2.14683` (cit. on p. 3).

[29] Marwin H. S. Segler, Mike Preuss, and Mark P. Waller. “Planning Chemical Syntheses with Deep Neural Networks and Symbolic AI”. In: *Nature* 555.7698 (Mar. 2018), pp. 604–610. ISSN: 0028-0836, 1476-4687. DOI: `10.1038/nature25978` (cit. on p. 3).

[30] Kai Guo et al. “Artificial Intelligence and Machine Learning in Design of Mechanical Materials”. In: *Materials Horizons* 8.4 (2021), pp. 1153–1172. ISSN: 2051-6347, 2051-6355. DOI: `10.1039/D0MH01451F` (cit. on p. 3).

[31] Philippe Schwaller et al. “Machine Intelligence for Chemical Reaction Space”. In: *WIREs Computational Molecular Science* 12.5 (Sept. 2022), e1604. ISSN: 1759-0876, 1759-0884. DOI: `10.1002/wcms.1604` (cit. on p. 3).

[32] Davide Rigoni, Nicolò Navarin, and Alessandro Sperduti. *A Systematic Assessment of Deep Learning Models for Molecule Generation*. Aug. 20, 2020. DOI: `10.48550/arXiv.2008.09168`. arXiv: `2008.09168 [cs, q-bio]`. preprint (cit. on p. 4).

[33] David Weininger. “SMILES, a Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules”. In: *Journal of Chemical Information and Modeling* 28.1 (Feb. 1, 1988), pp. 31–36. ISSN: 1549-9596. DOI: `10.1021/ci00057a005` (cit. on pp. 4, 14).

[34] Xiandong Zou et al. *Will More Expressive Graph Neural Networks Do Better on Generative Tasks?* Aug. 23, 2023. arXiv: `2308.11978 [cs, q-bio, stat]`. URL: `http://arxiv.org/abs/2308.11978` (visited on 09/11/2023). preprint (cit. on p. 4).

[35] Philipp Renz et al. “On Failure Modes in Molecule Generation and Optimization”. In: *Drug Discovery Today: Technologies*. Artificial Intelligence 32–33 (Dec. 1, 2019), pp. 55–63. ISSN: 1740-6749. DOI: `10.1016/j.ddtec.2020.09.003` (cit. on pp. 5, 37, 38).

[36] Nathan Brown et al. “GuacaMol: Benchmarking Models for de Novo Molecular Design”. In: *Journal of Chemical Information and Modeling* 59.3 (Mar. 25, 2019), pp. 1096–1108. ISSN: 1549-9596, 1549-960X. DOI: `10.1021/acs.jcim.8b00839` (cit. on pp. 5, 8, 37).

[37] Pieter P. Plehiers et al. “Automated Reaction Database and Reaction Network Analysis: Extraction of Reaction Templates Using Cheminformatics”. In: *Journal of Cheminformatics* 10.1 (Dec. 2018), p. 11. ISSN: 1758-2946. DOI: `10.1186/s13321-018-0269-8` (cit. on pp. 6, 17).

[38] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Computation* 9.8 (Nov. 1, 1997), pp. 1735–1780. ISSN: 0899-7667, 1530-888X. DOI: `10.1162/neco.1997.9.8.1735` (cit. on p. 8).

[39] Ashish Vaswani et al. “Attention Is All You Need”. Dec. 5, 2017. arXiv: 1706.03762 [cs]. URL: http://arxiv.org/abs/1706.03762 (visited on 03/18/2022) (cit. on pp. 8, 28, 29).

[40] Alec Radford et al. “Language Models Are Unsupervised Multitask Learners”. In: (2019), p. 24 (cit. on pp. 8, 12, 28).

[41] Chuanqi Tan et al. “A Survey on Deep Transfer Learning”. In: *Artificial Neural Networks and Machine Learning – ICANN 2018*. Ed. by Věra Kůrková et al. Vol. 11141. Cham: Springer International Publishing, 2018, pp. 270–279. ISBN: 978-3-030-01423-0 978-3-030-01424-7. DOI: 10.1007/978-3-030-01424-7_27 (cit. on p. 9).

[42] Teague Sterling and John J. Irwin. “ZINC 15 – Ligand Discovery for Everyone”. In: *Journal of Chemical Information and Modeling* 55.11 (Nov. 23, 2015), pp. 2324–2337. ISSN: 1549-9596. DOI: 10.1021/acs.jcim.5b00559 (cit. on p. 10).

[43] Daniil Polykovskiy et al. “Molecular Sets (MOSES): A Benchmarking Platform for Molecular Generation Models”. In: *Frontiers in Pharmacology* 11 (2020). ISSN: 1663-9812. URL: https://www.frontiersin.org/articles/10.3389/fphar.2020.565644 (visited on 07/14/2023) (cit. on pp. 10, 37).

[44] David Mendez et al. “ChEMBL: Towards Direct Deposition of Bioassay Data”. In: *Nucleic Acids Research* 47.D1 (Jan. 8, 2019), pp. D930–D940. ISSN: 1362-4962. DOI: 10.1093/nar/gky1075. pmid: 30398643 (cit. on p. 10).

[45] European Bioinformatics Institute. *EMBL-EBI Homepage*. URL: https://www.ebi.ac.uk/ (visited on 09/12/2023) (cit. on p. 10).

[46] Andreas Steffen et al. “Comparison of Molecular Fingerprint Methods on the Basis of Biological Profile Data”. In: *Journal of Chemical Information and Modeling* 49.2 (Feb. 23, 2009), pp. 338–347. ISSN: 1549-9596, 1549-960X. DOI: 10.1021/ci800326z (cit. on p. 11).

[47] Esben Jannik Bjerrum and Richard Threlfall. “Molecular Generation with Recurrent Neural Networks (RNNs)”. May 17, 2017. DOI: 10.48550/arXiv.1705.04612. arXiv: 1705.04612 [cs, q-bio] (cit. on p. 11).

[48] Thomas Blaschke et al. “Application of Generative Autoencoder in *De Novo* Molecular Design”. In: *Molecular Informatics* 37.1-2 (Jan. 2018), p. 1700123. ISSN: 1868-1743, 1868-1751. DOI: 10.1002/minf.201700123 (cit. on p. 11).

[49] Rafael Gómez-Bombarelli et al. "Automatic Chemical Design Using a Data-Driven Continuous Representation of Molecules". In: *ACS Central Science* 4.2 (Feb. 28, 2018), pp. 268–276. ISSN: 2374-7943. DOI: `10.1021/acscentsci.7b00572` (cit. on p. 11).

[50] Benjamin Sanchez-Lengeling et al. *Optimizing Distributions over Molecular Space. An Objective-Reinforced Generative Adversarial Network for Inverse-design Chemistry (ORGANIC)*. preprint. Chemistry, Aug. 17, 2017. DOI: `10.26434/chemrxiv.5309668.v3` (cit. on p. 11).

[51] Rocío Mercado et al. "Graph Networks for Molecular Design". In: *Machine Learning: Science and Technology* 2.2 (June 1, 2021), p. 025023. ISSN: 2632-2153. DOI: `10.1088/2632-2153/abcf91` (cit. on p. 11).

[52] Marcus Olivecrona et al. "Molecular De-Novo Design through Deep Reinforcement Learning". In: *Journal of Cheminformatics* 9.1 (Dec. 2017), p. 48. ISSN: 1758-2946. DOI: `10.1186/s13321-017-0235-x` (cit. on p. 11).

[53] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". May 24, 2019. DOI: `10.48550/arXiv.1810.04805`. arXiv: `1810.04805 [cs]` (cit. on pp. 12, 28, 30).

[54] Philipp Seidl et al. *Enhancing Activity Prediction Models in Drug Discovery with the Ability to Understand Human Language*. Mar. 6, 2023. DOI: `10.48550/arXiv.2303.03363`. arXiv: `2303.03363 [cs, q-bio, stat]`. preprint (cit. on p. 12).

[55] Ling Xue et al. "Design and Evaluation of a Molecular Fingerprint Involving the Transformation of Property Descriptor Values into a Binary Classification Scheme". In: *Journal of Chemical Information and Computer Sciences* 43.4 (July 1, 2003), pp. 1151–1157. ISSN: 0095-2338. DOI: `10.1021/ci030285+` (cit. on p. 14).

[56] Daniel de Marchi and Amarjit Budhiraja. "Augmenting Molecular Images with Vector Representations as a Featurization Technique for Drug Classification". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. May 2020, pp. 956–960. DOI: `10.1109/ICASSP40776.2020.9053425`. arXiv: `2008.03646 [cs, q-bio]` (cit. on p. 14).

[57] Joseph L. Durant et al. "Reoptimization of MDL Keys for Use in Drug Discovery". In: *Journal of Chemical Information and Computer Sciences* 42.6 (Nov. 2002), pp. 1273–1280. ISSN: 0095-2338. DOI: `10.1021/ci010132r` (cit. on p. 14).

[58] David Rogers and Mathew Hahn. "Extended-Connectivity Fingerprints". In: *Journal of Chemical Information and Modeling* 50.5 (May 24, 2010), pp. 742–754. ISSN: 1549-9596, 1549-960X. DOI: `10.1021/ci100050t` (cit. on p. 14).

[59] Pedro O. Pinheiro et al. “3D Molecule Generation by Denoising Voxel Grids”. Version 1. In: (2023). DOI: 10.48550/ARXIV.2306.07473 (cit. on p. 14).

[60] Greg Landrum et al. *Rdkit/Rdkit: 2022_09_5 (Q3 2022) Release*. Version Release 2022_09_5. Zenodo, Feb. 23, 2023. DOI: 10.5281/ZENODO.7671152 (cit. on pp. 15, 46, 85).

[61] Yinjie Jiang et al. “Artificial Intelligence for Retrosynthesis Prediction”. In: *Engineering* (Aug. 2022), S2095809922005665. ISSN: 20958099. DOI: 10.1016/j.eng.2022.04.021 (cit. on p. 17).

[62] Daylight Chemical Information Systems, Inc. *Daylight Theory: SMARTS - A Language for Describing Molecular Patterns*. SMARTS - A Language for Describing Molecular Patterns. URL: https://www.daylight.com/dayhtml/doc/theory/theory.smarts.html (visited on 04/11/2022) (cit. on p. 18).

[63] Philipp Seidl et al. “Improving Few- and Zero-Shot Reaction Template Prediction Using Modern Hopfield Networks”. In: *Journal of Chemical Information and Modeling* 62.9 (May 9, 2022), pp. 2111–2120. ISSN: 1549-9596, 1549-960X. DOI: 10.1021/acs.jcim.1c01065 (cit. on p. 18).

[64] Inc. Daylight Chemical Information Systems. *Daylight>SMARTS Examples*. URL: https://www.daylight.com/dayhtml_tutorials/languages/smarts/smarts_examples.html (visited on 10/10/2023) (cit. on p. 18).

[65] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep Learning”. In: *Nature* 521.7553 (May 28, 2015), pp. 436–444. ISSN: 1476-4687. DOI: 10.1038/nature14539. pmid: 26017442 (cit. on p. 18).

[66] “(1986) D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propogation," [i]Parallel Distributed Processing: Explorations in the Microstructures of Cognition[/i], Vol. I, D. E. Rumelhart and J. L. McClelland (Eds.) Cambridge, MA: MIT Press, Pp. 318-362.(1986) David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, "Learning Representations by Back-Propogating Errors," [i]Nature[/i] 323:533-536.” In: *Neurocomputing, Volume 1*. Ed. by James A. Anderson and Edward Rosenfeld. The MIT Press, Apr. 7, 1988, pp. 673–700. ISBN: 978-0-262-26713-7. DOI: 10.7551/mitpress/4943.003.0042 (cit. on p. 18).

[67] Warren S. McCulloch and Walter Pitts. “A Logical Calculus of the Ideas Immanent in Nervous Activity”. In: *The bulletin of mathematical biophysics* 5.4 (Dec. 1, 1943), pp. 115–133. ISSN: 1522-9602. DOI: 10.1007/BF02478259 (cit. on p. 18).

[68] Abien Fred Agarap. "Deep Learning Using Rectified Linear Units (ReLU)". In: *ArXiv* (Mar. 22, 2018). URL: https://www.semanticscholar.org/paper/Deep-Learning-using-Rectified-Linear-Units-(ReLU)-Agarap/b79e5e4622a95417deec313cd543617b19611bea (visited on 09/18/2023) (cit. on p. 19).

[69] Günter Klambauer et al. "Self-Normalizing Neural Networks". Sept. 7, 2017. arXiv: 1706.02515 [cs, stat]. URL: http://arxiv.org/abs/1706.02515 (visited on 03/23/2022) (cit. on p. 19).

[70] Peter Kottas. *How Does Deep Learning Work and How Is It Different from Normal Neural Networks Applied with SVM? How Does One Go about Starting to Under...* Quora. 2017. URL: https://www.quora.com/How-does-deep-learning-work-and-how-is-it-different-from-normal-neural-networks-applied-with-SVM-How-does-one-go-about-starting-to-understand-them-papers-blogs-articles/answer/Peter-Kottas (visited on 10/23/2023) (cit. on p. 19).

[71] Amanpreet Singh, Narina Thakur, and Aakanksha Sharma. "A Review of Supervised Machine Learning Algorithms". In: *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*. 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom). Mar. 2016, pp. 1310–1315. URL: https://ieeexplore.ieee.org/document/7724478 (visited on 10/17/2023) (cit. on p. 19).

[72] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Second edition. A Wiley-Interscience Publication. New York Chichester Weinheim Brisbane Singapore Toronto: John Wiley & Sons, Inc, 2001. 654 pp. ISBN: 978-0-471-05669-0 978-0-471-70350-1 (cit. on p. 20).

[73] Leonard Kaufman and Peter J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. 1st ed. Wiley Series in Probability and Statistics. Wiley, Mar. 8, 1990. ISBN: 978-0-471-87876-6 978-0-470-31680-1. DOI: 10.1002/9780470316801 (cit. on p. 20).

[74] R.S. Sutton and A.G. Barto. "Reinforcement Learning: An Introduction". In: *IEEE Transactions on Neural Networks* 9.5 (Sept. 1998), pp. 1054–1054. ISSN: 1045-9227. DOI: 10.1109/TNN.1998.712192 (cit. on p. 20).

[75] Qi Wang et al. "A Comprehensive Survey of Loss Functions in Machine Learning". In: *Annals of Data Science* 9.2 (Apr. 2022), pp. 187–212. ISSN: 2198-5804, 2198-5812. DOI: 10.1007/s40745-020-00253-5 (cit. on p. 23).

[76] Alec Radford et al. “Learning Transferable Visual Models From Natural Language Supervision”. In: *Proceedings of the 38th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, July 1, 2021, pp. 8748–8763. URL: `https://proceedings.mlr.press/v139/radford21a.html` (visited on 12/04/2023) (cit. on p. 24).

[77] Andreas Fürst et al. “CLOOB: Modern Hopfield Networks with InfoLOOB Outperform CLIP”. In: *Advances in Neural Information Processing Systems* 35 (Dec. 6, 2022), pp. 20450–20468. URL: `https://proceedings.neurips.cc/paper_files/paper/2022/hash/8078e76f913e31b8467e85b4c0f0d22b-Abstract-Conference.html` (visited on 12/04/2023) (cit. on p. 24).

[78] Ana Sanchez-Fernandez et al. “CLOOME: Contrastive Learning Unlocks Bioimaging Databases for Queries with Chemical Structures”. In: *Nature Communications* 14.1 (1 Nov. 13, 2023), p. 7339. ISSN: 2041-1723. DOI: `10.1038/s41467-023-42328-w` (cit. on p. 24).

[79] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 29, 2017. DOI: `10.48550/arXiv.1412.6980`. arXiv: `1412.6980 [cs]`. preprint (cit. on p. 24).

[80] Paul J. Werbos. “Applications of Advances in Nonlinear Sensitivity Analysis”. In: *System Modeling and Optimization*. Ed. by R. F. Drenick and F. Kozin. Vol. 38. Berlin/Heidelberg: Springer-Verlag, 1982, pp. 762–770. ISBN: 978-3-540-11691-2. DOI: `10.1007/BFb0006203` (cit. on p. 24).

[81] J.-F. Cardoso. “Infomax and Maximum Likelihood for Blind Source Separation”. In: *IEEE Signal Processing Letters* 4.4 (Apr. 1997), pp. 112–114. ISSN: 1070-9908, 1558-2361. DOI: `10.1109/97.566704` (cit. on p. 26).

[82] James P. Shaver. “What Statistical Significance Testing Is, and What It Is Not”. In: *The Journal of Experimental Education* 61.4 (July 1993), pp. 293–316. ISSN: 0022-0973, 1940-0683. DOI: `10.1080/00220973.1993.10806592` (cit. on p. 26).

[83] Xavier Amatriain. *Transformer Models: An Introduction and Catalog*. Feb. 16, 2023. DOI: `10.48550/arXiv.2302.07730`. arXiv: `2302.07730 [cs]`. preprint (cit. on p. 27).

[84] Tom B. Brown et al. *Language Models Are Few-Shot Learners*. July 22, 2020. DOI: `10.48550/arXiv.2005.14165`. arXiv: `2005.14165 [cs]`. preprint (cit. on p. 28).

[85] Alexey Dosovitskiy et al. *An Image Is Worth 16x16 Words: Transformers for Image Recognition at Scale*. June 3, 2021. DOI: `10.48550/arXiv.2010.11929`. arXiv: `2010.11929 [cs]`. preprint (cit. on p. 28).

[86] Aditya Ramesh et al. *Hierarchical Text-Conditional Image Generation with CLIP Latents*. Apr. 12, 2022. DOI: 10.48550/arXiv.2204.06125. arXiv: 2204.06125 [cs]. preprint (cit. on p. 28).

[87] OpenAI. *GPT-4 Technical Report*. Mar. 27, 2023. DOI: 10.48550/arXiv.2303.08774. arXiv: 2303.08774 [cs]. preprint (cit. on p. 28).

[88] OpenAI. *Introducing ChatGPT*. Nov. 30, 2022. URL: https://openai.com/blog/chatgpt (visited on 09/20/2023) (cit. on p. 28).

[89] Ashish Vaswani et al. *Attention Is All You Need*. July 23, 2023. DOI: 10.48550/arXiv.1706.03762. arXiv: 1706.03762 [cs]. preprint (cit. on p. 29).

[90] Austin Huang et al. *The Annotated Transformer*. 2022. URL: http://nlp.seas.harvard.edu/annotated-transformer/ (visited on 10/11/2023) (cit. on p. 29).

[91] Jonathan J. Webster and Chunyu Kit. "Tokenization as the Initial Phase in NLP". In: *Proceedings of the 14th Conference on Computational Linguistics -*. The 14th Conference. Vol. 4. Nantes, France: Association for Computational Linguistics, 1992, p. 1106. DOI: 10.3115/992424.992434 (cit. on p. 30).

[92] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Softcover reprint of the original 1st edition 2006 (corrected at 8th printing 2009). Information Science and Statistics. New York, NY: Springer New York, 2016. 738 pp. ISBN: 978-1-4939-3843-8 (cit. on p. 31).

[93] A Neelima and Shashi Mehrotra. "A Comprehensive Review on Word Embedding Techniques". In: *2023 International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS)*. 2023 International Conference on Intelligent Systems for Communication, IoT and Security (ICISCoIS). Coimbatore, India: IEEE, Feb. 9, 2023, pp. 538–543. ISBN: 9798350335835. DOI: 10.1109/ICISCoIS56541.2023.10100347 (cit. on p. 31).

[94] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units". June 10, 2016. DOI: 10.48550/arXiv.1508.07909. arXiv: 1508.07909 [cs] (cit. on p. 32).

[95] Mike Schuster and Kaisuke Nakajima. "Japanese and Korean Voice Search". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. ICASSP 2012 - 2012 IEEE International Conference on Acoustics, Speech and Signal Processing. Kyoto, Japan: IEEE, Mar. 2012, pp. 5149–5152. ISBN: 978-1-4673-0046-9 978-1-4673-0045-2 978-1-4673-0044-5. DOI: 10.1109/ICASSP.2012.6289079 (cit. on p. 32).

[96] Taku Kudo. *Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates*. Apr. 29, 2018. DOI: 10.48550/arXiv.1804.10959. arXiv: 1804.10959 [cs]. preprint (cit. on p. 32).

[97] Xinhao Li and Denis Fourches. "SMILES Pair Encoding: A Data-Driven Substructure Tokenization Algorithm for Deep Learning". In: (May 21, 2020). DOI: 10.26434/chemrxiv.12339368.v1 (cit. on p. 33).

[98] Alex Graves. *Generating Sequences With Recurrent Neural Networks*. June 5, 2014. DOI: 10.48550/arXiv.1308.0850. arXiv: 1308.0850 [cs]. preprint (cit. on p. 34).

[99] *OpenAI*. URL: https://openai.com/ (visited on 09/29/2023) (cit. on p. 34).

[100] Ashwin K. Vijayakumar et al. *Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models*. Oct. 22, 2018. arXiv: 1610.02424 [cs]. URL: http://arxiv.org/abs/1610.02424 (visited on 02/06/2023). preprint (cit. on pp. 35, 36).

[101] Angela Fan, Mike Lewis, and Yann Dauphin. *Hierarchical Neural Story Generation*. May 13, 2018. arXiv: 1805.04833 [cs]. URL: http://arxiv.org/abs/1805.04833 (visited on 09/21/2023). preprint (cit. on p. 35).

[102] Uri Shaham and Omer Levy. "What Do You Get When You Cross Beam Search with Nucleus Sampling?" In: *Proceedings of the Third Workshop on Insights from Negative Results in NLP*. Proceedings of the Third Workshop on Insights from Negative Results in NLP. Dublin, Ireland: Association for Computational Linguistics, 2022, pp. 38–45. DOI: 10.18653/v1/2022.insights-1.5 (cit. on p. 35).

[103] Yixuan Su et al. *A Contrastive Framework for Neural Text Generation*. Sept. 26, 2022. DOI: 10.48550/arXiv.2202.06417. arXiv: 2202.06417 [cs]. preprint (cit. on p. 36).

[104] Kishore Papineni et al. "BLEU: A Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics - ACL '02*. The 40th Annual Meeting. Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, p. 311. DOI: 10.3115/1073083.1073135 (cit. on p. 37).

[105] Kristina Preuer et al. "Fréchet ChemNet Distance: A Metric for Generative Models for Molecules in Drug Discovery". In: *Journal of Chemical Information and Modeling* 58.9 (Sept. 24, 2018), pp. 1736–1741. ISSN: 1549-9596. DOI: 10.1021/acs.jcim.8b00234 (cit. on p. 37).

[106] Martin Heusel et al. *GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium*. Jan. 12, 2018. DOI: 10.48550/arXiv.1706.08500. arXiv: 1706.08500 [cs, stat]. preprint (cit. on p. 37).

[107] Maurice Fréchet. “Sur La Distance de Deux Lois de Probabilité”. In: *Annales de l'ISUP* VI.3 (1957), pp. 183–198. URL: `https://hal.science/hal-04093677` (cit. on p. 37).

[108] L. N. Wasserstein. “Markov processes over denumerable products of spaces describing large systems of automata”. In: *Probl. Inform. Transmission* 5.3 (1969), pp. 47–52. ISSN: 0555-2923 (cit. on p. 37).

[109] Andreas Mayr et al. “Large-Scale Comparison of Machine Learning Methods for Drug Target Prediction on ChEMBL”. In: *Chemical Science* 9.24 (2018), pp. 5441–5451. ISSN: 2041-6520, 2041-6539. DOI: `10.1039/C8SC00148K` (cit. on p. 37).

[110] Wenhao Gao and Connor W. Coley. “The Synthesizability of Molecules Proposed by Generative Models”. In: *Journal of Chemical Information and Modeling* 60.12 (Dec. 28, 2020), pp. 5714–5723. ISSN: 1549-9596. DOI: `10.1021/acs.jcim.0c00174` (cit. on p. 39).

[111] Philippe Schwaller et al. “Molecular Transformer: A Model for Uncertainty Calibrated Chemical Reaction Prediction”. In: *ACS Central Science* 5.9 (Sept. 25, 2019), pp. 1572–1583. ISSN: 2374-7943. DOI: `10.1021/acscentsci.9b00576` (cit. on p. 42).

[112] Holzgruber. *'BPE' Tokenization Model Does Not Respect Custom 'RegEx' via 'Split' Pre-Tokenizer · Issue #1369 · Huggingface/Tokenizers*. GitHub. Oct. 18, 2023. URL: `https://github.com/huggingface/tokenizers/issues/1369` (visited on 10/30/2023) (cit. on pp. 43, 69).

[113] NVIDIA. *Tips for Optimizing GPU Performance Using Tensor Cores*. NVIDIA Technical Blog. URL: `https://developer.nvidia.com/blog/optimizing-gpu-performance-tensor-cores/` (visited on 10/03/2023) (cit. on p. 44).

[114] Nitish Shirish Keskar et al. *CTRL: A Conditional Transformer Language Model for Controllable Generation*. Sept. 20, 2019. arXiv: `1909.05858 [cs]`. URL: `http://arxiv.org/abs/1909.05858` (visited on 08/07/2023). preprint (cit. on p. 46).

[115] BenevolentAI. *GuacaMol*. BenevolentAI, Apr. 26, 2022. URL: `https://github.com/BenevolentAI/guacamol` (visited on 04/27/2022) (cit. on p. 50).

[116] Daniel Mark Lowe. “Extraction of Chemical Structures and Reactions from the Literature”. Thesis. University of Cambridge, Oct. 9, 2012. DOI: `10.17863/CAM.16293` (cit. on p. 56).

[117] Nadine Schneider, Nikolaus Stiefl, and Gregory A. Landrum. "What's What: The (Nearly) Definitive Guide to Reaction Role Assignment". In: *Journal of Chemical Information and Modeling* 56.12 (Dec. 27, 2016), pp. 2336–2346. ISSN: 1549-9596. DOI: 10.1021/acs.jcim.6b00564 (cit. on p. 56).

[118] Connor Coley. *Retrosim*. Feb. 23, 2022. URL: https://github.com/connorcoley/retrosim (visited on 03/13/2022) (cit. on p. 56).

[119] Connor W. Coley, William H. Green, and Klavs F. Jensen. "RDChiral: An RDKit Wrapper for Handling Stereochemistry in Retrosynthetic Template Extraction and Application". In: *Journal of Chemical Information and Modeling* 59.6 (June 24, 2019), pp. 2529–2537. ISSN: 1549-9596. DOI: 10.1021/acs.jcim.9b00286 (cit. on p. 56).

[120] Bowen Liu et al. "Retrosynthetic Reaction Prediction Using Neural Sequence-to-Sequence Models". In: *ACS Central Science* 3.10 (Oct. 25, 2017), pp. 1103–1113. ISSN: 2374-7943. DOI: 10.1021/acscentsci.7b00303 (cit. on p. 56).

[121] Philipp Seidl et al. "Modern Hopfield Networks for Few- and Zero-Shot Reaction Template Prediction". June 15, 2021. arXiv: 2104.03279 [cs, q-bio, stat]. URL: http://arxiv.org/abs/2104.03279 (visited on 11/02/2021) (cit. on p. 56).

[122] Connor Coley. *Rdchiral*. Aug. 26, 2023. URL: https://github.com/connorcoley/rdchiral (visited on 10/04/2023) (cit. on p. 59).

[123] Viraj Bagal et al. "MolGPT: Molecular Generation Using a Transformer-Decoder Model". In: *Journal of Chemical Information and Modeling* 62.9 (May 9, 2022), pp. 2064–2076. ISSN: 1549-9596. DOI: 10.1021/acs.jcim.1c00600 (cit. on p. 63).

[124] Esben Jannik Bjerrum. "SMILES Enumeration as Data Augmentation for Neural Network Modeling of Molecules". May 17, 2017. DOI: 10.48550/arXiv.1703.07076. arXiv: 1703.07076 [cs] (cit. on p. 69).

[125] Mario Krenn et al. "SELFIES and the Future of Molecular String Representations". Mar. 31, 2022. arXiv: 2204.00056 [physics]. URL: http://arxiv.org/abs/2204.00056 (visited on 04/08/2022) (cit. on p. 69).

[126] Noel O'Boyle and Andrew Dalke. "DeepSMILES: An Adaptation of SMILES for Use in Machine-Learning of Chemical Structures". In: (Sept. 27, 2018). DOI: 10.26434/chemrxiv.7097960.v1 (cit. on p. 69).

[127] Ross Taylor et al. *Galactica: A Large Language Model for Science*. Nov. 16, 2022. DOI: 10.48550/arXiv.2211.09085. arXiv: 2211.09085 [cs, stat]. preprint (cit. on p. 69).

[128] Martin Popel and Ondřej Bojar. “Training Tips for the Transformer Model”. In: *The Prague Bulletin of Mathematical Linguistics* 110.1 (Apr. 1, 2018), pp. 43–70. ISSN: 1804-0462. DOI: 10.2478/pralin-2018-0002. arXiv: 1804.00247 [cs] (cit. on p. 70).

[129] Yuchen Liu et al. “A Comparable Study on Model Averaging, Ensembling and Reranking in NMT”. In: *Natural Language Processing and Chinese Computing*. Ed. by Min Zhang et al. Vol. 11109. Cham: Springer International Publishing, 2018, pp. 299–308. ISBN: 978-3-319-99500-7 978-3-319-99501-4. DOI: 10.1007/978-3-319-99501-4_26 (cit. on p. 70).

[130] Python Software Foundation. *Python.Org*. Python.org. Oct. 2, 2023. URL: https://www.python.org/ (visited on 10/10/2023) (cit. on p. 85).

[131] Pytorch. *PyTorch*. URL: https://www.pytorch.org (visited on 10/10/2023) (cit. on p. 85).

[132] Thomas Wolf et al. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. July 13, 2020. DOI: 10.48550/arXiv.1910.03771. arXiv: 1910.03771 [cs]. preprint (cit. on p. 85).

[133] Lukas Biewald. *Experiment Tracking with Weights and Biases*. 2020. URL: https://www.wandb.com/ (cit. on p. 85).

A. Appendix

A.1. Implementation details

The development environment comprises macOS v13/14 as the foundation for coding and prototyping. For the actual execution of the experiments, we deploy a local Ubuntu v22.04 LTS installation with an NVIDIA 3080Ti GPU and 12 GB RAM. This decision is motivated by its widespread use in scientific computing and its compatibility with the requisite libraries.

`Python` v3.9 serves as the main programming language [130]. In terms of deep learning frameworks and natural language processing tools, `PyTorch` v1.13 [131] is used as the primary deep learning library. Additionally, Hugging Face `tokenizers` v0.13 and `transformers` v4.27 are incorporated into the pipeline [132]. These libraries offer pre-trained models and tokenization methods that are pivotal in handling and processing textual data. We employ `RDKit` v2022.09.05 for advanced chemical informatics processing [60]. `RDKit` offers a comprehensive suite of functionalities, including molecule and reaction representation, making it indispensable for cheminformatics tasks. Finally, `Weights and Biases` v0.15 is employed for experiment logging. This is beneficial for tracking and visualizing experiments in real-time [133]. After updating to more recent releases (`PyTorch` v2.1, `tokenizers` v0.14, and `transformers` v4.35), we repeat the experiments to ensure consistency of the results and compatibility with a more current AI software ecosystem.

The source code for our experiments is accessible to facilitate reproducibility and further research. This can be found on GitHub at `https://github.com/hogru/MolReactGen`. Additionally, for practical applications and ease of use, we provide pre-trained models for molecule and reaction template generation. These models are hosted on Hugging Face and can be accessed at the following URLs: for molecules, `https://huggingface.co/hogru/MolReactGen-GuacaMol-Molecules`, and for reaction templates, `https://huggingface.co/hogru/MolReactGen-USPTO50K-Reaction-Templates`.

A.1.1. RegEx patterns used for pre-tokenization

For the Atom and SMARTS pre-tokenizers, we use the following regular expression patterns. The patterns are almost identical, differing only in the handling of square brackets, as expressed at the start of the pattern sequence.

- Atom pre-tokenizer

```
\[|\]|A[cglmrstu]?|B[aeihkr]?|C[adeflmorsu]?|D[bsy]?|E[rsu]?| ⌋
 ↪  F[emr]?|G[ade]?|H[efgos]?|I[nr]?|K[r]?|L[aru]?|M[dgnot]?| ⌋
 ↪  N[abdeiop]?|Os?|P[abdmortu]?|R[aefghnu]?|S[bcdegimnr]?|T[ ⌋
 ↪  abcehilm]?|U|V|W|Xe|Yb?|Z[nr]?|as|b|c|n|o|p|se?|\(|\)|@@? ⌋
 ↪  |\+\d+|\++|-\d+|-+|\.|-|=|#\d{1}|\$\(?|:[0-9]*|~|@|\/\??| ⌋
 ↪  \\\??|\%[0-9]{2}|[0-9]|\*|>>?|D\d{1}|H\d{1}|h\d{1}|R\d{1} ⌋
 ↪  |r\d{1}|v\d{1}|X\d{1}|x\d{1}|#\d{1}|!|&|,|;
```

- SMARTS pre-tokenizer

```
\[[^\]]+]|A[cglmrstu]?|B[aeihkr]?|C[adeflmorsu]?|D[bsy]?|E[rs ⌋
 ↪  u]?|F[emr]?|G[ade]?|H[efgos]?|I[nr]?|K[r]?|L[aru]?|M[dgno ⌋
 ↪  t]?|N[abdeiop]?|Os?|P[abdmortu]?|R[aefghnu]?|S[bcdegimnr] ⌋
 ↪  ?|T[abcehilm]?|U|V|W|Xe|Yb?|Z[nr]?|as|b|c|n|o|p|se?|\(|\) ⌋
 ↪  |@@?|\+\d+|\++|-\d+|-+|\.|-|=|#\d{1}|\$\(?|:[0-9]*|~|@|\/ ⌋
 ↪  \??|\\\??|\%[0-9]{2}|[0-9]|\*|>>?|D\d{1}|H\d{1}|h\d{1}|R\ ⌋
 ↪  d{1}|r\d{1}|v\d{1}|X\d{1}|x\d{1}|#\d{1}|!|&|,|;
```