Decoding Molecular Language Model with Beam Search

Stephen Obonyo[1][0000−0002−6878−7802], Nicolas Jouandeau[1][0000−0001−6902−4324], and Dickson Owuor[3][0000−0002−0968−5742]

[1] LIASD, Paris 8 University, Paris, France
[2] SCES, Strathmore University, Nairobi, Kenya
{sobonyo,n}@up8.edu

Abstract. Keywords: molecular design · generative modelling · language models · tree search · beam search · decoding language model
Molecular design is one of the challenging problems in chemical synthesis due to the large search space of valid molecules. Existing methods are based on two key encoding approaches: molecular graph and textual SMILES. Molecular graph encoding methods are expressive and chemically-aware as they include atoms, bonds and other molecular properties. SMILES-based approaches on the other hand do not consider any chemical information and treat the molecules as a sequence of characters. Current generative molecular graphs and SMILES-based models learn the distribution of the input and then sample from the learned distribution to generate new molecules. SMILES-based methods are prone to generating invalid molecules and are not chemically aware. Despite this however, the success of Large Language Models (LLMs) in Natural Language Processing (NLP) has led to the development of strong LLM methods which are competitive with the state-of-the-art molecular graph-based methods. This paper shows how a fragment-based SMILES LLM can be trained and sampled effectively with beam search to improve the generated molecules' validity, novelty and uniqueness. We evaluate the model on two standard molecular design datasets: ZINC and PCBA. We show that our model can generate accurate molecules with high validity, novelty and uniqueness while recording results comparable to or better than the state-of-the-art molecular graph-based methods.

1 Introduction

Molecular design involves the development of new chemical compounds with desired target properties to solve problems in several areas of application such as drug design, synthesis of new target materials and materials science [12, 52]. The problem is challenging as there is a large discrete search space which includes up to 10^{60} valid synthesizable molecules [52].

The twofold increase in computing power in the last two decades [46] and the development of novel Artificial Intelligence (AI) methods [37, 33, 36] has inspired the development of new computer-aided molecular design methods. Specifically, the advancements in Deep Learning (DL) has led to a new suite of a strong set of neural network architectures which can learn to *approximate any function* [26]. A simple neural network architecture, the Perceptron, can be used to approximate any function by mimicking the human brain's neurons. To solve more complex problems, the Perceptron can be stacked to form a Multi-Layer Perceptron (MLP) which can learn to solve harder problems. However, training deep neural networks (MLPs) can be challenging due to the vanishing gradient problem [5] which leads to performance degradation as the network depth increases. This limitation is addressed by the development of novel training methods such as batch and layer normalization [27, 2], residual connections [22], and dropout [59].

Several DL methods have been proposed to solve the molecular design problem. Such include deep generative models [18, 24], deep reinforcement learning [45, 57], machine translation [4, 43, 61], and DL on graphs [32, 21]. Deep generative models are a suite of DL algorithms that can learn the distribution of the data then generate new data from the learned distribution. Different data formats can be used to train deep generative models e.g. text, audio, images, and graphs (i.e. molecular graphs). Generative models have emerged as a strong alternative to the classical Machine Learning (ML) molecular design methods.

Molecules can be represented in different ways i.e. molecular graphs, SMILES strings, fragment molecular graphs or SMILES [47]. A molecular graph is a graph representation of a molecule where the nodes represent atoms and the edges represent bonds between the atoms. SMILES (Simplified Molecular Input Line Entry System) [65] on the other hand is a textual representation of a molecule consistent with the molecular graph where the atoms, bonds and other molecular properties are encoded as a string of characters. A molecular fragment is a substructure which is a part of a molecule i.e. functional groups, rings, atoms and other molecular properties, and can be combined to form a valid molecule. The fragment-based design approach has been reported to be effective in several molecular design tasks as the design task is decomposed into smaller subtasks which can be solved independently. For instance, in Fragment-based Drug design (FBDD), a target molecule is designed according to the fragments that are known to bind to the target protein [15, 34]. The approach is a chemically sound and effective complement to the traditional drug design methods such as large-scale throughput screening and virtual screening [47].

Researchers have proposed several SMILES-based generative models which learn the distribution of the molecular text and sample from the learned distribution to generate new molecules [20, 10, 35, 17]. These works are based on neural network architectures such as Recurrent Neural Networks (RNNs) [25], Gated Recurrent Units (GRUs) [9], Variational Autoencoders (VAEs) [31], sequence-to-sequence models [4, 43, 61], and, recently, the transformers [62]. Despite the earlier success of RNNs, GRUs, and VAEs in molecular design, these models are limited by the long-term range of dependencies and are hard to train due to the vanishing gradient problem [5]. This limitation is also inherited by the sequence-to-sequence models which use RNNs or GRUs as the encoder and decoder. Transformers on the other hand have been shown to be effective in learning long-term dependencies by using a self-attention mechanism [62]. The success of transformer-based models on previously hard NLP tasks such as machine translation, text generation, summarization and question-answering [62, 14, 48] has inspired the application to other areas such as molecular design [3].

SMILES-based molecular have two key limitations: (i) they are prone to generating invalid molecules as the next token is sampled autoregressively, and (ii) they are not chemically-aware i.e. the strings are generated without considering the chemical properties of the atoms and bonds. With the recent advancements in DL on graphs [8, 21], graph-based molecular design models are increasingly becoming popular. These models convert the SMILES into a molecular graph including atom, bond and other molecular properties then train deep generative models on the graph itself [32, 30, 19, 58, 68, 38]. While molecular generative models are chemically aware and more accurate, they require the specification of the node ordering which can be challenging for large molecules [12]. Additionally, optimization of molecular generation from the continuous space has been reported as a daunting task by several studies such as [31, 12].

In this paper, we present a fragment-based transformer language model for molecular design. Contrary to the SMILES-based models are trained on the molecular text, our model is trained on the molecular fragments recording results which are comparable to or better than the state-of-the-art graph-based models. Training the model on the fragments has two key advantages: (1) the model can generate fragments which can be used in fragment-based problems, (2) the model is more expressive as the fragments can be combined to form a valid molecule sequentially.

Contributions: The main contributions of this paper are as follows:

(I) **Language model.**: We present a fragment-based transformer language model for molecular design. The model is trained on the fragments of the molecules and then decoded using beam search to generate new molecules.

(II) **Decoding optimization with search.** We show how beam search can be used to improve the validity, uniqueness and novelty of the generated molecules.

2 Related Work

Fragment-based sequence-to-sequence model by [47] is the closest to our work. The authors proposed a fragment-based sequence-to-sequence GRU model for molecular design where the SMILES are broken down into fragments using Breaking of Retrosynthetically Interesting Chemical Substructures (BRICS) [13] method. The fragments are embedded and then passed to the GRU encoder-decoder model. The model is trained on the shifted (similar to the machine translation sequence to sequence) values of the fragments as the target. The main difference between our work and [47] is that we use the transformer model instead of the GRU model. Also, we fine-tune our model generation process using beam search to generate more valid, unique and diverse molecules. GRUs inherit the long-term memory dependencies problem which is addressed by the transformer model through the self-attention mechanism. Also, training GRUs can be challenging due to the vanishing gradient when the size of the input is long.

In [20], the authors proposed a SMILES objective-reinforced generative adversarial network (ORGAN) model for designing molecules with desired properties. The model is inspired by [69] to learn molecular sequence output while optimizing molecular objectives with REINFORCE [66] algorithm. REINFORCE is a Reinforcement Learning (RL) algorithm that learns to optimize policy by learning the actions taken by the agent and updating the policy according to the gradient with respect to the reward obtained. While the model proved effective in generating molecules with desired properties, it inherited the limitations of the SMILES-based models such as generating invalid molecules as well as convergence issues associated with the REINFORCE algorithm and the adversarial training [1].

In [12], the authors proposed a molecular generative adversarial network (GAN) [18] which reinforces the molecular property objective using Deep Deterministic Policy Gradient [39] (DDPG) RL algorithm. GAN learns the distribution of the molecular space using a generator and a discriminator. The generator is trained to generate molecules that are indistinguishable from the real molecules while the discriminator is trained to distinguish between the real and generated molecules. The DDPG algorithm is used to optimize the molecular property objective towards a non-differentiable reward objective such as drug-likeness. While the model recorded impressive and competitive results, it also inherited the limitations of the GANs and DDPG algorithms. GANs suffer from the mode collapse problem and are also hard to train while DDPG is known to be unstable and requires careful tuning of the hyperparameters [53]. [54] also proposed RL-based based on Monte Carlo Tree Search (MCTS) and deep neural networks as value and policy. Despite the mentioned limitations of the RL methods, they are effective in reinforcing the molecular property objectives in end-to-end molecular design pipelines.

CharacterVAE model proposed by [17] is a SMILES-based language model which uses VAEs to learn the distribution of the molecular text. VAEs are a class of generative models which learn the distribution of the data by encoding the input into a latent space and then reconstructing the input from sampling the latent space. In the work, a recurrent encoder-decoder VAE is used to learn the distribution of the molecules and then sampled to generate new molecules. GrammarVAE [35] also uses VAEs to model the distribution of the molecular molecules, however, the generation process is constrained by a set of syntactic rules to ensure that the generated molecules are valid. A similar design method was also employed by SDVAE [10].

Several studies have also approached the molecular design through molecular graph encoding. Common design approaches in this category include: (i) encode-decoder models [51, 19, 11], (ii) likelihood-based models which generate arbitrary graphs sequentially [30, 38, 68] and (iii) direct graph generation models such as

GraphVAE [58], GraphGAN [68], Junction Tree VAE [29] and NeVAE [49], and, (iv) link prediction models [44, 63, 7].

Recently, the transformer model has recorded state-of-the-art performance in several NLP tasks such as machine translation, text generation and summarization and question answering [62, 14, 48]. The model's success has inspired the development of strong SMILES-based molecular design models. For instance, [3] proposed a transformer-based molecular design model called MolGPT. The model is trained on the SMILES tokens are used to generate new molecules. The model is shown to be effective in generating valid molecules and is at par with the state-of-the-art graph-based and SMILES-based models. While this work, is also closely related to our work, we focus on the fragment-based language model and how beam search can be used to improve the validity, uniqueness and diversity of the generated molecules. In addition, in the work, the model is trained on the molecular scaffolds.

3 Methodology

3.1 Fragmenting molecules

Molecular Fragments are small-weight compounds generally composed of less than 20 atoms. This site has several advantages: (i) they can be manipulated easily compared to the larger molecules, (ii) it is easier to work with and characterize the fragments, (iii) in drug design, working with fragments is convenient as the fragments which can bind to the target protein can be easily identified and experimented with. In this work, we used the BRICS method [13] to fragment the molecules. Figure 1 shows a sample molecule broken according to the BRICS method while algorithm 1 shows the pseudocode of the fragmentation process. Given CC(NC(=O)C1CCCN1S(C)(=O)=O)c1ccc2c(c1)OCO2 as the molecule, three fragments are generated: *C(C)c1ccc2c(c1)OCO2, *NC(*)=O, *C1CCCN1* and *S(C)(=O)=O.

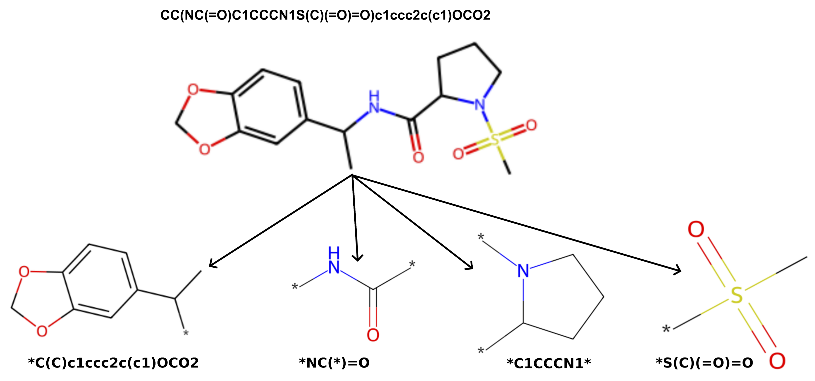


Fig. 1: Fragmentation of with BRICS

BRICS breaks the molecule at strategic bonds such that the resulting fragments are chemically sound. After a single step of fragmentation, a dummy atom is added to the cleavage site to mark the position where the two fragments were separated (or can be joined). The dummy atom is denoted by an asterisk $*$. BRICS bond-breaking rules generally preserve the chemical properties such as the rings, functional groups and the stereochemistry of the molecule.

Our implementation of the BRICS method is shown in algorithm 1. The iterative algorithm first identifies the first bond in the molecule and then splits the molecule at the bond. While the molecule contains fragments, the algorithm continues to split the molecule at the first available bond. In each split iteration, the molecule is updated to the remaining molecule after the split. If there are no fragments in the molecule, the algorithm returns the fragments. In the implementation, we add the leftmost fragment to the list of fragments which is consistent with the SMILES notation.

Algorithm 1 Iterative fragmentation of molecules

```
Require: Molecule $\mathcal{M}$, Fragments $\mathcal{O} \leftarrow \{\ \}$
 1: procedure Fragment($\mathcal{M}$, $\mathcal{O}$)
 2:     Initialize start bond $b$
 3:     while fragment in $\mathcal{M}$ do
 4:         $b \leftarrow$ SplitBond($\mathcal{M}$)
 5:         if $b$ is not empty then
 6:             $o, \mathcal{M}' \leftarrow$ SplitAtBond($\mathcal{M}, b$)
 7:             $\mathcal{O} \leftarrow \mathcal{O} \cup o$
 8:             $\mathcal{M} \leftarrow \mathcal{M}'$
 9:         else
10:             return $\mathcal{O}$
11:         end if
12:     end while
13: end procedure
```

3.2 Training Fragment-based Language Model

In this section, we describe the training of the fragment-based language model. The model is trained on the fragments of the molecules. For instance, given a molecule $\mathcal{M}$ = CC(NC(=O)C1CCCN1S(C)(=O)=O)c1ccc2c(c1)OCO2, the molecule is broken into fragments (tokens) *C(C)c1ccc2c(c1)OCO2, *NC(*)=O, *C1CCCN1* and *S(C)(=O)=O. Here we used an autoregressive generative model, GPT-2 [48], as the language model. An autoregressive generative model is a generative model that models the distribution of the data by predicting the next token given the previous tokens. Autoregressive models are also referred to as decoder models as they use a single decoder network to predict the next token. Formally, given tokens $t_1, t_2, \ldots, t_{n-1}$, the model predicts the next token t_n by maximizing the likelihood of the token t_n given the previous tokens $t_1, t_2, \ldots, t_{n-1}$. This can be expressed as shown in equation 1.

$$\max_{\theta} \sum_{i=1}^{n} \log P(t_i | t_1, t_2, \ldots, t_{i-1}; \theta) \tag{1}$$

We trained a smaller version of the GPT-2 model [48] called DistilGPT2 (82M parameter) on the molecular fragments. DistilGPT2 is an English language model which is trained through a knowledge distillation process [23] from the 124M parameter GPT-2 model. The knowledge distillation process involves training a smaller model which is faster and lighter yet retains the performance of the larger model. The DistilGPT2 knowledge distillation process follows the work of [50] which showed how the knowledge in a large masked language model can be transferred to a smaller model while retaining the performance. A masked language model is another type of language modelling which is trained to predict the masked tokens in the input sequence. These models are also referred to as encoder models as they use a single encoder to predict the masked tokens [14].

We trained a new molecular tokenizer using Byte Pair Encoding (BPE) [55] algorithm. Accordingly, we modified the DistilGPT2 embedding layer to be consistent without vocabulary size i.e. in the original architecture, the embedding layer dimensions are 50257 X 768 while the head dimensions are 768 X 50257. We changed these dimensions to be consistent with our vocabulary size, 913, i.e. the embedding layer dimensions are 911 X 768 while the head dimensions are 768 X 913 for training on the ZINC dataset. For training on the PCBA dataset, with a vocabulary size of 1208, the embedding layer and head dimensions were changed to 1208 X 768 and 768 X 1208 respectively. The model was trained starting with the pre-trained weights of the DistilGPT2 model. Training the model from scratch did not yield good results compared to the pre-trained model version. The model was trained using AdamW optimizer [42] with a learning rate of 1e-4 and, batch size of 32 and 50 epochs. We used cosine annealing learning rate scheduler [41] to periodically scale the learning rate such that it starts at the maximum value and then decays to the minimum value over the training period. The model was trained on three NVIDIA v100 (32 GB memory) for two and a half days.

Importantly, we trained a new tokenizer for the model. The model was trained by updating the GPT-2 tokenizer used by the DistilGPT2 model. In the end, the ZINC dataset included 913 tokens and 1208 tokens for the PCBA dataset. The tokenizer and the model were trained using Huggingface's tokenizers and Transformers libraries [67].

3.3 Greedy and beam search

Generally, LLMs generate sequences by sampling through a greedy approach i.e. the model selects the token with the highest probability at each time step. While the greedy search approach is simple and efficient, it tends to generate suboptimal solutions as it does not consider the future implications of the current token. We discuss these two search methods in the following section.

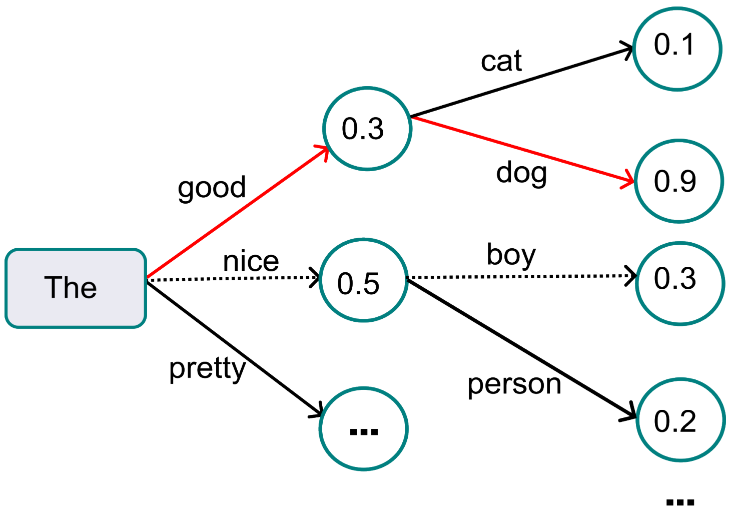


Fig. 2: Beams and greedy search example.

I) **Greedy search**: Algorithms based on greedy selection prioritize the optimal choice at each step, without taking into account potential future implications. In LLMs decoding process this can be expressed as $t_t = \arg\max_t P(t|t_{1:t-1})$ where t_t is the token at time step t, $P(t|t_{1:t-1})$ is the probability of the token t given the previous tokens $t_{1:t-1}$. The fundamental stages of greedy search comprise (i) defining an objective function for a given problem, (ii) selecting the best option that optimizes the objective function at each step of the search process, and (iii) repeating this process until a solution is obtained. However, the greedy search methodology neglects alternatives that could potentially yield superior solutions, resulting in suboptimal outcomes. To address the limitations of greedy search, several approaches have been proposed, including ϵ-greedy and beam search. The ϵ-greedy algorithm operates similarly to the greedy algorithm but introduces a probability ϵ for exploring alternative options and a probability of $1-\epsilon$ for exploiting the best option. The ϵ-greedy approach is extensively utilized in RL, where the agent aims to learn an optimal policy through trial and error to maximize the expected cumulative future reward. To achieve this, the agent explores the environment by selecting random actions with a probability ϵ and exploits the best action with a probability of $1-\epsilon$.

II) **Beam search**: While the greedy search approach tends to disregard solutions with higher probabilities that are obscured by lower probability solutions, the beam search method mitigates this issue. Beam search retains the most probable solutions at each time step and subsequently employs these to determine the one with the overall highest probability. The quantity of probable solutions (paths) retained is regulated by the number of beams parameter in the beam search algorithm. Although a higher number of beams may result in improved solutions, it is also more likely to increase computational complexity. An illustrative example (in Figure 2) is the generation of words using a LLM e.g. GPT-2 [48], where the goal is to produce the subsequent sequence of words given a specific input sequence. In the given example, the number of beams is set to 2, and at time step 1, the most probable solution is "the nice" with a probability of 0.5, while the second most probable solution is "the good" with a probability of 0.3. During the second time step, the most probable solution (sequence) is "the good dog" with an overall probability of $0.3 \times 0.9 = 0.27$. Although the "the nice" sequence appeared to be the best choice according to the greedy search in step 1, it is outperformed by the "the good dog" sequence in step 2. The beam search algorithm has been successfully employed in various NLP tasks, such as machine translation and causal language modelling [60, 56], to enhance the sequence generation process.

The beam search algorithm can be applied to optimize numerous bioinformatics problems, including RNA/DNA sequence modelling, gene prediction, protein structure prediction, molecular design, and drug discovery. For instance, in molecular design, the objective is to produce a sequence of molecular fragments that optimize the desired properties, such as solubility or drug-likeness. The beam search can be adapted to focus on the given molecular property objective instead of the likelihood of the sequence (probability) to generate the sequence that optimizes the molecular objective. A pseudocode of the beam search algorithm using a language model to generate the token in every time step is presented in algorithm 2 where the language model can produce a character or a word at each time step based on the problem context, e.g., in molecular design, the model can generate a sequence of molecular fragments or generate a base in RNA/DNA sequence problems.

Algorithm 2 Decoding LLM with Beam Search

```
Require: Model $M_\theta$, input $x$, vocabulary $V$, sequence length $T$, beam width $B$
1: Initialize output $\mathcal{O}_0 \leftarrow \{(\ )\}$
2: for 1 ... T do
3:     $C_t \leftarrow \{o \bigcup M_\theta(x)\}$ where $M_\theta(x) \in V$ and $o \in \mathcal{O}_{t-1}$    ▷ Sample next tokens
4:     $\mathcal{O}_t \leftarrow \arg\max_{S \subseteq \mathcal{C}_t, |S|=B} \log M_\theta(S|x)$    ▷ Best $B$ sequences
5: end for
6: return $\arg\max_{S \in \mathcal{O}_T} \log M_\theta(S|x)$
```

4 Experiments

In our experiments, we train and evaluate the model on the ZINC [28] and PCBA [64] datasets. The ZINC dataset is composed of approximately 250,000 drug-like molecules while the PCBA dataset is composed of 440,000 bioactive molecules. The dataset was preprocessed following the work of [47] where the duplicate molecules were first removed, then followed by a fragmentation process according to algorithm 1. After this, the resulting molecules with less than 2 fragments were filtered out. In the end, the ZINC dataset used in our experiments contained 227K samples and 383K for the PCBA dataset. The model was trained on the ZINC datasets.

Table 1: Summary of the dataset statistics used in the experiments.

	ZINC	PCBA
Original dataset samples (before preprocessing)	249,455	437,929
Filtered samples (more than 2 fragments)	227,940	383,788
Average fragments	2.3	2.4
Vocabulary size	913	1208
Average atoms	23.60	27.30
Average bonds	27.30	29.63
Average rings	2.80	3.92

During the decoding process, we experimented with a beam width of 2, 4 and 6 and compared the performance of the beam search variations with greedy search as well baseline models such as ChemVAE [17] GrammarVAE [35], SD-VAE [10], CGVAE [40], NeVAE [49], MolGAN [12] and LFM [47]. The baseline models include both the SMILES-based (ChemVAE, GrammarVAE, SDVAE) and graph-based models (CGVAE, NeVAE, MolGAN). The evaluation metrics used in the experiments include validity, uniqueness and novelty. We discuss each of these metrics in the following section.

(i) **Validity**: Out of the generated molecules, validity corresponds to the fraction of molecules which map to a valid SMILES string. A valid SMILES string is a string that can be converted to a molecular graph.
(ii) **Uniqueness**: this is the ratio of the unique molecules generated to the total generated molecules.
(iii) **Novelty**: the fraction of valid molecules to the total generated molecules and not present in the training set.

5 Results

We present the model results in Table 2. All the results of the baseline models were obtained from the [12] and [47]. We did not run these models in our experiments. In the table, we present the results of the baselines as well as our model with beam search and greedy search. We experimented with 2, 4 and 6 number of beams. The abbreviations are as follows LM: Language Model, Mol. graph: Molecular graph, fragLM: Fragment-based Language Model, beam number: number of beams, and SMILES/Graph: model trained on SMILES or molecular graph.

Table 2: Model and baseline results on the ZINC and PCBA datasets.

Model	LM/Graph	Dataset	Validity	Novelty	Uniqueness
ChemVAE	LM	ZINC	0.170	0.980	0.310
GrammarVAE	LM	ZINC	0.310	1.000	0.108
SDVAE	LM	ZINC	0.435	-	-
GraphVAE	Mol. graph	ZINC	0.140	1.000	0.316
CGVAE	Mol. graph	ZINC	1.000	1.000	0.998
NeVAE	Mol. graph	ZINC	1.000	0.999	1.000
LFM	fragLM	ZINC	1.000	0.995	0.998
LFM	fragLM	PCBA	1.000	0.991	0.972
MolGAN	Mol. graph	ZINC	0.981	0.942	0.104
Ours (Greedy)	fragLM	ZINC	0.982	0.988	0.901
Ours (Beam 2)	**fragLM**	**ZINC**	**1.000**	**0.982**	**0.992**
Ours (Beam 4)	fragLM	ZINC	0.994	0.951	0.981
Ours (Beam 6)	fragLM	ZINC	0.995	0.961	0.995
Ours (Greedy)	fragLM	ZINC	0.973	0.963	0.921
Ours (Beam 2)	**fragLM**	**PCBA**	**1.000**	**0.956**	**0.998**
Ours (Beam 4)	fragLM	PCBA	0.995	0.945	0.996
Ours (Beam 6)	fragLM	PCBA	0.994	0.938	0.993

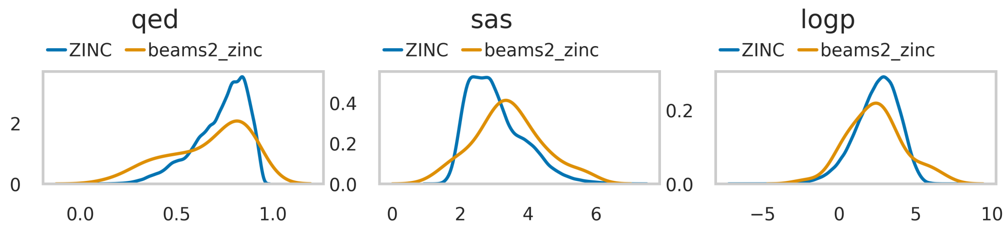


Fig. 3: Molecular property on ZINC dataset with beams=2

We also performed an analysis of our models' molecular properties on the ZINC and PCBA datasets. These properties are key to evaluating our model apart from the validity, uniqueness and novelty by showing the extent to which

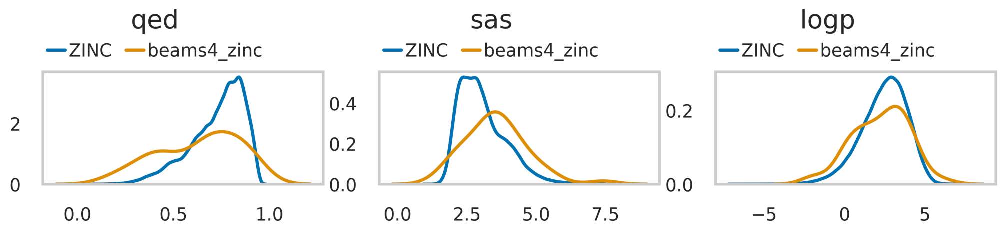


Fig. 4: Molecular property on ZINC dataset with beams=4

the model simulated the training data distributions. We present the results of three molecular properties (i) Drug-likeness: the extent to which the molecule is similar to the known drugs [6], Synthetic accessibility: the ease of synthesizing a given molecule [16] and, (iii) LogP: measures the solubility of a molecule. Figures 3, 4, 5 and 6 show the results of the molecular properties on the ZINC and PCBA datasets. In the Figure drug-likeness is abbreviated as qed, synthetic accessibility as sa and logP as logp. The results show that the model can generate molecules with similar properties to the training data. Notably, our model can optimize these objectives despite not being programmed explicitly to do so. Other baselines such as MolGAN, however, learn the distribution of the molecular properties while also optimizing the network towards the molecular property objective.

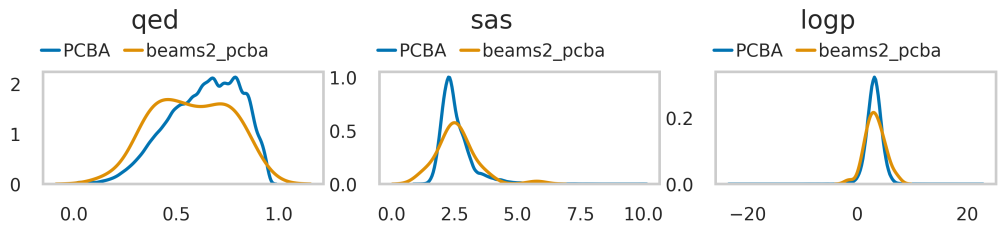


Fig. 5: Molecular property on PCBA dataset with beams=2

We generated some sample molecules based on the ZINC dataset and plotted them in Figure 7. In the Figure, the left includes samples drawn from the ZINC dataset and the right includes samples generated by our model with several beams set to 2. The generated molecules are chemically sound and structurally similar to the molecules in the ZINC dataset. The generated molecules are also diverse and unique which is consistent with the results in Table 2.

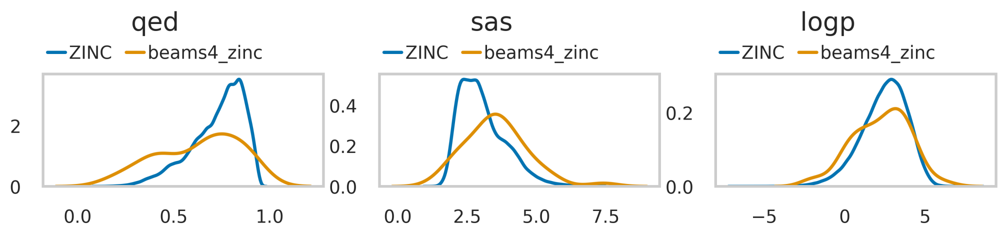
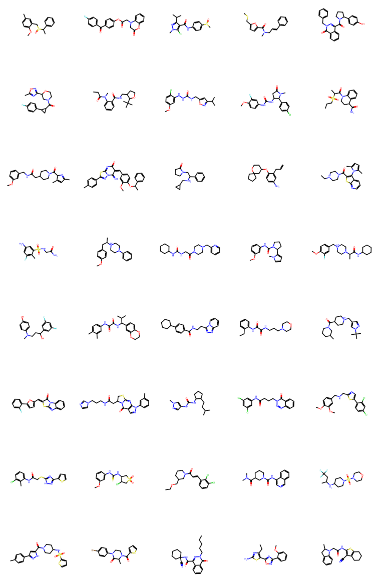


Fig. 6: Molecular property on PCBA dataset with beams=4


(a) ZINC samples

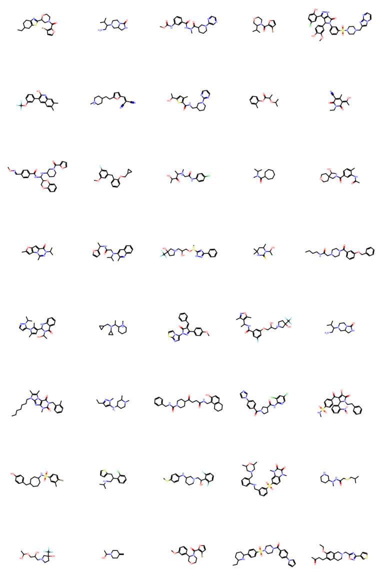

(b) Samples from beam = 2 model

Fig. 7: Generated sample compared to the ZINC dataset

6 Discussion

In this paper, we presented a fragment-based language model for molecular design. The model was then decoded in a greedy fashion as well as using beam search with varying numbers of beams; 2, 4 and 6. We evaluated the performance of the model on the ZINC and PCBA datasets. Our model was able to generate valid molecules with several beams set to 2 on both datasets. Decoding the language model with greedy search leads to suboptimal solutions compared to the beam search. For instance, on PCBA and ZINC datasets, the greedy search recorded validity of 0.973 and 0.982 respectively while the beam search with 2 beams recorded 1.000 on both. Another key observation is that decoding the language model with beam search leads to more unique and diverse molecules compared to the greedy search. For instance, on the ZINC dataset, the greedy search recorded a uniqueness of 0.921 while the beam search with 2, 4 and 6 beams recorded 0.992, 0.981 and 0.995 respectively. On the PCBA dataset, the greedy search recorded a uniqueness of 0.938 while the beam search with 2, 4 and 6 beams recorded 0.998, 0.996 and 0.993 respectively. Beam search allows the generation of molecules which are not only valid but also unique and diverse. Overall, the results recorded by our model are comparable to the state-of-the-art graph-based models such as MolGAN and NeVAE. Notably, our results are as competitive as the LFM model which is also a fragment-based model, however, our implementation is based on the transformer model while LFM is based on the Encoder-Decoder GRU model. Another key variation between our model and LFM is the complexity; our model is a decoder only while LFM is encoder-decoder and thus faster to train and converge faster.

Importantly, both greedy and beam search-based decoding models can optimize molecular properties such as drug-likeness, synthetic accessibility and logP. While the results are promising, they are not as competitive as the MolGAN model where such properties are explicitly optimized. This lays an interesting future research direction on how the fragment-based language model can be used to optimize molecular space for the generation of new molecules as well as optimize the molecular properties.

7 Conclusion and future research

In this paper, we presented a fragment-based language model for molecular design. The model is trained on the fragments of the molecules and is used to generate new molecules. We showed that the language model can be used to generate valid molecules comparable to the state-of-the-art graph-based models. We also showed how beam search can be used to improve the validity, uniqueness and diversity of the generated molecules. As part of future work, we plan to investigate fragment-based molecular modelling with larger language models such as GPT-3 and LLAMA models. Optimization of the molecular property with validity, uniqueness and novelty in an end-to-end fashion is also open for future research.

8 Acknowledgement

This work was granted access to the HPC/AI resources of IDRIS under the allocation 2023-AD010614071 made by GENCI.

9 On data and code

The datasets used in this work are public. The code and information on the model is available at https://github.com/steveowk/molfragement.

References

1. Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein gan: Machine learning. stat. ML (2017)
2. Ba, J.L., Kiros, J.R., Hinton, G.E.: Layer normalization. arXiv preprint arXiv:1607.06450 (2016)
3. Bagal, V., Aggarwal, R., Vinod, P., Priyakumar, U.D.: Molgpt: molecular generation using a transformer-decoder model. Journal of Chemical Information and Modeling **62**(9), 2064–2076 (2021)
4. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
5. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks **5**(2), 157–166 (1994)
6. Bickerton, G.R., Paolini, G.V., Besnard, J., Muresan, S., Hopkins, A.L.: Quantifying the chemical beauty of drugs. Nature chemistry **4**(2), 90–98 (2012)
7. Bojchevski, A., Shchur, O., Zügner, D., Günnemann, S.: Netgan: Generating graphs via random walks. In: International conference on machine learning. pp. 610–619. PMLR (2018)
8. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine **34**(4), 18–42 (2017)
9. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y.: Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
10. Dai, H., Tian, Y., Dai, B., Skiena, S., Song, L.: Syntax-directed variational autoencoder for molecule generation. In: Proceedings of the international conference on learning representations (2018)
11. Davidson, T.R., Falorsi, L., De Cao, N., Kipf, T., Tomczak, J.M.: Hyperspherical variational auto-encoders. arXiv preprint arXiv:1804.00891 (2018)
12. De Cao, N., Kipf, T.: Molgan: An implicit generative model for small molecular graphs. arXiv preprint arXiv:1805.11973 (2018)
13. Degen, J., Wegscheid-Gerlach, C., Zaliani, A., Rarey, M.: On the art of compiling and using'drug-like'chemical fragment spaces. ChemMedChem **3**(10), 1503 (2008)
14. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
15. Erlanson, D.A., McDowell, R.S., O'Brien, T.: Fragment-based drug discovery. Journal of medicinal chemistry **47**(14), 3463–3482 (2004)
16. Ertl, P., Schuffenhauer, A.: Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. Journal of cheminformatics **1**, 1–11 (2009)
17. Gómez-Bombarelli, R., Wei, J.N., Duvenaud, D., Hernández-Lobato, J.M., Sánchez-Lengeling, B., Sheberla, D., Aguilera-Iparraguirre, J., Hirzel, T.D., Adams, R.P., Aspuru-Guzik, A.: Automatic chemical design using a data-driven continuous representation of molecules. ACS central science **4**(2), 268–276 (2018)
18. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)
19. Grover, A., Zweig, A., Ermon, S.: Graphite: Iterative generative modeling of graphs. In: International conference on machine learning. pp. 2434–2444. PMLR (2019)

20. Guimaraes, G.L., Sanchez-Lengeling, B., Outeiral, C., Farias, P.L.C., Aspuru-Guzik, A.: Objective-reinforced generative adversarial networks (organ) for sequence generation models. arXiv preprint arXiv:1705.10843 (2017)
21. Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: Methods and applications. arXiv preprint arXiv:1709.05584 (2017)
22. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
23. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531 (2015)
24. Ho, J., Jain, A., Abbeel, P.: Denoising diffusion probabilistic models. Advances in neural information processing systems **33**, 6840–6851 (2020)
25. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
26. Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural networks **2**(5), 359–366 (1989)
27. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. pp. 448–456. pmlr (2015)
28. Irwin, J.J., Shoichet, B.K.: Zinc- a free database of commercially available compounds for virtual screening. Journal of chemical information and modeling **45**(1), 177–182 (2005)
29. Jin, W., Barzilay, R., Jaakkola, T.: Junction tree variational autoencoder for molecular graph generation. In: International conference on machine learning. pp. 2323–2332. PMLR (2018)
30. Johnson, D.D.: Learning graphical state transitions. In: International conference on learning representations (2022)
31. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
32. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
33. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Communications of the ACM **60**(6), 84–90 (2017)
34. Kumar, A., Voet, A., Zhang, K.Y.: Fragment based drug design: from experimental to computational approaches. Current medicinal chemistry **19**(30), 5128–5147 (2012)
35. Kusner, M.J., Paige, B., Hernández-Lobato, J.M.: Grammar variational autoencoder. In: International conference on machine learning. pp. 1945–1954. PMLR (2017)
36. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)
37. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)
38. Li, Y., Vinyals, O., Dyer, C., Pascanu, R., Battaglia, P.: Learning deep generative models of graphs. arXiv preprint arXiv:1803.03324 (2018)
39. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015)
40. Liu, Q., Allamanis, M., Brockschmidt, M., Gaunt, A.: Constrained graph variational autoencoders for molecule design. Advances in neural information processing systems **31** (2018)

41. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)
42. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
43. Luong, M.T., Le, Q.V., Sutskever, I., Vinyals, O., Kaiser, L.: Multi-task sequence to sequence learning. arXiv preprint arXiv:1511.06114 (2015)
44. Minervini, P., Demeester, T., Rocktäschel, T., Riedel, S.: Adversarial sets for regularising neural link predictors. arXiv preprint arXiv:1707.07596 (2017)
45. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)
46. Moore, G.E.: Cramming more components onto integrated circuits. Proceedings of the IEEE **86**(1), 82–85 (1998)
47. Podda, M., Bacciu, D., Micheli, A.: A deep generative model for fragment-based molecule generation. In: International conference on artificial intelligence and statistics. pp. 2240–2250. PMLR (2020)
48. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners (2019)
49. Samanta, B., De, A., Jana, G., Gómez, V., Chattaraj, P., Ganguly, N., Gomez-Rodriguez, M.: Nevae: A deep generative model for molecular graphs. Journal of machine learning research **21**(114), 1–33 (2020)
50. Sanh, V., Debut, L., Chaumond, J., Wolf, T.: Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108 (2019)
51. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: The semantic web: 15th international conference, ESWC 2018, Heraklion, Crete, Greece, June 3–7, 2018, proceedings 15. pp. 593–607. Springer (2018)
52. Schneider, G., Fechner, U.: Computer-based de novo design of drug-like molecules. Nature Reviews Drug Discovery **4**(8), 649–663 (2005)
53. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
54. Segler, M.H., Preuss, M., Waller, M.P.: Planning chemical syntheses with deep neural networks and symbolic ai. Nature **555**(7698), 604–610 (2018)
55. Sennrich, R., Haddow, B., Birch, A.: Neural machine translation of rare words with subword units. arXiv preprint arXiv:1508.07909 (2015)
56. Shao, L., Gouws, S., Britz, D., Goldie, A., Strope, B., Kurzweil, R.: Generating high-quality and informative conversation responses with sequence-to-sequence models. arXiv preprint arXiv:1701.03185 (2017)
57. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. nature **529**(7587), 484–489 (2016)
58. Simonovsky, M., Komodakis, N.: Graphvae: Towards generation of small graphs using variational autoencoders. In: Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27. pp. 412–422. Springer (2018)
59. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research **15**(1), 1929–1958 (2014)

60. Sun, H., Liu, X., Gong, Y., Zhang, Y., Jiang, D., Yang, L., Duan, N.: Allies: Prompting large language model with beam search. In: The 2023 Conference on Empirical Methods in Natural Language Processing (2023)
61. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. Advances in neural information processing systems **27** (2014)
62. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
63. Wang, H., Wang, J., Wang, J., Zhao, M., Zhang, W., Zhang, F., Xie, X., Guo, M.: Graphgan: Graph representation learning with generative adversarial nets. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)
64. Wang, Y., Bryant, S.H., Cheng, T., Wang, J., Gindulyte, A., Shoemaker, B.A., Thiessen, P.A., He, S., Zhang, J.: Pubchem bioassay: 2017 update. Nucleic acids research **45**(D1), D955–D963 (2017)
65. Weininger, D.: Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. Journal of chemical information and computer sciences **28**(1), 31–36 (1988)
66. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine learning **8**, 229–256 (1992)
67. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al.: Huggingface's transformers: State-of-the-art natural language processing. arXiv preprint arXiv:1910.03771 (2019)
68. You, J., Ying, R., Ren, X., Hamilton, W., Leskovec, J.: Graphrnn: Generating realistic graphs with deep auto-regressive models. In: International conference on machine learning. pp. 5708–5717. PMLR (2018)
69. Yu, L., Zhang, W., Wang, J., Yu, Y.: Seqgan: Sequence generative adversarial nets with policy gradient. In: Proceedings of the AAAI conference on artificial intelligence. vol. 31 (2017)