EXPLORING THE RECALL OF LANGUAGE MODELS: CASE STUDY ON MOLECULES

Anonymous authors
Paper under double-blind review

ABSTRACT

Most current benchmarks evaluate Generative Language Models based on the accuracy of the generated output. However, in some scenarios, evaluating the recall of the generations is more valuable, i.e., whether a model can generate all correct outputs, such as all security vulnerabilities of a given codebase. There are two challenges in evaluating the recall: the lack of complete sets of correct outputs for any task and many distinct but similar outputs (e.g., two exploits that target the same vulnerability).

In this paper, we propose a benchmark from the domain of small organic molecules. We define several sets of molecules of varying complexity and fine-tune language models on subsets of those sets. We characterize set complexities via recall of model generations and show that we can predict the recall for a given number of generations in advance, using only perplexity on a held-out validation set. This prediction method extends to several i.i.d. sampling generation methods. Subsequently, we propose a novel decoding method based on beam search that maximizes recall by avoiding duplicates. Finally, we design a recall-aware loss function that leverages intuition from prior experiments to improve model recall for small language models. We perform additional analyses on the impact of molecular representation and model pretraining.

1 INTRODUCTION

Evaluating the performance of generative models, particularly language models, is an important challenge in modern deep learning (Chang et al., 2024). Most of the current benchmarks evaluate the ability of models to produce correct output, e.g., correctly answer questions, translate a sentence, generate a coherent story on a given topic, etc.

An overlooked aspect of evaluation is the ability of generative models to generate *all* correct outputs for the given input. This capability is crucial for security-focused applications. In software code analysis, generating exploits that target all vulnerabilities (Liguori et al., 2021; Yang et al., 2023) is a significant challenge. In language model security, one would like to find all "jailbreaks", i.e., the prompts that would force the target model to produce undesired outputs (Samvelyan et al., 2024), which allows for patching language models in the post-training phase before making them publicly available. In healthcare, AI assistants can suggest causes for the given symptoms, and sometimes, it is desirable to list not only the most probable, but all possible conditions that could produce those symptoms. In scientific discovery, generating new molecules or materials with given characteristics is a cornerstone problem. For example, in drug discovery, most of the correctly generated molecules may prove useless in subsequent phases of drug development (e.g., in toxicity analysis), so generating a diverse and complete set of initial molecules is useful. Another related problem is the exhaustive generation of all conformations (3D positions) for a given molecule. These conformations then become inputs for multiple downstream tools in drug discovery (Watts et al., 2010).

The ability of language models to cover all correct outputs without incorporating retrieval has yet to be systematically evaluated. We believe there are two significant obstacles. First, developing a benchmark that includes all correct outputs is hard. Second, the representations of the objects we are trying to generate are typically not unique. For example, the same security exploit can be written in multiple ways by changing variable names or refactoring; the same health condition can be described in different terms. More formally, the set of correct outputs is usually split into equivalence classes.

An ideal evaluation benchmark should ignore multiple generations from the same equivalence class and count the number of distinct classes a generative model can cover.

In this paper, we propose a benchmark that overcomes both obstacles and enables research into optimizing the recall of language models. We start from GDB-13 (Blum & Reymond, 2009), a complete database of molecules with specific characteristics and, most significantly, having at most 13 heavy atoms. We design four sets of molecules of varying complexity and train language models on small subsets of them. We represent molecules by SELFIES strings (Krenn et al., 2020), for which the equivalence classes are well-defined and can be computed algorithmically. We use the models to generate millions of molecules and evaluate the precision and recall of the generations by leveraging information about equivalence classes. We show that we can predict recall in advance without performing generation and devise novel recall-optimized generation methods and loss functions. We hope that this system for LLM evaluation will prove to be a valuable tool in other domains, like security, where the same software vulnerability can be exploited with similar attacks or the same language model exploit can be triggered with different wording.

2 Related Work

2.1 Evaluating Molecular Generation

A variety of methods have been successfully applied to molecular generation tasks, including GFlowNets (Bengio et al., 2021; Kim et al., 2024), recurrent neural networks (Guo & Schwaller, 2023; Blaschke et al., 2020), and graph-based genetic algorithms (Jensen, 2019) among others. Finally, LLMs have recently demonstrated strong performance on these tasks, especially when combined with iterative training, prompt design, and reinforcement learning methods (Wang et al., 2024; Guevorguian et al., 2024; Guo & Schwaller, 2024). In order to directly compare these methods, benchmarks in molecular generation and molecule optimization emphasize measuring the number of molecules generated under constraints that satisfy pre-specified criteria. For instance, the Practical Molecular Optimization benchmark involves a suite of tasks evaluated using the area under the curve of the scores of generations throughout the optimization process (Gao et al., 2022). However, this does not measure the diversity of generations, a critical criterion for generative models in industrial settings. Similarly, benchmarks involving docking simulations count the number of molecules generated that exceed certain property thresholds (Lee et al., 2024; Guo & Schwaller, 2023).

2.2 Recovering Molecular Datasets

Blum & Reymond (2009) introduced GDB-13, a complete set of small molecules with certain characteristics. Arús-Pous et al. (2019) attempted to re-generate the GDB-13 by training a recurrent neural network (RNN) on 1 million randomly sampled molecules. Arús-Pous et al. (2019) explored the effect of SMILES randomization on the recall. This line of papers motivated our work to explore the recall abilities of modern language models. However, the set of molecules these papers were trying to recover are simple and do not require significant knowledge of chemistry, as they can be described as all possible graphs with certain rules on node labels and graph properties. The sets we introduce in this paper have varying complexity and rely on more complex chemical features.

2.3 Recall of Generative Models

For generative image models, the need for and introduction of precision and recall-based evaluation methods is an extension of Fréchet Inception Distance (Heusel et al., 2017), which allow for a more comprehensive understanding of tradeoffs between the quality and diversity of generated images. In a seminal work, (Sajjadi et al., 2018) demonstrated a method to estimate these metrics by comparing dataset and model distributions, with subsequent works making these estimations more accurate (Kynkäänniemi et al., 2019; Park & Kim, 2023) and efficient to implement (Liang et al., 2024). Further research has been done to provide stronger theoretical motivation and a unifying framework for these methods (Sykes et al., 2024). Evaluating language models in terms of recall is a comparatively new direction, with fewer existing application methods and theoretical background. Several benchmark datasets evaluate models' ability to recall information from a provided corpus of text (Amouyal et al., 2022; Akhtar et al., 2024). In order to characterize the diversity of generations,

the MAUVE score describes the divergence criteria via a suite of comparison measures between ground truth and generated texts (Pillutla et al., 2021) , which was expanded and improved upon in (Pimentel et al., 2022) by performing more comprehensive ablations and operating on more granular textual features. The research closest to our research work continues in this direction and uses a K-nearest neighbor estimator on text embeddings of generated text under reduced dimensionality (Bronnec et al., 2024).

For molecular generation tasks, the ability to precisely evaluate the recall of generations for a predefined closed set of desired molecules provides more evidence for the effectiveness of a modeling approach compared to current threshold-based evaluatory frameworks. Methods proposed for general recall evaluation of generative models typically focus on other generative architectures (Generative Adversarial Networks, Variational Autoencoders, Gaussian Processes) or operate directly on image data. Existing methods that target LLMs require retrieving data from a text corpus or using a K-nearest neighbors (KNN) estimator to approximate model recall. By contrast, our problem setting allows for the direct calculation of recall. We show that we can predict this recall from quantities already implemented in standard model evaluation pipelines (perplexity) without additional modeling considerations. Finally, this paper is the only work that proposes methods for improving language model recall derived from the recall problem formulation we define.

3 Problem Definition

Let $\mathbb{S}$ be the set of all correct generations, i.e., strings in the language modeling context. Assume there is an equivalence relation among the strings in $\mathbb{S}$ which divides $\mathbb{S}$ into M equivalence classes. We denote the set of equivalence classes by $\mathbb{S}^u$ (u stands for *unique*). Each equivalence class corresponds to a single object. For any object $m \in \mathbb{S}^u$, the size of its equivalence class, i.e., the number of distinct strings corresponding to that object, is denoted by $\|m\|$.

The goal is to train a model that can generate strings from the maximum number of equivalence classes, i.e., a maximum number of unique objects. To achieve that, we train a (potentially pretrained) language model on a subset of M objects. Suppose we also have a *validation set*, i.e., a subset of V objects ($V < M$) distinct from the training set. After training, we generate G strings by sampling from the model. Let $\mathbb{G}$ denote all generated strings.

We evaluate the model using the following metrics. True positives, denoted by TP, are the generated strings that belong to $\mathbb{S}$. Note that $\mathbb{G}$ can contain duplicate strings but may also contain distinct strings that belong to the same equivalence class. Hence we also define *unique true positives*, $TP^u = |\mathbb{G}^u|$, as the number of equivalent classes represented in $\mathbb{G}$. We track two metrics:

$$Precision(G) = \frac{TP}{G} \quad , \qquad Recall(G) = \frac{TP^u}{M}$$

Suppose a process samples $\mathbb{G}$ in an i.i.d. fashion. In that case, the number of true positives will scale linearly with G, and precision will not depend on G (after a sufficiently large number of generations). On the contrary, TP^u does not scale indefinitely with G as it is upper bounded by $M = |\mathbb{S}^u|$. Hence, the recall increases with G and can ideally reach M and stay constant. Section 4.3 shows how precision can depend on G if the sampling process is not i.i.d..

The ideal model can learn to put uniform $p = \frac{1}{M}$ probability on all objects of the set $\mathbb{S}^u$. Note that in this ideal scenario, the probability of each object can be arbitrarily distributed over its string representations. The recall of the ideal model after G generations will be $1 - (1 - p)^G$. Thus, this quantity is the theoretical upper bound for the recall of i.i.d. sampling methods.

Below, we define four sets of molecules. Each molecule can be represented by multiple strings that form the equivalence class of the molecule. Hence, $\mathbb{S}$ denotes the set of all string representations of all molecules in the set.

3.1 Molecular Datasets

We take GDB-13 (Blum & Reymond, 2009), an exhaustive set of molecules with at most 13 heavy atoms that satisfy certain conditions. Although the list of conditions is not small, it is well-defined. Essentially, GDB-13 is constructed by taking all planar graphs of at most 13 vertices (enumerated

using Nauty package (McKay & Piperno, 2014)) and putting atoms on the vertices in a way that satisfies a set of chemical criteria. Several filters are further applied to enforce chemical stability and synthetic feasibility rules. We refer the reader to the supplementary material of (Blum & Reymond, 2009) for the details. For the scope of this paper, we note that the set is exhaustive under the chosen conditions. Next, we define four subsets of GDB-13 in Table 1. In the rest of the paper, we define

Table 1: Definitions of four subsets of GDB-13.

Name	Size	Description
$\mathbb{S}_{asp}$	8,284,280	The set of (all strings of) molecules from GDB-13 that have at least 0.4 similarity with aspirin.
$\mathbb{S}_{sas}$	6,645,440	The set of easily synthesizable molecules from GDB-13, more precisely, the subset of molecules having SAS score less than 3.
$\mathbb{S}_{d>p}$	9,331,077	The set of molecules that have at least 0.4 similarity to paracetamol (a famous drug) and have less than 0.4 similarity to 4-nitroanisole (a famous toxic molecule, a "poison").
$\mathbb{S}_{d=p}$	8,051,185	The set of molecules m that are at a similar distance from paracetamol (d) and 4-nitroanisole (p): $0.2 \leq sim(m,d) \leq 0.2165$ and $0.2 \leq sim(m,p) \leq 0.2165$.

the similarity between molecules (denoted by $\text{sim}(m_1, m_2)$) as the Tanimoto similarity (Tanimoto, 1958) between MACCS fingerprints (Durant et al., 2002). We also use synthetic accessibility score (SAS) (Ertl & Schuffenhauer, 2009), a score between 1 (easily synthesizable) and 10 (very difficult to make).

Note that these sets have different complexities. Intuitively, similarity in terms of MACCS fingerprints implies certain shared substructures of molecules. Hence, $\mathbb{S}_{asp}$ contains molecules that share a certain percentage of substructures with aspirin. $\mathbb{S}_{d>p}$ is more complicated as it contains molecules that share some substructures with paracetamol but also do not share many structures with a toxic substance. $\mathbb{S}_{d=p}$ is the most complicated, as it cannot be described solely with substructures. $\mathbb{S}_{sas}$ is technically more complicated than $\mathbb{S}_{asp}$ as its formula includes statistics about various fragments, but on the other hand, the contribution of those statistics in the final score is too little; the score is dominated by "easier" variables like number of atoms.

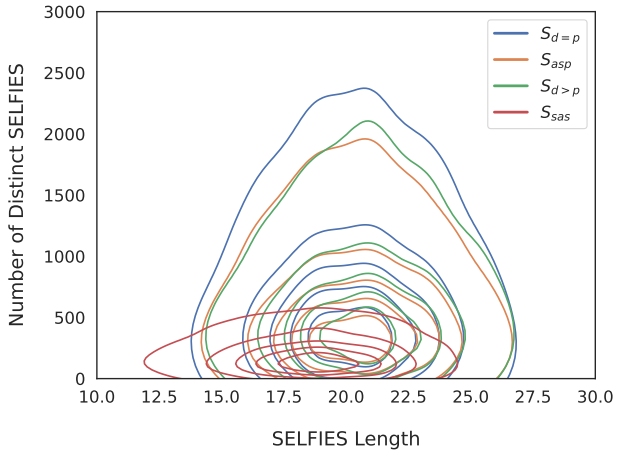

Figure 1: The statistics of the four sets of molecules we describe in Section 3.1. $\mathbb{S}_{sas}$ has a noticeably different distribution of molecules compared to the other three sets, which might help to explain certain divergences from the overall patterns seen in the later sections.

4 EXPERIMENTS

For pretraining, we utilize a large subset of 848 million molecules from GDB-13 that explicitly excludes the four sets defined above. We split this data into a training set and a 10,000-instance validation set. We adopt most of the pretraining settings and model architecture from the OPT 1.3B model (Zhang et al., 2022), with additional information in A.1. We represent molecules using SELFIES strings (Krenn et al., 2020). Note that while it guarantees that the generated strings in its alphabet correspond to some molecules, it is not necessarily helpful for the quality of the generated molecules as shown by Skinnider (2024). We perform experiments demonstrating the impacts of the SELFIES representation in the analysis section. Canonical SELFIES correspond to a specific, uniform, rule-based molecular structure traversal. Similarly, we define randomized SELFIES as corresponding to random traversals of the molecule. Refer to the appendix for more details on definitions of canonical and random SELFIES.

Training is performed from scratch for one epoch over the entire pretraining dataset, ensuring each training string is used exactly once. For fine-tuning, we take our pretrained models and fine-tune them on the four sets, from which we randomly select 1 million molecules as training data. For the baseline models, we utilize the same OPT 1.3B architecture and employ the finetuning procedure on the four subsets without pretraining.

We adopt the following methodology for each subset to construct evaluation sets encompassing nearly all randomized versions of a given set of molecules. We randomly select 500 SELFIES strings from each subset. For each selected SELFIES string, we loop 1 million times and ask the RDKit library to generate an unrestricted randomized version. We finalized the set by removing all duplicates to ensure a unique set of randomized versions for each molecule.

4.1 CHARACTERIZING DATA MODELLING DIFFICULTY WITH RECALL

Above, section 3.1 described the relative complexity of our molecular subsets based on domain knowledge of the rules used to create them. Generating molecules from models trained on these sets, in Table 2, we provide an interpretable characterization of the difficulty of modeling these sets by directly evaluating recall for the trained language models, a unigram model trained on the same data, and the i.i.d. upper bound sampling process described in section 3. Note that we show results for a setting where the number of generations equals the number of molecules in the corresponding set. Otherwise, the recall numbers between different subsets would not be directly comparable because the size of the respective sets (M) varies. Here, the upper bound is the same for all sets: 63.21%. The unigram model performs poorly compared to our models, confirming that a more sophisticated language modeling approach is necessary to represent these molecular sets. The results for the trained language models show that in terms of recall, $\mathbb{S}_{sas}$ is the easiest one, followed by $\mathbb{S}_{asp}$, $\mathbb{S}_{d>p}$ and $\mathbb{S}_{d=p}$. The observed ranking of modeling difficulty across molecular subsets aligns with what was proposed in 3.1 based on domain-informed analysis of the set definitions.

Table 2: **Recall (%)** of OPT-1.3B language models fine-tuned on four sets of molecules, on $G = M$ strings generated with random sampling.

Pretraining	**Fine-tuning**	$\mathbb{S}_{asp}$	$\mathbb{S}_{sas}$	$\mathbb{S}_{d>p}$	$\mathbb{S}_{d=p}$
Canonical	Canonical	48.36	**58.44**	40.72	12.40
Canonical	Randomized	47.58	56.24	39.12	10.35
Randomized	Canonical	48.21	57.97	41.00	12.89
Randomized	Randomized	**50.12**	58.05	**41.56**	**12.92**
Upper bound (i.i.d.)		63.21	63.21	63.21	63.21
Unigram model		0.03	0.48	0.0034	0.0057

Figure 2 shows how true positives and unique true positive molecules grow as the number of generated strings grows. The plot indicates that the recall is close to saturation at 10 million generations, implying that this model will not cover 90% of the molecules even with 50 million generations. This result motivates us to look for other approaches to improve the recall scores of the language models in section 4.3.

Table 3: **Precision (%)** of OPT-1.3B language models fine-tuned on four sets of molecules, on 10 million strings generated with random sampling.

Pretraining	**Fine-tuning**	$\mathbb{S}_{asp}$	$\mathbb{S}_{sas}$	$\mathbb{S}_{d>p}$	$\mathbb{S}_{d=p}$
Canonical	Canonical	75.69	**80.58**	68.27	14.07
Canonical	Randomized	70.59	73.26	61.63	10.86
Randomized	Canonical	**76.16**	80.26	**68.93**	**14.58**
Randomized	Randomized	75.15	76.17	65.66	13.67
Unigram model		0.05	15.72	0.02	0.01

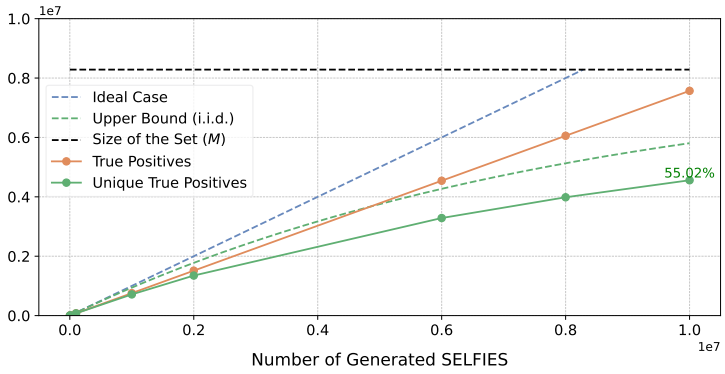

Figure 2: Number of true positive strings and unique true positive molecules generated by the OPT-1.3B model fine-tuned on aspirin-like molecules.

4.2 Predicting Recall Without Generating

Given that evaluating recall provides a meaningful and interpretable measure of an approach's ability to model data, estimating recall without needing to perform generations would be useful. This subsection shows that this is possible and that the predicted recall can be derived from the autoregressive loss quantity on a representative set. We first compute the probability that the model will generate a molecule from the validation set in G attempts. Let the j-th SELFIES string of the i-th molecule $s_{i,j}$ contain K tokens $s^k_{i,j}$, $k = 1, \ldots, K$. The language model, when sampling a new string, will select $s^1_{i,j}$ as the first token with $p(s^1_{i,j})$ probability, the second token with $p(s^2_{i,j}|s^1_{i,j})$ probability, continuing in this manner for subsequent tokens. Under this formulation, if we can access the model, we can compute the expected probability of generating the entire string. Let us denote this probability by $p_{i,j}$. Then, the probability of the i-th molecule m_i to be generated in a single attempt is $\sum_{j=1}^{\|m_i\|} p_{i,j}$. The average probability of a desired molecule to be generated in a single attempt becomes $\sum_{i=1}^{M} \sum_{j=1}^{\|m_i\|} p_{i,j}$. Note that this is the expected precision of the model, as the model will produce a correct string with exactly this probability at each sampling iteration.

To compute the expected value for recall in G sampling iterations, we take the probability that the i-th given molecule will *not* be sampled in G iterations, and subtract it from one: $1 - \left(1 - \sum_{j=1}^{\|m_i\|} p_{i,j}\right)^G$. The expected value of this quantity over all molecules is the expected recall at G generations.

Assuming access to a small validation set of V molecules, one can estimate the precision and recall using only those:

$$Precision = \frac{M}{V} \sum_{i=1}^{V} \sum_{j=1}^{\|m_i\|} p_{i,j} \qquad Recall = \frac{1}{V} \sum_{i=1}^{V} \left(1 - \left(1 - \sum_{j=1}^{\|m_i\|} p_{i,j}\right)^G\right) \tag{1}$$

Note that this formula holds for any i.i.d. sampling method. For example, if the probabilities are scaled by temperature or by removing the tail by top-p or top-K approaches, then $p_{i,j}$ scores in the above formulas will correspond to the scaled values.

We use these formulas to estimate precision and recall for various combinations of pretraining and fine-tuning, random sampling, and temperature sampling across various sets of molecules. The Pearson correlation between predicted and actual precision scores is 0.99975 and 0.99982 for the recall scores. Figure 3 shows the correlation between predicted and actual recall scores for 4 subsets. Here we use the model with canonical fine-tuning after canonical pretarining and generate 1M and 10M SELFIES for each subset. These results demonstrate that we can reliably use the predicted scores for model selection.

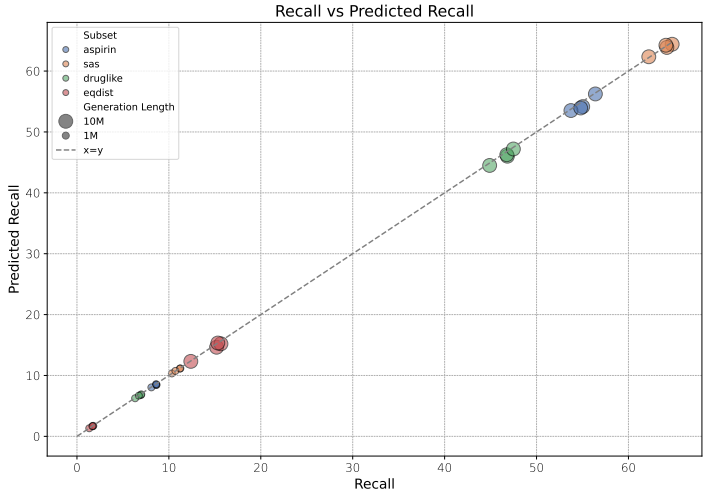

Figure 3: The correlation plot of actual recall and predicted recall for all 4 subsets.

4.3 Recall-oriented Generation

The experiments so far used only random sampling. Over the past few years, many papers have focused on improving the sampling procedures of language models. Here, we investigate how basic sampling strategies affect precision and recall. We then present a new generation method motivated by our findings to improve model recall. For the model pretrained on canonical SELFIES and fine-tuned on canonical ones from $\mathbb{S}_{asp}$, we performed generation with **temperature sampling**. High temperature means high entropy and more diverse generations, simultaneously increasing the risk of incorrect generations. Figure 4 shows how the true positive generations (i.e., precision) drop as the temperature grows. It also shows that the optimal number of unique true positive molecules is achieved with $T = 0.8$ temperature.

All generation methods studied above suffer from duplicates in $\mathbb{G}$. **Beam search**, another commonly used sampling strategy for language models, can be used to mitigate this issue. The regular beam search is an extension of the greedy algorithm and uses a relatively small beam size to keep the best B generations at each token. At the end of the generation, the algorithm produces B distinct sequences, and the top one is selected. We propose to keep all generations of the beam search and use a beam size equal to G. We used beam search with $B = 10^6$ and $B = 10^7$ on one of the language models fine-tuned on $\mathbb{S}_{asp}$. The results are presented in Table 4, demonstrating a significant improvement in recall using this approach. Furthermore, the beam size can be adjusted to modulate precision, or the selection can be limited to the top-ranked generated sequences. We leave the prediction of the recall of the generations using beam search to future work.

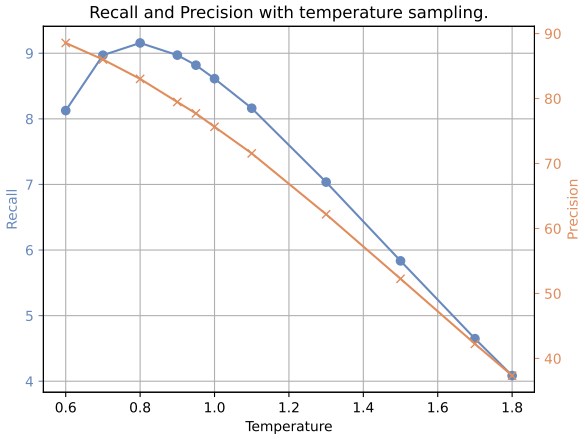

Figure 4: Temperature sampling for the model fine-tuned on $\mathbb{S}_{asp}$. $G = 10^6$ molecules are generated to calculate Recall (%) and Precision (%).

Table 4: Precision and recall of the OPT-1.3B model trained on $\mathbb{S}_{asp}$ (only canonical SELFIES in both phases) using different sampling methods, including beam search. Note that the upper bound does not apply to beam search, as it is not an i.i.d. generation. We did not perform 10^7 generations with low-temperature sampling.

Sampling	**Precision (%)**		**Recall (%)**	
	$G = 10^6$	$G = 10^7$	$G = 10^6$	$G = 10^7$
Random sampling	75.65	75.69	8.61	55.02
Temperature ($T = 0.8$)	83.02	83.02	9.16	51.21
Upper bound (i.i.d.)	100	100	11.37	70.01
Beam search ($B = G$)	93.08	71.03	10.71	69.17

4.4 Recall-oriented Loss Function

Our next target is the loss function in the fine-tuning phase. One potential problem is that the models "waste efforts" on generating multiple SELFIES strings of the same molecules. We notice from Tables 3 and 8 that a model trained on randomized SELFIES during both training phases has 18.75 percentage points higher precision than recall, which means 1.875 million generated strings are correct (i.e., they belong to $\mathbb{S}_{asp}$), but represent the same molecules that are duplicated in the 56.40%. Some of these strings are duplicated SELFIES strings, but others are different SELFIES representations of the same molecule. Notably, using canonical SELFIES for both training phases does not improve this gap.

In this subsection, we perform an experiment explicitly forcing the network to focus on one SELFIES only. In contrast to earlier experiments, we enlarge the training set with 8 SELFIES for each of the 1 million molecules. Then, we design the batches such that all variants of the same molecule appear in the same batch. We use batch size 128, which covers 16 molecules with 8 variants each. The loss function during fine-tuning first aggregates the loss over the 8 variants and then takes the average across 16 distinct molecules.

We use three kinds of aggregation functions: **mean**, **minimum**, and **maximum**. Mean aggregation is equivalent to the regular loss function; it puts equal weight on all SELFIES strings available to the model. Minimum aggregation forces the model to optimize only the best SELFIES string, i.e., the one with the lowest loss. The hypothesis is that the model will direct its capacity on the easiest string only. We hypothesize that the rest of its capacity will be allocated to covering more molecules (instead of more SELFIES of the "known" molecules), hence increasing the recall. Maximum ag-

gregation has the opposite effect, it forces the model to put efforts on optimizing even the hardest SELFIES string.

Table 5 shows that our hypothesis did not hold generally. Mean aggregation performs the best both in terms of precision and recall. Diverging from the regular loss function in both directions does not help. We thought that the 1.3B parameter model might have too much capacity. Hence, there is no actual competition between storing information about diverse molecules vs. about various SELFIES strings of the same molecule. We tried the same experiment with smaller 125M and 800K parameter versions of OPT (pretrained on the same set of canonical SELFIES). Although the same result was observed for the 125M parameter model, the 800K model benefited from the minimum aggregation loss. This discrepancy suggests that there may be a relationship between total model capacity and the benefit gained from the proposed loss formulations. A more detailed investigation of this relationship belongs to future work.

Table 5: Performance of the fine-tuned models on $\mathbb{S}_{asp}$ with different aggregation functions and model sizes.

Aggregation	**Precision (%)**			**Recall (%)**		
	OPT-1.3B	OPT-125M	OPT-800K	OPT-1.3B	OPT-125M	OPT-800K
Minimum	74.35	73.68	**56.4**	8.48	8.39	**6.43**
Maximum	64.60	61.60	33.4	7.42	7.09	3.87
Mean	**78.35**	**76.04**	47.5	**8.96**	**8.70**	5.46

Designing recall-oriented loss functions for fine-tuning remains a challenge for future work.

5 Analysis and Additional Results

5.1 Impact of pretraining

First, we examine the impact of pretraining on model performance via recall. To make the comparison, we performed the same fine-tuning as in section 4 on a randomly initialized OPT-1.3B model for the $\mathbb{S}_{asp}$ set and generated 1 million molecules using random sampling. The results, presented in Table 6, show that pretraining consistently helps across all sets of molecules for both precision and recall.

Table 6: **Precision** and **Recall** of OPT-1.3B language models fine-tuned on four sets of molecules (only on canonical SELFIES), on 1 million strings generated with random sampling.

Pretraining	**Precision (%)**				**Recall (%)**			
	$\mathbb{S}_{asp}$	$\mathbb{S}_{sas}$	$\mathbb{S}_{d>p}$	$\mathbb{S}_{d=p}$	$\mathbb{S}_{asp}$	$\mathbb{S}_{sas}$	$\mathbb{S}_{d>p}$	$\mathbb{S}_{d=p}$
No pretraining	61.01	68.07	52.19	6.45	6.92	9.47	5.29	0.79
Canonical	75.64	80.55	68.31	14.04	8.61	11.25	6.95	1.72
Randomized	76.22	80.28	68.88	14.54	8.66	11.20	7.00	1.78
Upper bound	100	100	100	100	11.37	13.97	10.16	11.68

5.2 Molecular Representations

5.2.1 SMILES vs SELFIES Representations

We represent molecules using SELFIES strings (Krenn et al., 2020). Note that while it guarantees that the generated strings in its alphabet correspond to some molecules, it is not necessarily helpful for the quality of the generated molecules as shown by Skinnider (2024). To verify this in terms of precision and recall, we train two additional models with SMILES representations, one with canonical SMILES and another with randomized SMILES. We compare the performance of these two models with that of the SELFIES-based models. In Table 7, we present the differences in precision

and recall between SMILES and SELFIES. We evaluate the precision and recall of OPT-1.3B language models, pretrained on randomized representations and fine-tuned on four molecular datasets. After fine-tuning, 1 million strings were generated using random sampling. The results indicate that SMILES performs better with canonical fine-tuning, while SELFIES excels with randomized fine-tuning. It is important to note that all pretraining was conducted using the randomized versions for both SMILES and SELFIES.

Table 7: The comparison of models' performance trained with SMILES and SELFIES representations. For example, the precision difference between models trained with SMILES and SELFIES, such as $\Delta\mathbb{S}_{asp} = 1.74$, highlights the performance gap between the two representations.

Fine-tuning	**Precision (%)**				**Recall (%)**			
	$\Delta\mathbb{S}_{asp}$	$\Delta\mathbb{S}_{sas}$	$\Delta\mathbb{S}_{d>p}$	$\Delta\mathbb{S}_{d=p}$	$\Delta\mathbb{S}_{asp}$	$\Delta\mathbb{S}_{sas}$	$\Delta\mathbb{S}_{d>p}$	$\Delta\mathbb{S}_{d=p}$
Canonical	1.74	0.80	1.94	1.83	0.20	0.12	0.20	0.22
Randomized	-2.28	-0.26	-0.63	-0.05	-0.26	-0.03	-0.07	-0.01

5.2.2 Canonical vs Randomized Representations

In section 4, we use canonical SELFIES representations for pretraining and randomized SELFIES for fine-tuning. We define Canonical SELFIES of molecules as SELFIES generated via conversion from the canonical SMILES representation produced by the RDKit library. By assigning unique numbers to each atom in a molecule, RDKit enables a consistent traversal of the molecular structure. This process ensures that the SMILES string generated will always be the same for a given molecule. Likewise, to achieve the random SELFIES representation, we randomize the SMILES strings and then transform them into SELFIES. It is possible to obtain randomized SMILES by altering the atom ordering, which does not change how the algorithm traverses the graph (e.g., depth-first in the case of RDKit) but changes the starting point and the order in which branching paths are selected. Arús-Pous et al. (2019) defines two versions of randomization: restricted and unrestricted. The unrestricted version allows the graph to be traversed without any constraints, whereas the restricted version imposes certain restrictions, such as prioritizing sidechains when traversing a ring. We used restricted versions in the pretraining data, but we used the unrestricted version during fine-tuning.

We used random sampling with temperature 1.0 to generate 10 million molecules from each of the four fine-tuned models to better understand the tradeoffs between training on canonical vs randomized SELFIES. Tables 3 and 2 show the key evaluation metrics. Surprisingly, there is little difference between the models trained on randomized and canonical SELFIES. The largest difference in recall is less than 3 percentage points over 10 million samples. This result is in contrast with the findings of Arús-Pous et al. (2019) (note the sets of molecules are different). While the difference is small, pretraining on randomized SELFIES is better for precision and recall on three out of four sets, with $\mathbb{S}_{sas}$ being the only exception. For precision, fine-tuning only on canonical is better, while fine-tuning on randomized is preferable for recall. Pretraining on canonical and fine-tuning on randomized SELFIES performs the worst on all molecular sets. We do not have a clear understanding of the causes of these patterns. The recall numbers from Table 8 are not directly comparable across molecular sets because the size of the respective sets (M) varies. This is also evident from the upper bound numbers. To make the numbers comparable, Table 2 shows the recall when the number of generations equals the number of molecules in the respective set.

6 Conclusion

In this paper, we presented a benchmark for evaluating the recall of language models. We showed how different factors affect the recall and highlighted a path toward improving the recall through sampling methods and loss functions. We also present a novel method to predict generative recall without doing generation. Whether pretraining or fine-tuning strategies of language models can significantly improve their recall is still an open question. We will publicly release the sets of molecules and the pretrained models to foster further research on methods for improved recall of language models. Future work should develop methods covering much smaller datasets with few to no training samples.

REFERENCES

Mubashara Akhtar, Chenxi Pang, Andreea Marzoca, Yasemin Altun, and Julian Martin Eisenschlos. Tanq: An open domain dataset of table answered questions. *ArXiv*, abs/2405.07765, 2024. URL https://api.semanticscholar.org/CorpusID:269757585.

Samuel Joseph Amouyal, Tomer Wolfson, Ohad Rubin, Ori Yoran, Jonathan Herzig, and Jonathan Berant. Qampari: A benchmark for open-domain questions with many answers. In *IEEE Games Entertainment Media Conference*, 2022. URL https://api.semanticscholar.org/CorpusID:249062559.

Josep Arús-Pous, Thomas Blaschke, Silas Ulander, Jean-Louis Reymond, Hongming Chen, and Ola Engkvist. Exploring the gdb-13 chemical space using deep generative models. *Journal of cheminformatics*, 11:1–14, 2019.

Josep Arús-Pous, Simon Johansson, Oleksii Prykhodko, Esben Jannik Bjerrum, Christian Tyrchan, Jean-Louis Reymond, Hongming Chen, and Ola Engkvist. Randomized smiles strings improve the quality of molecular generative models. *Journal of Cheminformatics*, 11, 2019. URL https://api.semanticscholar.org/CorpusID:208206203.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Advances in Neural Information Processing Systems*, 34:27381–27394, 2021.

Thomas Blaschke, Josep Arús-Pous, Hongming Chen, Christian Margreitter, Christian Tyrchan, Ola Engkvist, Kostas Papadopoulos, and Atanas Patronov. Reinvent 2.0: an ai tool for de novo drug design. *Journal of chemical information and modeling*, 60(12):5918–5922, 2020.

Lorenz C Blum and Jean-Louis Reymond. 970 million druglike small molecules for virtual screening in the chemical universe database gdb-13. *Journal of the American Chemical Society*, 131(25): 8732–8733, 2009.

Florian Le Bronnec, Alexandre Verine, Benjamin Négrevergne, Yann Chevaleyre, and Alexandre Allauzen. Exploring precision and recall to assess the quality and diversity of llms. In *Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://api.semanticscholar.org/CorpusID:267740404.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 15(3):1–45, 2024.

Joseph L Durant, Burton A Leland, Douglas R Henry, and James G Nourse. Reoptimization of mdl keys for use in drug discovery. *Journal of chemical information and computer sciences*, 42(6): 1273–1280, 2002.

Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1:1–11, 2009.

Wenhao Gao, Tianfan Fu, Jimeng Sun, and Connor W. Coley. Sample efficiency matters: A benchmark for practical molecular optimization. *ArXiv*, abs/2206.12411, 2022. URL https://api.semanticscholar.org/CorpusID:250072218.

Philipp Guevorguian, Menua Bedrosian, Tigran Fahradyan, Gayane Chilingaryan, Hrant Khachatrian, and Armen Aghajanyan. Small molecule optimization with large language models. *ArXiv*, abs/2407.18897, 2024. URL https://api.semanticscholar.org/CorpusID:271516332.

Jeff Guo and Philippe Schwaller. Augmented memory: Capitalizing on experience replay to accelerate de novo molecular design. *ArXiv*, abs/2305.16160, 2023.

Jeff Guo and Philippe Schwaller. Saturn: Sample-efficient generative molecular design using memory manipulation. *arXiv preprint arXiv:2405.17066*, 2024.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Neural Information Processing Systems*, 2017. URL https://api.semanticscholar.org/CorpusID:326772.

Jan H Jensen. A graph-based genetic algorithm and generative model/monte carlo tree search for the exploration of chemical space. *Chemical science*, 10(12):3567–3572, 2019.

Hyeon-Seob Kim, Minsu Kim, Sanghyeok Choi, and Jinkyoo Park. Genetic-guided gflownets: Advancing in practical molecular optimization benchmark. *ArXiv*, abs/2402.05961, 2024.

Mario Krenn, Florian Häse, AkshatKumar Nigam, Pascal Friederich, and Alan Aspuru-Guzik. Self-referencing embedded strings (selfies): A 100 *Machine Learning: Science and Technology*, 1 (4):045024, oct 2020. doi: 10.1088/2632-2153/aba947. URL https://dx.doi.org/10.1088/2632-2153/aba947.

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. In *Neural Information Processing Systems*, 2019. URL https://api.semanticscholar.org/CorpusID:118648975.

Seul Lee, Seanie Lee, Kenji Kawaguchi, and Sung Ju Hwang. Drug discovery with dynamic goal-aware fragments. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=xuX2rDSSco.

Yuanbang Liang, Jing Wu, Yu-Kun Lai, and Yipeng Qin. Efficient precision and recall metrics for assessing generative models using hubness-aware sampling. In *International Conference on Machine Learning*, 2024. URL https://api.semanticscholar.org/CorpusID:270974756.

Pietro Liguori, Erfan Al-Hossami, Vittorio Orbinato, Roberto Natella, Samira Shaikh, Domenico Cotroneo, and Bojan Cukic. Evil: exploiting software via natural language. In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, pp. 321–332. IEEE, 2021.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Brendan D McKay and Adolfo Piperno. Practical graph isomorphism, ii. *Journal of symbolic computation*, 60:94–112, 2014.

Dogyun Park and Suhyun Kim. Probabilistic precision and recall towards reliable evaluation of generative models. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 20042–20052, 2023. URL https://api.semanticscholar.org/CorpusID:261530847.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaïd Harchaoui. Mauve: Measuring the gap between neural text and human text using divergence frontiers. In *Neural Information Processing Systems*, 2021. URL https://api.semanticscholar.org/CorpusID:244488758.

Tiago Pimentel, Clara Meister, and Ryan Cotterell. On the usefulness of embeddings, clusters and strings for text generator evaluation. 2022. URL https://api.semanticscholar.org/CorpusID:253735234.

Mehdi S. M. Sajjadi, Olivier Bachem, Mario Lucic, Olivier Bousquet, and Sylvain Gelly. Assessing generative models via precision and recall. *ArXiv*, abs/1806.00035, 2018. URL https://api.semanticscholar.org/CorpusID:44104089.

Mikayel Samvelyan, Sharath Chandra Raparthy, Andrei Lupu, Eric Hambro, Aram H Markosyan, Manish Bhatt, Yuning Mao, Minqi Jiang, Jack Parker-Holder, Jakob Foerster, et al. Rainbow teaming: Open-ended generation of diverse adversarial prompts. *arXiv preprint arXiv:2402.16822*, 2024.

Michael A. Skinnider. Invalid smiles are beneficial rather than detrimental to chemical language models. *Nature Machine Intelligence*, 2024. URL https://api.semanticscholar.org/CorpusID:268788258.

Benjamin Sykes, Loic Simon, and Julien Rabin. Unifying and extending precision recall metrics for assessing generative models. *ArXiv*, abs/2405.01611, 2024. URL https://api.semanticscholar.org/CorpusID:269587938.

Taffee T Tanimoto. Elementary mathematical theory of classification and prediction. 1958.

Haorui Wang, Marta Skreta, Cher Tian Ser, Wenhao Gao, Lingkai Kong, Felix Streith-Kalthoff, Chenru Duan, Yuchen Zhuang, Yue Yu, Yanqiao Zhu, Yuanqi Du, Alán Aspuru-Guzik, Kirill Neklyudov, and Chao Zhang. Efficient evolutionary search over chemical space with large language models. *ArXiv*, abs/2406.16976, 2024. URL https://api.semanticscholar.org/CorpusID:270711201.

K Shawn Watts, Pranav Dalal, Robert B Murphy, Woody Sherman, Rich A Friesner, and John C Shelley. Confgen: a conformational search method for efficient generation of bioactive conformers. *Journal of chemical information and modeling*, 50(4):534–546, 2010.

Guang Yang, Yu Zhou, Xiang Chen, Xiangyu Zhang, Tingting Han, and Taolue Chen. Exploitgen: Template-augmented exploit code generation based on codebert. *Journal of Systems and Software*, 197:111577, 2023.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona T. Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. Opt: Open pre-trained transformer language models. *ArXiv*, abs/2205.01068, 2022. URL https://api.semanticscholar.org/CorpusID:248496292.

ZJUNLP. Molgen-large. https://huggingface.co/zjunlp/MolGen-large, 2024. Accessed: 2024-05-22.

A Appendix

A.1 Model Training Details

Our models were trained using the AdamW optimizer (Loshchilov & Hutter, 2019) with the following hyperparameters: $\beta_1 = 0.9$, $\beta_2 = 0.95$ and $\epsilon = 10^{-5}$. The learning rate follows a linear schedule, annealing to zero, with a peak learning rate of $4e^{-4}$ and 2,588 warmup steps. We use a weight decay of 0.1, gradient clipping of 1.0, and a batch size of 128 with gradient accumulation steps of 32. The maximum sequence length is 64, and dropout is set to 0. We employ mixed-precision training (fp16). All training was conducted using the Hugging Face library. We pretrain a 1.3B parameter model on eight A100 GPUs, each with 40GB of VRAM. Training on our dataset, which comprises 20 billion tokens, takes approximately two days.

For tokenization, we use an off-the-shelf tokenizer from ZJUNLP (2024) with a vocabulary size of 192, including a few additional tokens for debugging the models.

A.2 Additional Tables

Table 8: Recall of OPT-1.3B language models fine-tuned on four sets of molecules, on 10 million strings generated with random sampling.

Pretraining	Fine-tuning	$\mathbb{S}_{asp}$ (%)	$\mathbb{S}_{sas}$ (%)	$\mathbb{S}_{d>p}$ (%)	$\mathbb{S}_{d=p}$ (%)
Canonical	Canonical	55.02	**64.76**	46.83	15.19
Canonical	Randomized	53.74	62.21	44.90	12.39
Randomized	Canonical	54.80	64.18	46.76	**15.67**
Randomized	Randomized	**56.40**	64.10	**47.48**	15.33
Upper bound (i.i.d.)		70.09	77.79	65.76	71.12