Applying language models to algebraic topology: generating simplicial cycles using multi-labeling in Wu's formula

Kirill Brilliantov [* 1] **Fedor Pavutnitskiy** [* 2] **Dmitry Pasechnyuk** [3 4] **German Magai** [5 6]

Abstract

Computing homotopy groups of spheres has long been a fundamental objective in algebraic topology. Various theoretical and algorithmic approaches have been developed to tackle this problem. In this paper we take a step towards the goal of comprehending the group-theoretic structure of the generators of these homotopy groups by leveraging the power of machine learning. Specifically, in the simplicial group setting of Wu's formula, we reformulate the problem of generating simplicial cycles as a problem of sampling from the intersection of algorithmic datasets related to Dyck languages. We present and evaluate language modelling approaches that employ multi-label information for input sequences, along with the necessary group-theoretic toolkit and non-neural baselines.

1. Introduction

Mathematical progress often depends on the identification of patterns and the formulation of conjectures, which are statements believed to be true but yet to be proven. For centuries, mathematicians have relied on data, from early prime tables to modern computer-generated data, to aid in this process. While computational techniques have proven useful in some areas of pure mathematics (Appel & Haken, 1977), (Hales, 2005), the full potential of artificial intelligence (AI) in the field is yet to be fully explored.

In mathematics, AI has demonstrated remarkable abilities in tasks such as identifying counterexamples to con-

jectures (Wagner, 2021), accelerating calculations (Peifer et al., 2020), generating symbolic solutions (Lample & Charton, 2019), detecting structural patterns in mathematical objects (He, 2021), automated generation of formal proofs (Han et al., 2021), (Polu et al., 2022) and guiding human intuition to making hypothesis (Davies et al., 2021). These capabilities offer vast potential for expanding mathematical knowledge and advancing the field.

Inspired by these recent advanced in AI driven mathematical research, we propose a proof-of-concept framework for generating simplicial cycles in simplicial groups. This serves as an initial step towards a broader initiative of utilizing machine learning techniques to sample elements from the homotopy groups of spheres and more general spaces. In the current work[1] we apply our ideas in particular case of free simplicial group which models (loops over) a two-dimensional sphere. We adopt a simplicial and group-theoretic approach due to its combinatorial nature and computer-friendly representation.

The theoretical background for Wu's formula (4) is outlined in Section 3 with a more detailed exposition given in Appendix B. This formula expresses our object of interest as a certain quotient of the intersection of normal subgroups of free group. Generating elements from this intersection $\cap_i R_i$ (see notation (1) below) is the main topic of the present work. The key features of the problem include the similarity between R_i's and sets of Dyck-n words (allowing a suitable distribution on it to sample from) and a clear algorithmic procedure for determining if a given element w lies within the intersection.

We develop the following non-neural-based baselines for this problem: random search, evolutionary method and greedy algorithm. These baselines are extensively described in Section 4.2. Concurrently, our main deep learning approaches described in Section 4.3 are based on language modelling and differ in the various usages of training dataset's meta information.

The goal of introducing the deep learning approaches is to

---

*Equal contribution [1]ETH, Zurich [2]Beijing Institute of Mathematical Sciences and Applications, Beijing [3]Mohamed bin Zayed University of Artificial Intelligence, Abu Dhabi [4]Ivannikov Institute for System Programming of the Russian Academy of Sciences, Moscow [5]HSE University, Moscow [6]Noeon Research, Tokyo. Correspondence to: Fedor Pavutnitskiy <fedor.pavutnitskiy@gmail.com>.

---

[1]Code available at `https://github.com/ml-in-algebraic-topology/gen-simplicial-cycles`

increase the performance in terms of the number of generated words from the full intersection, while also achieving scalability of the model. This is crucial due to the rapid growth in computational complexity observed in the baseline algorithms, rendering them impractical for application in higher dimensions. The relevant set of metrics, datasets description and overall evaluation results are presented in Section 5 together with some relevant discussions pertaining to the findings.

We hope that our work will be of interest not only to algebraic topologists and specialists in formal languages but also to the general public interested in text generation. For them, a potential real-world application scenario may involve controllable text generation that combines multiple text classes without explicitly knowing their intersections.

2. Related Work

The computation of homotopy groups of spheres stands as one of the most challenging problems in mathematics. Over the past century, the complexity of this problem has spurred the development of numerous sophisticated mathematical techniques. For instance, in the 1950s, Toda introduced the concept of secondary operations (now called Toda brackets), which allows him to compute the homotopy group of spheres up to a certain dimension (Toda, 1963). In the 1960s, unstable Adams spectral sequences were developed, providing a general framework for such computations and extending the range of results achieved by Toda (Adams, 1958). In the 1990s a fresh approach using Goodwillie tower of identity functor (Arone & Mahowald, 1999), (Behrens, 2012) was employed, leveraging modern techniques to recompute Toda's range.

In recent years, advancements have been made in computational and computer science approaches concerning the computation of homotopy groups.

A classical result by Brown (Brown, 1957) established that homotopy groups of spheres are finitely computable. However, it should be noted that the algorithm presented in Brown's work, as discussed in (Cadek et al., 2014), exhibits superexponential complexity. While this result lays a solid theoretical foundation for the development of algorithms and computational methods for computing these groups, the proof itself does not directly yield practical algorithms for their computation. Nonetheless, it has sparked new avenues of research and possibilities for researchers interested in devising effective computational methods for computing homotopy groups.

While the problem of computing even the weaker invariants like the rationalization $\pi_n X \otimes \mathbb{Q}$ for a general space X is known to be $\#P$-hard (Anick, 1987), there have been notable advancements in algorithmic computation of homo-

topy groups and homology of spaces.

The authors of the Kenzo system (Sergeraert, 1994),(Rubio & Sergeraert, 2002), (Romero et al., 2006) since the 1990s have been actively developing an algorithmic framework based on spectral sequences to compute homotopy groups of certain classes of spaces. Their concept of *spaces with effective homology* was further enhanced in (Cadek et al., 2014), where a polynomial-time algorithm was introduced for computing $\pi_n X$ for a **fixed** value of n within the framework of *simplicial sets with polynomial time homology*.

Besides the computational challenges, one limitation of current approaches is that $\pi_n X$ is typically computed as an abelian group, without providing a detailed account of its internal homotopical structure, such as a description of generators and relations in algebraic, geometric, or topological terms. However, there have been efforts (Filakovsky et al., 2018) proposing approaches to address this limitation.

Since our work operates on a group-theoretic level, it relates to a corpus of research on generating formal languages. The general limitations of such approaches are discussed in (Delétang et al., 2022), (Hahn, 2020) and (Weiss et al., 2018), with more specific discussions on Dyck languages appearing in (Weiss et al., 2021), (Ebrahimi et al., 2020).

The utilization of machine learning techniques in the calculation of homotopy groups of spaces is a relatively novel and emerging approach. So far, the corresponding research concerning both machine learning and algebraic topology, primarily dedicated to enhancing machine learning algorithms with ideas from algebraic topology, but lacks exploration of the inverse. The present work initiates the process of filling this gap.

3. Group Theory of Wu's Formula

To keep the preliminary exposition concise, we will provide a brief overview of the algebraic constructions employed in this paper, reserving more formal and detailed information for Appendix A. Wu's formula for $\pi_n S^2$, introduced in (Wu, 2001), combines the simplicity of group theoretic language of free group with a powerful combinatorial structure of simplicial groups. In our perspective, the group theoretic structures in Wu's formula, related to formal languages, presents a compelling target for the application of language models. Additionally, the usage of simplicial groups opens up further avenues for incorporating machine learning to a more sophisticated mathematical constructions of simplicial homotopy theory, like the unstable Adams spectral sequence (Bousfield et al., 1966).

By *free group* F on a set $x_1, \ldots x_n$ we will mean a set of words in letters x_i (called *generators*) and their inverses with group operation of string concatenation. The identity

$x_i x_i^{-1} = x_i^{-1} x_i = 1$ is assumed in F, here 1 is an empty word. Additionally, we will denote by $x_0 = x_1 \dots x_n$ the product of all generators x_i. We will further use the following notation: $u^v = v^{-1} u v$ called *conjugation* and $[u, v] = u^{-1} v^{-1} u v$ called *commutator* for $u, v \in F$.

We will be interested in generating words from the following subsets of F:

$$R_i = \{x_i^{\varepsilon_1 y_1} \dots x_i^{\varepsilon_k y_k} \mid k \in \mathbb{N}, y_j \in F\}, \qquad (1)$$

where $\varepsilon_j \in \{1, -1\}$, $i = 0..n$. The subgroups R_i, also denoted by $\langle x_i \rangle^F$, to emphasize the generating element, are called the *normal closures* of x_i's. These subgroups can be conceptually understood as follows: for $i > 0$ the subgroup R_i consists of words that become trivial after substituting x_i for 1, for example:

$$[x_1, x_2] x_1 \in R_1, \ [x_1, x_2] x_1 |_{x_1 = 1} = 1.$$

Similarly, R_0 consist of words which become trivial after any of the equivalent substitutions like $x_1 \mapsto x_n^{-1} x_{n-1}^{-1} \dots x_2^{-1}$, which maps x_0 to 1. Equivalently, R_0 consists of words which become trivial upon removal the cyclic permutations of x_0.

As a result, the intersection $\cap_{i=0}^n R_i$ consists precisely of words that become trivial when any of the (cyclic permutations of) x_i is substituted with 1. The objective of this paper is to propose a several deep learning generators of such words.

Since $[u, 1] = [1, v] = 1$ for all $u, v \in F$, it is clear that the elements of the form $[x_i, x_j]$ are in the intersection $R_i \cap R_j$ and, more generally, $[R_i, R_j] \subseteq R_i \cap R_j{}^2$. This idea can be iterated by introducing the *iterated commutators*

$$[u_1, \dots, u_k] = [[[u_1, u_2], u_3] \dots, u_k], \ u_i \in F, \quad (2)$$

and for a set of subgroups $\{R_{i_1}, \dots, R_{i_k}\}$ we have:

$$\prod_{\sigma \in \Sigma_k} [R_{\sigma(i_1)}, \dots R_{\sigma(i_k)}] \subseteq \bigcap_{j=1}^k R_{i_j}, \qquad (3)$$

where the left-hand side have a product over all permutations on k letters, reflecting the symmetric nature of the intersection on the right-hand side. We will abbreviate the left-hand side of the inclusion (3) as $[R_0, \dots, R_n]_S$ and call it a *symmetric commutator subgroup*.

The following surprising result is important for training our models:

Theorem 3.1 ((Wu, 2001), Corollary 3.5)**.** *For any proper subset $\{i_0, \dots, i_k\} \subsetneq \{0, \dots, n\}$ the inclusion (3) is an equality, i.e.*

$$R_{i_1} \cap \dots \cap R_{i_k} \cong [R_{i_1}, \dots, R_{i_k}]_S$$

---

²Here $[R_i, R_j]$ denotes the subgroup generated by the elements $[u, v]$, $u \in R_i$, $v \in R_j$.

$\pi_{n+1}(S^2)$	Non-trivial element
$\mathbb{Z}$	$[x_1, x_2]$
$\mathbb{Z}/2$	$[[x_1, x_2], [x_1, x_2 x_3]]$
$\mathbb{Z}/2$	$[[[x_1, x_2], [x_1, x_2 x_3]], [[x_1, x_2], [x_1, x_2 x_3 x_4]]]$

Table 1: Elements from $\cap_{i=0}^n R_i$, that are not contained within $[R_0, \dots, R_n]_S$ for $n = 2, \ 3, \ 4$.

for all proper subsets $\{i_1, \dots, i_k\} \subsetneq \{0, \dots, n\}$.

This implies that any partial (different from $\cap_{i=0}^n R_i$) intersection of R_i can be easily described and, after defining the suitable distribution, sampled from. In (Wu, 2001) simplicial methods are used to demonstrated that the difference between the symmetric commutator subgroup and the **full** intersection, as expressed in formula (3), possesses a homotopical nature:

Theorem 3.2 (Wu's formula, (Wu, 2001) Theorem 1.7)**.** *For $n \geq 1$ there is an isomorphism*

$$\pi_{n+1}(S^2) \cong \frac{R_0 \cap R_1 \cap \dots \cap R_n}{[R_0, R_1, \dots, R_n]_S}. \qquad (4)$$

Although not computational in nature, the formula (4) gives an elegant combinatorial descrtiption of homotopy groups of S^2. This description motivates the present work.

Remark 3.3. It is important to note that the problem of classifying whether a given word $w \in \cap_i R_i$ is *trivial* (is an element of the denominator of formula (4)) or *non-trivial* (contributes to a non-zero element of $\pi_* S^2$) is almost as hard as the problem of computing homotopy groups themselves. To our knowledge, there are no known algorithms or approaches to tackle this problem in a general setting, each elements requires an individual method for classification. For instance, for $n = 2, \ 3, \ 4$ the group theoretic form of the corresponding generators of the homotopy groups is given in Table 1, see also (Mikhailov, 2021).

In our research, we aim to generate elements from the numerator of Wu's formula without providing any information about its denominator to the model, thereby enforcing it not to "learn" the pattern of the trivial words in the denominator. Furthermore, during the training phase of our algorithms, we do not intend to utilize the knowledge of non-trivial elements listed in Table 1, since we want our approaches to be scalable. However, we can consider utilizing the non-trivial elements and the denominator in the evaluation phase to assess the performance of our approaches.

4. Generating Elements from the Intersection of Normal Closures

4.1. Creating Datasets from Normal Closures and their Commutator Subgroups

Generating a consistent synthetic dataset is of paramount importance for data-driven projects. In this regard, one of the biggest challenges is to ensure that the dataset is not only large enough but also exhibits a reliable structure. In our project, we encountered a similar challenge while constructing a training dataset comprising words from R_i and their partial intersections.

In order to effectively train our models, we need to have a certain amount of control on properties of training dataset, for example being able to control to some extent the distribution of lengths of the words and their internal structure.

When it comes to generating words from R_i, we discovered that using a *naive* sampling approach, where each y_k in formula (1) is sampled independently, yielded less desirable properties in the generated words: the adjacent letters of different conjugators y_k, y_{k+1} do not interact with each other, resulting in reduced variability within the generated dataset.

To address this issue, we developed an alternative description of R_i, based on balanced bracket sequences and Dyck-n languages. To generate a word from R_i, we can uniformly sample a balanced bracket sequence (Atkinson & Sack, 1992) and then replace every matching pairs of brackets with words a and b such that the word ab is either identity or the rotation of the normal closure's generator. For instance, if $n = 3$, then in case of R_0 the opening bracket a could be x_2 and the corresponding closing bracket will be $b = x_3 x_1$. We refer to this sampling method as *bracket-style sampling*.

We use the notion of Dyck paths to analyse the difference between naive and bracket-based sampling. By *Dyck-n word* we will mean a balanced bracket sequence with brackets of n types. For Dyck-n word b the corresponding *Dyck path* $p(b)$ is a plot that depicts the variation in the number of unbalanced brackets as a function of the position in the word. By *valley* on Dyck path $p(b)$ we will mean a local minimum which is greater than zero.

Figure 1 illustrates the difference in distribution of valleys for naive and bracket-style sampling methods. While the distributions of word lengths remain similar, the bracket-style sampling method shows more variety in possible number of valleys in corresponding Dyck paths. Note that each valley corresponds to a mutual cancellation of letters in adjacent conjugators y_k and y_{k+1}.

Now, to generate elements from iterative commutator subgroups like $[R_{i_0}, \ldots, R_{i_k}]$ we use the following procedure. First, we sample a collection of random binary trees that represent products of various commutators. Then for each leaf j of every tree the element of $R_{\sigma(j)}$ is sampled using bracket-style sampling, where $\sigma \in \Sigma_k$ is a random permutation. Note that although commutators of arbitrary complexity can be expressed as product of iterated commutators of the form (2) using Hall-Witt identity (see Appendix A for details), we found that sampling not only iterated, but arbitrary commutators, such as $[[x_{i_1}, x_{i_2}], [x_{i_3}, x_{i_4}]]$ yields better results.

Finally, to ensure that the length distribution of words is equal among various partial intersections, we employed the Kolmogorov-Smirnov test. Further details regarding supplementary properties of the sampling methods can be found in Appendix D.

Remark 4.1. For a fixed word length we can also measure the coverage of the sampler by the proportion of the unique words generated by it per batch. In particular, for a fixed subgroup R_i for small lengths l we can count the total number $N = N(l)$ of words of length l in R_i. Define the *l-variability* $V = V(l)$ of the sampler s as the number of generated by s unique words of length l divided by N. Table 2 shows the advantage of the bracket style sampler against the naive one in terms of the variability.

Length	3	4	5	6	7	8
Bracket	1.0	1.0	1.0	1.0	0.74	0.25
Naive	0.86	0.68	0.39	0.24	0.14	0.05

Table 2: Variability V of bracket-style and naive sampler for generating elements from R_1. Sampling parameters: batch size $b = 10^5$, `radius` = 10, $n = 4$ generators.

4.2. Baselines

We choose a range of non-neural based algorithms to approach a problem of generating elements from the intersection $\cap_{i=0}^{n} R_i$, given algorithms for generating elements from each individual R_i.

Random Search Guided by Theorem 3.1, the most straightforward and readily implementable method is to randomly sample words belonging to $[R_0, \ldots, \hat{R}_i, \ldots, R_n]_S$ and check whenever such these words lie in R_i. Notation $\hat{R}_i$ here means that R_i is excluded from the symmetric commutator subgroup. We are performing this procedure for all $i \neq 0$ simultaneously. Already in random search the difference between naive sampling and bracket-style sampling from R_i, discussed in Section 4.1, can be seen in terms of number of generated words from intersection per batch (see Appendix D for additional experiments).

---

[3]The words are: $y^2[x, z^{-1}]y^{-2}[z^{-1}, p][x, z^{-1}][p, z^{-1}]$ with 0 valleys and $p^2[z, x][pxp, y]p^{-2}$ with 2 valleys

(a) Dyck paths of a typical word[3]

(b) Distribution of valleys
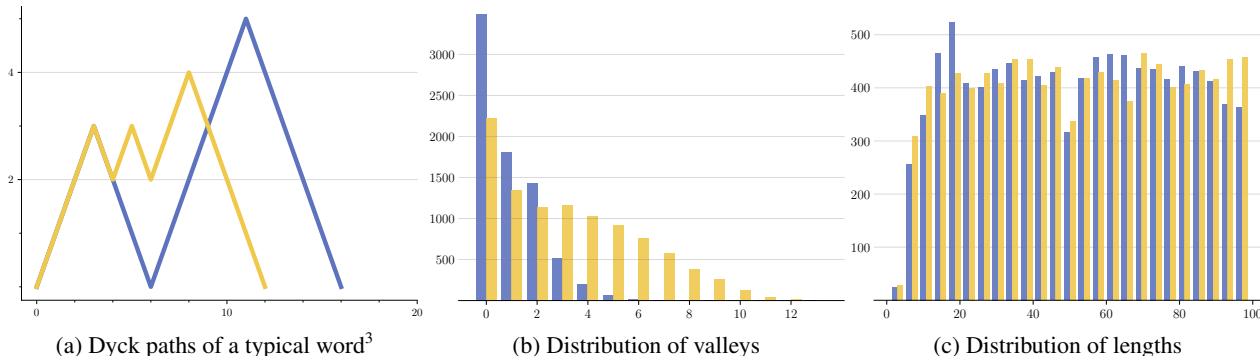
(c) Distribution of lengths

Figure 1: Comparison of naive (blue) and bracket-style (yellow) sampling. Sample of 10^4 elements of R_1 with $n = 3$ with a maximum length of 100.

Evolutionary Method As described in Section 3, the words in $\cap_i R_i$ have are characterized by the following property: replacing any of x_i with 1 will reduce the word to 1. This criterion underlies the extremal reformulation of the task of generating words from the intersection of R_i and the following algorithm.

For a normal closure $R = \langle v \rangle^F$ of a word v and a word w, let the *algebraic distance* $d(w, R)$ be defined by the length of word after throwing away all occurrences of cyclic permutations of v from w. Note that $d(w, R) = 0$ iff $w \in F$. This definition resembles algebraic distance (Taubin, 1991) between a point x and the kernel of a map $f : \mathbb{R}^d \to \mathbb{R}$, defined by $d(x, \ker f) = |f(x)|$.

Next, let's introduce the non-negative objective function

$$d(w) = \sum_{i=0}^{n} d(w, R_i)$$

and consider the corresponding discrete optimisation problem $\min_{w \in F} d(w)$, solution to which will give an element from $\cap_{i=0}^{n} R_i$.

To solve this discrete optimisation problem we apply a modification of the $(1 + 1)$-evolutionary algorithm (Droste et al., 2002) with mutation replacing each of randomly chosen letters in the word with (uniformly) random sampled generator or its inverse. It is worth mentioning that one of our metrics (see Section 5) is inspired by the notion of distance between the intersection $\cap_i R_i$ and a given word.

Greedy Algorithm Experiments with language models (Section 4.3 below) that generate a full word based on a prefix inspired us to find a non-neural based iterative procedure of appending a next token to a given prefix to generate a full word from the intersection. For each normal closure R_i, a stack S_i is created from the given prefix. Specifically, S_i is obtained by removing x_i, its inverse, and cyclic permutations (in the case where $i = 0$) from the prefix.

To determine the next token in the sequence, the algorithm employs a greedy strategy based on the current tops of the stacks. Each token is assigned points based on two mutually exclusive criteria. The first criterion is that the token candidate should decrease the length of a particular stack. For instance, if the top of a stack S_i is x_j, then the next token candidate x_j^{-1} would receive points. The second criterion is that the token candidate should not increase the length of any stack. For example, the token x_i does not increase the length of S_i since it is already excluded from it. Tokens that would reduce the length of the given prefix itself are excluded and assigned zero points. The algorithm then selects the next token with the highest score.

For illustrative purposes we included a toy example of application of this method in Appendix D

4.3. Deep Models

General Architecture Details We now move to the neural-based algorithms for sampling elements from $\cap_i R_i$. We mostly use decoder-only transformers (Radford & Narasimhan, 2018), (Vaswani et al., 2017) as a main architecture for our models. We will briefly review its features relevant to our problem at hand.

Decoder-only models are often used in tasks where the input sequence is fixed or pre-processed, such as language modelling (Jurafsky & Martin, 2000), text generation (Li et al., 2022), which aligns with our task of generating words as sequences of tokens that matches a certain target distribution.

We adopt the classical auto-regressive approach, where each token in the output sequence is generated based on the previous tokens and the decoder's internal state. Specifically, at each time step, the decoder takes in the previous token in the output sequence, along with a context vector that summarizes the encoded input sequence, and generates a new token using a softmax (M., 2006) function.

5

We also utilize standard modifications of decoder-only transformer models, such as causal masking, which ensures that the model only attends to previous tokens in the output sequence, and positional embeddings, which provide information about the position of each token in the sequence. These modifications help the model generate high-quality and coherent output sequences (Brown et al., 2020).

To train such a model we use a cross-entropy loss (Bengio, 2008):

$$L(y, \hat{y}) = -\sum_k \sum_i y_{k,i} \log(\hat{y}_{k,i}) \tag{5}$$

where $y_{k,i}$, is a true probability distribution (of k-th token in the sequence having value i), $\hat{y}_{k,i} = \mathbf{Pr}_\theta(s_k = i \mid s_1 \ldots s_{k-1})$ is the corresponding predicted probability distribution (which depends on model parameters θ) , $s_1, \ldots, s_t$ is a sequence of tokens from the training dataset.

We use generators x_k and their inverses x_k^{-1} as tokens, together with the special tokens: $\langle\mathbf{bos}\rangle$ to identify the beginning of the sequence, $\langle\mathbf{eos}\rangle$ to identify the end of the sequence, and $\langle\mathbf{pad}\rangle$ to identify *empty* tokens after the end of the sequence (ignored in formula (5)). The inclusion of the $\langle\mathbf{bos}\rangle$ token has been shown to significantly improve the performance of attention-based models in Dyck-n language recognition (Ebrahimi et al., 2020), see also (Hahn, 2020), (Weiss et al., 2021).

Finally, to sample from models we use two configurations: beam search (Tillmann & Ney, 2003) with the repetition penalty (Keskar et al., 2019) and nucleus sampling (Holtzman et al., 2020).

4.3.1. METHODS

The key ingredient in our deep learning-based generating approaches is multi-label for each word in the training dataset. Multi-label of the word w provides information about the membership of w in each of the subgroups R_i:

Definition 4.2. For an element w of a free group on n generators, the *multi-label* is a binary array of length $(n+1)$ such that its i-th position decodes whether w contained in R_i or not.

We developed a range of approaches which are different from each other in a way how they treat multi-labels.

Ignore This procedure involves the deliberate exclusion of multi-label information during both the training and inference phases. This approach is based on the underlying assumption that the model exhibits sufficient capacity to effectively generalize from a dataset containing words associated with various multi-labels. In essence, this can be described as a zero-shot generation, a problem that has been extensively studied in recent years (Wang et al., 2019), (Cao et al., 2020).

Mask The second approach involves selectively masking specific components of the model in accordance with the corresponding multi-labels. In this method each attention head is assigned a normal closure, and during the training phase we mask the attention heads that do not correspond to normal closures containing the given word by multiplying their output with zero matrix. In the inference phase we do not mask anything to prompt the model to generate from the complete intersection. Using this masking approach we try to help the model to distribute the knowledge about the normal closures across its parts.

Remark 4.3. It is worth noting the *ensemble technique*, which is a particular case of the masking approach. This technique involves training of $n + 1$ models on each normal closure. During the inference phase, the given prefix is fed to each model, and next token is sampled from the joint distribution, which is essentially the sum of the outputs distributions

$$f(w) = \text{Softmax}\left(\sum_{i=0}^n M_i(w) \times \chi_{\{w \in R_i\}}\right),$$

with χ being a characteristic function of a subset. This approach is functionally equivalent to creating a model that comprises $n + 1$ transformers, with the logits of the last output layers being summed together, as Figure 2 illustrates. During training, the outputs of each model in ensemble are correspondingly masked according to the multi-label.

Prompt Our third approach utilizes multi-label information as a *prompt* for the model. To facilitate this, we expand the vocabulary by introducing tokens for each normal closure $R_0, \ldots, R_n$ along with a token $\langle:\rangle$ that serves as an identifier for the end of the prompt. Prior to inputting the words to the model, we prefix the corresponding prompt to every word. For instance, in the case of a word $x_1 x_2 \ldots x_t$ with multi-label [1, 1, 0, 0], we provide the model $R_0 R_1 : x_1 x_2 \ldots x_t$ as input. In the evaluation phase we prompt the model with $R_0 \ldots R_n$:, indicating our intention to obtain a word from the complete intersection. Prompting has become increasingly popular in recent years (Liu et al., 2021) as a means of fine-tuning pre-trained models for specific tasks such as question-answering, summarization, and sentiment analysis. By providing targeted prompts, the model's output can be optimized for specific use cases, making it more accurate and efficient at performing specific tasks.

Negative Baseline We can collect the dataset of words from the $[R_0, \ldots, R_n]_S$, i.e. a denominator of Wu's for-
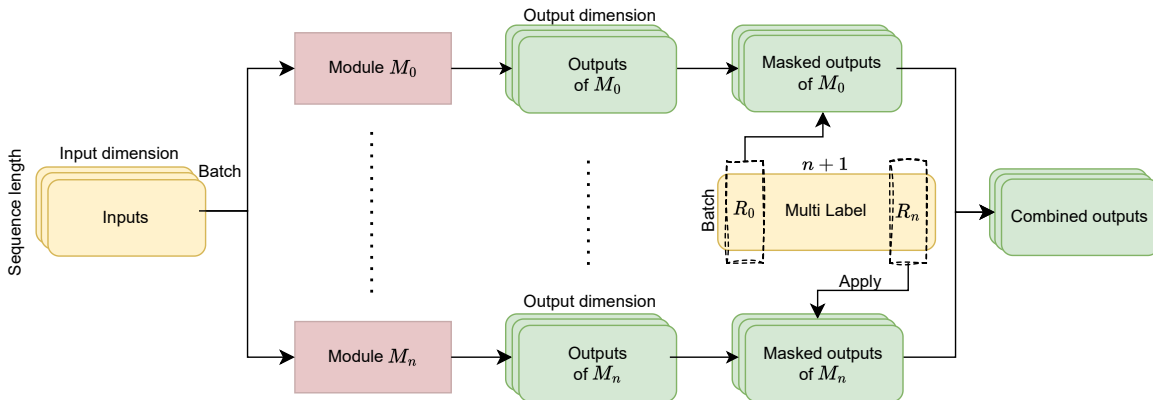
Figure 2: Architecture for a generalized masking model. If each module M_i represents an attention head (or a group of attention heads), then we get a masking model. If the modules M_i represent the complete transformers, we obtain an ensemble model. In both cases the combined outputs is a result of application of the linear transformation (trainable or constant) to the stacked masked outputs.

mula (4) and train a model to generate elements from it. Although we can not use this method in our comparison, since it was trained only on trivial words (see terminology of Remark 3.3), but we can employ its metrics in order to gain new insights about our primary methods. Furthermore, negative baseline can be used for so-called *negative knowledge distillation* (Gou et al., 2021): fine-tuning one of the primary models to predict the outputs that are **not** predicted by the negative baseline.

5. Evaluation

Before discussing the results of the evaluation of the proposed models, we explain the relevant metrics and datasets, with other technical implementations details given in Appendix C.

Metrics There are families of metrics commonly used in natural language processing, which, following (Celikyilmaz et al., 2020), may be divided in three groups: human-centric evaluation metric, automatic metrics and machine-learned metrics. We decide to concentrate our attention on automatic metrics, since our primary objective is to understand the quantitative performance of neural-based algorithms in a task of generating the words from $\cap_i R_i$.

Our main metric, the *completion ratio*, is an average number of the successfully generated words per batch. Specifically, for a batch of randomly generated words prefixes, the model generates a set of possible outputs based on each prefix. The number of generated outputs that simultaneously belong to all normal closures R_i is then computed, and this number is divided by the size of the batch. A higher completion ratio signifies a greater proficiency of the model in generating desired outputs.

Random	Evo	Greedy	Ignore	Mask	Prompt
0.058	0.035	0.479	**0.36**	0.16	0.30
0.032	0	0.016	0.356	0.584	**0.684**
0.025	0	0.004	0.263	0.487	**0.550**

Table 3: Completion ratio of developed algorithms for various number of generators $n = 3,\ 4,\ 5$.

In addition to CE loss (5), we also utilize a *reduction ratio* for guiding the training process. This metric measures the distance (see Section 4.2) from the generated output to the given R_i divided by the length of the input, batch averaged and averaged over all i.

Reduction ratio sits "in between" the loss and the final completion ratio metric: once the loss on the validation dataset has stabilized, and the completion ratio remains relatively low, the changing rate of the reduction ratio becomes informative. This changing rate indicates how the model's output gradually approaches the numerator of Wu's formula (see Figure 3).

Remark 5.1. Regarding the metric for evaluating generators of the *non-trivial* words in $R_0 \cap \cdots \cap R_n$, it appears that human evaluation remains challenging due to our limited understanding of the intricate structure of the homotopy groups. The problem is exacerbated by the relatively limited number of mathematicians deeply immersed in this topic as well. On the other hand, the machine-learned metrics (for instance, probability of the word being contained in the symmetric commutator subgroup) may be beneficial and play significant role.

Datasets In order to maximize the performance of our models, specifically in terms of the number of generated

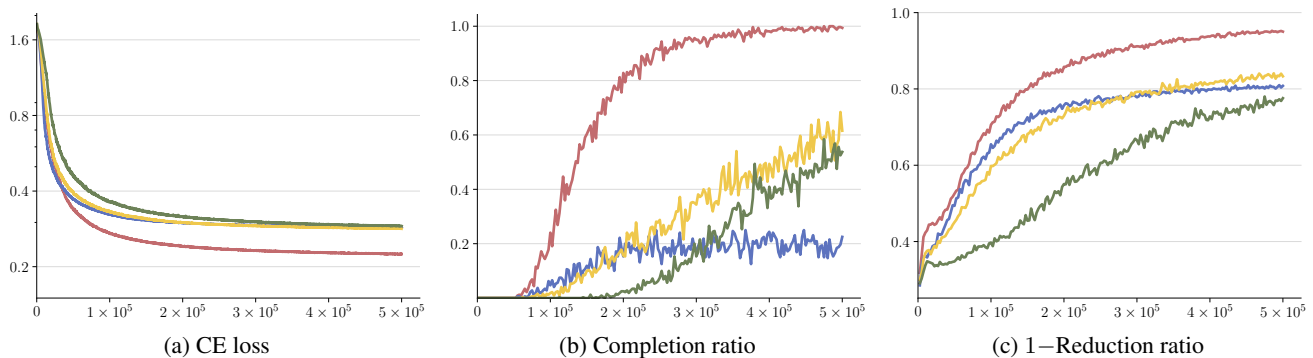(a) CE loss   (b) Completion ratio   (c) $1-$Reduction ratio

Figure 3: Illustration of the learning process for *negative baseline* (red), *prompt* (yellow), *masking* (green) and *ignore* (blue) methods, $n = 4$.

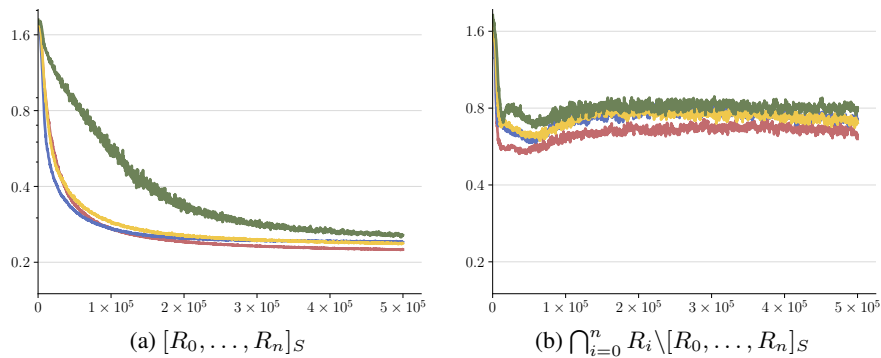(a) $[R_0, \ldots, R_n]_S$ (b) $\bigcap_{i=0}^{n} R_i \setminus [R_0, \ldots, R_n]_S$

Figure 4: Comparison of CE loss per epoch on validation for the trivial and non-trivial words, $n = 4$.

words from random prefixes, we adopt an approach where overfitting is not a primary concern. To achieve this, we employ an infinite, online generated training dataset that encompasses words from $[R_0, \ldots, \hat{R}_i, \ldots, R_n]_S$ for all i, i.e. the multi-labels associated with these words contain only a single zero. The validation dataset is also generated in an online fashion, drawing samples from the same distribution. For a negative baseline model, described in Section 4.3.1, both training and validation dataset consist of words from the full symmetric commutator subgroup $[R_0, \ldots, R_n]_S$.

For an auxilary tests, we have assembled small datasets consisting of cyclic permutations of generators of $\pi_n S^2$ for $n = 3, 4$ from Table 1. By evaluating our metrics on these cyclic permutations, we can assess the models proficiency in generating *non-trivial* words.

5.1. Results

We compare our deep learning approaches with non-neural-based baselines using completion ratio, see Table 3. We also report the metric values on validation during training progress (Figure 3), to illustrate the difference between various multi-label approaches. We report our results for $n \leq 5$ generators, due to the available computation resources. We also notice (see discussion below), that for a higher number

of generators the high length of words from the full intersection $\cap_i R_i$ suggests to use another, more sophisticated representation of words.

Table 3 provides compelling evidence of the scalability and superior performance achieved by language models when compared to the baselines. These results signify a promising avenue for future applications of such models in understanding simplicial homotopy groups. In terms of comparing the performance of the methods between each other, the prompt-based approach outperformed the other methods on the validation dataset. Finally, Figure 4 illustrates that the non-trivial words are significantly more challenging (as expected) for the models to generate.

6. Conclusion

We evaluate perspectives of using machine lerning in generating cycles in simplicial groups. From a machine learning perspective, the problem can be reformulated as the task of generating elements from the intersection of subsets of (a form of) Dyck-n words. Some classical algorithms are implemented, some are developed from scrath and used as baselines for deep learning algorithms. Built of the idea of ensemble of generators, the additional multi-label information is added to the training dataset, which allows the single

model to work as a generalization of an ensemble. Resulting models, in contrast to baselines, are scalable, and will serve as a building blocks for future algorithms specialized in sampling from homotopy groups of spaces.

Further Directions Employing a more sophisticated tokenization scheme could lead to a more compact representations of elements in higher simplicial homotopy groups. One of the choices would be tokenizing commutator brackets and delimiters to encode commutator presentation of the words from $\cap_i R_i$. However, in the present work we choose the generators of free group as tokens for its simplicity. Even more compact, but at the same time more elaborate and theoretically involved representation is given by Λ-*algebra* (Bousfield et al., 1966). Our ongoing research is devoted to employ machine learning in building connections between cycles in simplicial groups and elements in Λ-algebra.

The ultimate goal of our research is to develop a method capable of sampling elements from simplicial homotopy groups. It is crucial for such method to distinguish between the intersection of R_i and the symmetric commutator subgroup. The present work focuses solely on simplicial cycles, without considering simplicial boundaries. However, we believe that the developed deep learning approach will expand the researchers' toolkits and bring this goal closer to fruition.

Impact Statement

This paper presents work whose goal is to advance the application of machine learning in mathematics. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgement

References

Adams, J. F. On the structure and applications of the steenrod algebra. *Commentarii Mathematici Helvetici*, 32:180–214, 1958.

Anick, D. On the computational complexity of rational homotopy. In *Homotopy Theory: Proceedings of the Durham Symposium 1985*, volume 117, pp. 1. Cambridge University Press, 1987.

Appel, K. and Haken, W. The solution of the four-color-map problem. *Scientific American*, 237(4):108–121, 1977.

Arone, G. and Mahowald, M. The goodwillie tower of the identity functor and the unstable periodic homotopy of spheres. *Inventiones mathematicae*, 135:743–788, 1999.

Atkinson, M. and Sack, J.-R. Generating binary trees at random. *Information Processing Letters*, 41(1):21–23, 1992. ISSN 0020-0190. doi: https://doi.org/10.1016/0020-0190(92)90075-7. URL https://www.sciencedirect.com/science/article/pii/0020019092900757.

Bartholdi, L., Fialkovski, D., and Ivanov, S. O. On commutator length in free groups. *arXiv preprint arXiv:1504.04261*, 2015.

Behrens, M. *The Goodwillie tower and the EHP sequence*, volume 218. American Mathematical Society, 2012.

Bengio, Y. Neural net language models. *Scholarpedia J.*, 3(1):3881, 2008.

Bousfield, A. K., Curtis, E. B., Kan, D. M., Quillen, D. G., Rector, D. L., and Schlesinger, J. W. The mod-p lower central series and the adams spectral sequence. *Topology*, 5(4):331–342, 1966.

Brown, E. H. Finite computability of postnikov complexes. *Annals of Mathematics*, 65(1):1–20, 1957. ISSN 0003486X. URL http://www.jstor.org/stable/1969664.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020.

Cadek, M., Krcal, M., Matousek, J., Vokrinek, L., and Wagner, U. Polynomial-time computation of homotopy groups and postnikov systems in fixed dimension, 2014.

Cao, W., Zhou, C., Wu, Y., Ming, Z., Xu, Z., and Zhang, J. Research progress of zero-shot learning beyond computer vision. In *Algorithms and Architectures for Parallel Processing: 20th International Conference, ICA3PP 2020, New York City, NY, USA, October 2–4, 2020, Proceedings, Part II 20*, pp. 538–551. Springer, 2020.

Celikyilmaz, A., Clark, E., and Gao, J. Evaluation of text generation: A survey. June 2020.

Curtis, E. B. Simplicial homotopy theory. *Advances in Mathematics*, 6(2):107–209, 1971.

Davies, A., Veličković, P., Buesing, L., and et al. Advancing mathematics by guiding human intuition with ai. *Nature*, 600:70–74, 2021.

Delétang, G., Ruoss, A., Grau-Moya, J., Genewein, T., Wenliang, L. K., Catt, E., Cundy, C., Hutter, M., Legg, S., Veness, J., et al. Neural networks and the chomsky hierarchy. *arXiv preprint arXiv:2207.02098*, 2022.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

Droste, S., Jansen, T., and Wegener, I. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1-2):51–81, 2002.

Ebrahimi, J., Gelda, D., and Zhang, W. How can self-attention networks recognize dyck-n languages?, 2020.

Eilenberg, S. and Zilber, J. A. Semi-simplicial complexes and singular homology. *Annals of Mathematics*, pp. 499–513, 1950.

Filakovsky, M., Franek, P., Wanger, U., and Zhechev, S. Computing simplicial representatives of homotopy group elements. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1135–1151. SIAM, 2018.

Gabriel, P. and Zisman, M. *Calculus of fractions and homotopy theory*, volume 35. Springer Science & Business Media, 2012.

Gou, J., Yu, B., Maybank, S. J., and Tao, D. Knowledge distillation: A survey. *Int. J. Comput. Vis.*, 129(6):1789–1819, June 2021.

Hahn, M. Theoretical limitations of self-attention in neural sequence models. *Trans. Assoc. Comput. Linguist.*, 8:156–171, December 2020.

Hales, T. A proof of the kepler conjecture. *Annals of the mathematics*, pp. 1065 – 1185, 2005.

Hall, M. A basis for free lie rings and higher commutators in free groups. *Proceedings of the American Mathematical Society*, 1 (5):575–581, 1950.

Han, J. M., Rute, J., Wu, Y., Ayers, E. W., and Polu, S. Proof artifact co-training for theorem proving with language models. *arXiv preprint arXiv:2102.06203*, 2021.

He, Y.-H. Machine-learning mathematical structures, 2021.

Hilton, P. A brief, subjective history of homology and homotopy theory in this century. *Mathematics magazine*, 61(5):282–291, 1988.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. The curious case of neural text degeneration, 2020.

Jabbar, A., Li, X., and Omar, B. A survey on generative adversarial networks: Variants, applications, and training, 2020.

Jurafsky, D. and Martin, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, USA, 1st edition, 2000. ISBN 0130950696.

Kargapolov, M. I. and Merzljakov, J. I. *Fundamentals of the Theory of Groups*, volume 62. Springer, 1979.

Keskar, N. S., McCann, B., Varshney, L. R., Xiong, C., and Socher, R. CTRL: A conditional transformer language model for controllable generation. September 2019.

Killoran, N., Lee, L. J., Delong, A., Duvenaud, D., and Frey, B. J. Generating and designing dna with deep generative models. *arXiv preprint arXiv:1712.06148*, 2017.

Lample, G. and Charton, F. Deep learning for symbolic mathematics, 2019.

Li, J., Tang, T., Zhao, W. X., Nie, J.-Y., and Wen, J.-R. Pretrained language models for text generation: A survey, 2022.

Linder, J. and Seelig, G. Fast activation maximization for molecular sequence design. *BMC Bioinformatics*, 22(1):510, October 2021.

Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. Pretrain, prompt, and predict: A systematic survey of prompting methods in natural language processing, 2021.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization, 2019.

M., C. *Pattern Recognition and Machine Learning*. Information Science and Statistics. Springer, New York, NY, 1 edition, August 2006.

Mac Lane, S. *Categories for the working mathematician*, volume 5. Springer Science & Business Media, 2013.

Mikhailov, R. Homotopy patterns in group theory, 2021. URL https://arxiv.org/abs/2111.00737.

Nguyen, A., Yosinski, J., and Clune, J. Understanding neural networks via feature visualization: A survey, 2019.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Peifer, D., Stillman, M., and Halpern-Leistner, D. Learning selection strategies in buchberger's algorithm, 2020.

Polu, S., Han, J. M., Zheng, K., Baksys, M., Babuschkin, I., and Sutskever, I. Formal mathematics statement curriculum learning. *arXiv preprint arXiv:2202.01344*, 2022.

Quillen, D. G. *Homotopical algebra*, volume 43. Springer, 2006.

Radford, A. and Narasimhan, K. Improving language understanding by generative pre-training. 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language models are unsupervised multitask learners. 2019.

Romero, A., Rubio, J., and Sergeraert, F. Computing spectral sequences. *Journal of symbolic computation*, 41(10):1059–1079, 2006.

Rubio, J. and Sergeraert, F. Constructive algebraic topology. *Bulletin des Sciences Mathématiques*, 126(5):389–412, 2002. ISSN 0007-4497. doi: https://doi.org/10.1016/S0007-4497(02)01119-3. URL https://www.sciencedirect.com/science/article/pii/S0007449702011193.

Sarkar, R. Low distortion delaunay embedding of trees in hyperbolic plane. In *Graph Drawing: 19th International Symposium, GD 2011, Eindhoven, The Netherlands, September 21-23, 2011, Revised Selected Papers 19*, pp. 355–366. Springer, 2012.

Sergeraert, F. The computability problem in algebraic topology. *Advances in Mathematics*, 104(1):1–29, 1994.

Taubin, G. Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(11):1115–1138, 1991.

Tillmann, C. and Ney, H. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Comput. Linguist. Assoc. Comput. Linguist.*, 29(1):97–133, March 2003.

Toda, H. *Compositional methods in homotopy groups of spheres. (AM-49).* Annals of Mathematics Studies. Princeton University Press, Princeton, NJ, January 1963.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need, 2017.

Wagner, A. Z. Constructions in combinatorics via neural networks, 2021.

Wang, W., Zheng, V. W., Yu, H., and Miao, C. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10 (2):1–37, 2019.

Weiss, G., Goldberg, Y., and Yahav, E. On the practical computational power of finite precision rnns for language recognition, 2018.

Weiss, G., Goldberg, Y., and Yahav, E. Thinking like transformers. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 11080–11090. PMLR, 18–24 Jul 2021. URL https://proceedings. mlr.press/v139/weiss21a.html.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38– 45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https: //aclanthology.org/2020.emnlp-demos.6.

Wu, J. Combinatorial descriptions of homotopy groups of certain spaces. *Mathematical Proceedings of the Cambridge Philosophical Society*, 130(3):489–513, 2001. doi: 10.1017/ S030500410100487X.

Yao, Z., Zhang, B., Wang, Z., Ouyang, W., Xu, D., and Feng, D. Intersectgan: Learning domain intersection for generating images with multiple attributes. pp. 1842–1850, 10 2019. ISBN 978-1-4503-6889-6. doi: 10.1145/3343031.3350908.

Yu, L., Zhang, W., Wang, J., and Yu, Y. Seqgan: Sequence generative adversarial nets with policy gradient, 2017.

Zubkov, A. N. On a matrix representation of a free group. *Mathematical Notes*, 64:745–752, 1998.

A. Background on Group Theory

As a reminder, we will present a series of basic definitions from group theory. These definitions can serve as points of reference from the main body of the paper.

Definition A.1. A *group* G is a set equipped with a binary operation $\mu : G \times G \to G$, usually denoted just by $(g_1, g_2) \mapsto g_1 g_2$, satisfying the following axioms:

- **Associativity**: For all $g_1, g_2, g_3 \in G$, $(g_1 g_2) g_3 = g_1 (g_2 g_3)$.

- **Identity Element**: There exists an element $1 \in G$, called the *identity element*, such that for all $g \in G$, $g1 = 1g = g$.

- **Inverse Element**: For every element $g \in G$, there exists an element $g^{-1} \in G$, called the *inverse element* of g, such that $gg^{-1} = g^{-1}g = 1$.

An arbitrary group can be expressed as a quotient (Definition A.6) of a free group.

Definition A.2. Let X be a set. The *free group* F on X is a group defined as follows: F consists of all reduced words (finite sequences) on elements of S and their formal inverses, equipped with the operation of concatenation followed by reduction. The reduction operation eliminates pairs of the form xx^{-1} and $x^{-1}x$. The identity element of F is the empty word.

Free groups, in comparison with arbitrary groups, have a simple algebraic structure, so as their *subgroups*, i.e. subsets closed under group operations.

Theorem A.3 (Nielsen–Schreier theorem, (Kargapolov & Merzljakov, 1979)[Section 14.3]). *Every subgroup of a free group is free.*

Subgroups R_i of Definition 1, that we are dealing with, are defined in a way that they are closed under conjugation operation.

Definition A.4. Let G be a group. A subgroup N of G is called a *normal subgroup* if and only if for every $g \in G$ and $n \in N$, the element $n^g = g^{-1}ng$ is also in N.

Definition A.5. Let G be a group and S be a subset of G. The *normal closure* of S in G, denoted by $\langle S \rangle^G$, is the smallest normal subgroup of G that contains S. It is the intersection of all normal subgroups of G containing S.

Wu's formula (4) involves a quotient of an intersection of R_i's by their symmetric commutator subgroup.

Definition A.6. Let G be a group and N be a normal subgroup of G. The *quotient group* of G by N, denoted by G/N, is the set of *cosets* of N in G, i.e. subsets of G of the form $gN = \{gn | n \in N\}$, with group operations induced from G.

Finally, some of our methods rely on the presentations of R_i as kernels of *homomorphisms* d_i, i.e. maps between groups that preserve group operations, see also Formula (8).

Definition A.7. A *kernel* of a group homomorphism $f : G \to H$ is the following subset:

$$\ker f = \{g \in G | f(g) = 1\}.$$

Note that the kernel of a group homomorphism $G \to H$ is always a normal subgroup of G.

Throughout the paper, we utilize the notion of commutators and conjugations, which were introduced in Section 3, to provide a shorter representation of words in a free group. There is a number of *identities*, involving commutatrors, that hold for all elements of any group, for instance:

$$[x, zy] = [x, y][x, z][x, z, y]$$
$$[xz, y] = [x, y][x, y, z][z, y]$$
$$[x, y^{-1}, z]^y [y, z^{-1}, x]^z [z, x^{-1}, y]^x = 1.$$

The last identity, called the *Hall-Witt identity* is a "non-abelian" version of the Jacobi identity in Lie algebras. As mentioned at the end of Section 4.1, these identities enable us to express every commutator as a product of iterated commutators **of the same length**. For example, for commutator $[[a, b], [c, d]]$, the Hall-Witt identity reads:

$$[a, b, [c, d]]^{b^{-1}} [b^{-1}, [d, c], a]^{[c,d]} [c, d, a^{-1}, b^{-1}]^a = 1$$

and since

$$[b^{-1}, [d, c], a] = [d, c, b^{-1}, a^{[b^{-1}, [d,c]]}]^{-1},$$

we get

$$[[a, b], [c, d]] = [c, d, a^{-1}, b^{-1}]^{-ab} [d, c, b^{-1}, a^{[b^{-1}, [d,c]]}]^{[c,d]b}.$$

B. Background on Simplicial Homotopy Theory

In this section we will provide a brief overview of the historical development of the concept of homotopy groups (see also (Hilton, 1988)). Subsequently, we will delve into the basics of simplicial homotopy theory and the description of Wu's formula within the context of simplicial groups.

B.1. History Overview

As a distinguished area of mathematics, algebraic topology was surfaced since the end of XIX century. During this time, mathematicians realized that certain important properties of geometric objects, like properties of vector fields on a manifolds or solutions of differential equations on them are controlled by *topological* properties of the underlying objects of study, rather than geometric or analytic characteristics. This realization prompted the need for algebraic invariants that could discern between classes of spaces with similar topological properties and aid in their classification. Historically, one of the first invariant was *homology* H_* of space, shortly followed by a fundamental group π_1.

Such invariants were able to distinguish spaces *up to homotopy equivalence*. Specifically, two maps $f, g : X \to Y$ are called homotopic, if there is a family of maps $h_t : X \to Y$, $t \in [0, 1]$, continuous in t, which "connects" f to g in a sense that $h_0 = f$, $h_1 = g$. Similarly, two spaces X, Y are *homotopy equivalent* if there exist maps $X \leftrightarrows Y$, which compositions with each other are homotopic to identities.

Generalizing π_1, the series of higher invariants π_*, called the *homotopy groups*, were defined as homotopy classes of maps from higher dimensional spheres S^n to a space of interest. Stronger than homology, homotopy groups capture the substantial part of a homotopy type of space, which we nowadays call a *weak homotopy type*. Extremely difficult to compute and comprehend, till today, homotopy groups remain the main object of interest in algebraic topology. Various flavors of homotopy theory have been developed, including stable homotopy, v_n-periodic homotopy and others, each with corresponding homotopy groups that are more tractable and accessible than the classical ones.

In parallel with the development of various algebraic invariants for classifying spaces, the concept of space itself has undergone several formalizations within homotopy theory. Initially, the notion of a space referred to a *topological space*, which consisted of a set equipped with a chosen collection of subsets called a *topology*. Later the concept of an (abstract) simplicial complex emerged as a combinatorial notion of space, suitable for defining homology groups. This notion was subsequently refined to that of a simplicial set.

Homotopy theory of simplicial sets, which is both combinatorial and algebraic in nature, now serves as the foundation for more advanced concepts in abstract homotopy theory, such as simplicial model categories, quasicategories, simplicial localizations, and others. Within the scope of the current paper, a specific type of simplicial sets called *simplicial groups* holds particular significance. Simplicial groups serve as a crucial link between simplicial homotopy theory and group theory, offering a concrete, algebraic, and computational-friendly framework.

B.2. Simplicial Groups and Wu's Formula

We will now describe the relevant part of simplicial homotopy theory in greater detail. Simplicial set (formally described as a functor $\Delta^{op} \to \text{Set}$) can be visualized (Mac Lane, 2013)[Ch. 7.5] as a diagram of sets

$$X : \qquad \pi_0 X \xleftarrow{d_0} X_0 \underset{\underset{d_1}{\xleftarrow{\hspace{1em}}}}{\overset{\overset{d_0}{\xleftarrow{\hspace{1em}}}}{\xrightarrow{s_0}}} X_1 \underset{\underset{d_2}{\xleftarrow{\hspace{1em}}}}{\overset{\overset{d_0}{\xleftarrow{\hspace{1em}}}}{\underset{\underset{s_1}{\xrightarrow{\hspace{1em}}}}{\overset{\overset{s_0}{\xrightarrow{\hspace{1em}}}}{\xleftarrow{d_1}}}}} X_2 \qquad \ldots,$$

with maps d_i, called *faces* and s_j, called *degeneracies* which satisfy the *simplicial identities*:

$$d_i d_j = d_{j-1} d_i, \ i < j$$
$$s_i s_j = s_j s_{i-1}, \ i > j$$
$$d_i s_j = \begin{cases} s_{j-1} d_i, \ i < j \\ \text{id}, \ i \in \{j, j+1\} \\ s_j d_{i-1}, \ i > j+1 \end{cases} \qquad . \tag{6}$$

Drawing intuition from simplicial complexes, each X_n can be thought of as a set of n-simplices, with maps $d_i : X_n \to X_{n-1}$ indicate an i-th face of an n-simplex and maps $s_j : X_{n-1} \to X_n$ indicate ways how to consider $(n-1)$-simplex as a (degenerate) n-simplex.

Simplicial sets were introduced by Eilenberg and Zilber (Eilenberg & Zilber, 1950) (originally under the name of semi-simplicial complexes) for the needs of formalization of homology theory. When the abstract homotopy theory became more articulated in the fundamental works of Gabriel and Zisman (Gabriel & Zisman, 2012) and Quillen (Quillen, 2006), it was shown that the corresponding homotopy theory is equivalent to the classical homotopy theory of topological spaces. Although for some concrete work in homotopy theory simplicial sets may better suited than topological spaces/cell complexes, regarding computations of homotopy groups, bare simplicial sets have some difficulties when it comes to computation of homotopy groups. For instance, the naive notion of homotopy is not an equivalence relation for all simplicial sets.

13

One of the possible solutions is to consider *simplicial groups*, i.e. simplicial sets which have a group structure on the sets of their simplices, such that the maps d_i, s_j preserve this structure. A notable advantage of simplicial groups is the very explicit description of their homotopy groups: if G is a simplicial group, then

$$\pi_n G = \frac{\cap_{i=0}^n \ker d_i}{\operatorname{im} \partial_{n+1}}, \tag{7}$$

where ∂_{n+1} is a restriction of d_{n+1} to the intersection of kernels of the next dimension. So the homotopy groups of G are identified with the quotients of intersections of kernels of face maps d_i by the images of the last face maps (restricted to a suitable intersection).

To get the Wu's formula for $\pi_n S^2$ from (7), we will consider a particular simlicial group (more precisely, the functor $s\text{Set} \to s\text{Gr}$) called the *Milnor construction* (Curtis, 1971)[Ch. 4]. Let X be a connected simplicial set with a basepoint $*$. The Milnor construction for X, denoted by $F[X]$ is a simplicial group obtained by level-by-level-wise application of the free group functor to X, quotient by the relation $* = 1$. This means that n-simplices in Milnor's construction $F[X]$ form free group on n-simplices X_n with the identity of the group identified with a basepoint $*$. The reason for introducing this construction is that for a connected simplicial set X the Milnor's construction $F[X]$ has a homotopy type of loops over the suspension of X. For $X = S^1$ on the level of homotopy groups it means that

$$\pi_n F[S^1] = \pi_{n+1} S^2.$$

The simplicial circle S^1 consists of one non-degenerate 0-simplex, which is a basepoint $*$ and one non-degenerate 1-simplex σ (plus the degeneracy $s_0(*)$), and there are no non-degenerate simplices in all higher dimensions.
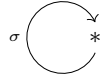
Now simplicial identities (6) show that

$$F[S^1]_n = F[\{x_0, \ldots, x_{n-1}\}]$$

and faces are given by

$$d_0(x_j) = \begin{cases} 1, & j = 0 \\ x_j, & j > 0 \end{cases}$$

$$d_i(x_j) = \begin{cases} x_j, & j < i \\ x_{i-1} x_i, & j \in \{i-1, i\} \\ x_{j-1}, & j > i \end{cases}$$

$$d_n(x_j) = \begin{cases} x_j, & j < n-1 \\ 1, & j = n-1 \end{cases}.$$

It is shown in (Wu, 2001) that the kernels $\ker d_i$, after an appropriate change of basis and relabeling, can be identified with normal subgroups

$$\ker d_0 = R_0 = \langle x_1 \ldots x_n \rangle^F \quad \ker d_i = R_i = \langle x_i \rangle^F, i > 0, \tag{8}$$

and the image in the denominator of Formula (7) with their *symmetric commutator subgroup*

$$\operatorname{im} \partial_{n+1} = [R_0, \ldots, R_n]_S = \prod_{\sigma \in \Sigma_{n+1}} [R_{\sigma(0)}, \ldots, R_{\sigma(n)}].$$

This is the idea behind the Wu's formula.

C. Implementation Details

To operate with words in free groups we implemented a small `Python` module `freegroup`[4]. In this module we use lists of integers as the representation of words with the following mapping: $x_k \mapsto k, x_k^{-1} \mapsto -k$.

---

[4]Code available at `https://github.com/ml-in-algebraic-topology/freegroup`

As a base for our models we used GPT2 (Radford et al., 2019) implemented by `huggingface` (Wolf et al., 2020). For $n = 3$ we used the model with $\sim 2 \cdot 10^6$ parameters and for $n = 4, 5$ we used $\sim 50 \cdot 10^6$ parameters. We employed standard training framework with iterating through available data and gathering examples to batches. We trained models for $2 \cdot 10^5, 5 \cdot 10^5$ iterations with batches of sizes $64, 16$ for $n = 3$ and $n = 4, 5$ cases, respectively. For the optimization we used AdamW (Loshchilov & Hutter, 2019) implemented by `PyTorch` (Paszke et al., 2017) with learning rates: 10^{-4} for $n = 3$ and 10^{-6} for $n = 4, 5$. For the inference we used generation algorithms implemented by `huggingface`. During sampling `top_p = 0.9`; during beam search `num_beams = 5`, `repetition_penalty = 1.2`.

Dataset parameters are the following:

- maximal length of a word $= 200, 400, 600$,

- maximal lengths of a word from R_0 are $10, 9, 8$ for $n = 3, 4, 5$ respectively,

- maximal lengths of a word from R_i, $i > 0$ are $30, 30, 30$ for $n = 3, 4, 5$ respectively.

D. Additional Experiments, Models and Examples

D.1. Other Models

In addition to the deep learning methods discussed in the main body of the paper, we explored several other trainable and non-trainable machine learning approaches for word generation.

Fixed Embeddings Alongside the development of deep learning models, which are in fact suitable deep feature extractors coupled with a classifiers, we also investigated the usefulness of non-trainable embeddings and experimented with different optimizers and machine learning classifiers applied on top of them. The underlying motivation for this investigation was to transform the problem setting from a discrete one to a continuous one, as most optimization and machine learning methods operate more effectively in continuous domains.

The variant of the non-trainable embeddings that we investigated is based on the matrix representation of free group. Using the group homomorphism $\rho : F_2 \to SL_2(\mathbb{R})$, called *Sanov representation* (Zubkov, 1998)

$$x \mapsto \begin{pmatrix} 1 & 2 \\ 0 & 1 \end{pmatrix}, \, y \mapsto \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}, \tag{9}$$

the free group F_2 on two generators x and y can be embedded in a space of two-by-two matrices with real coefficients (in fact, into a subgroup of matrices M with $\det M = 1$)

The embedding of a free group F on an arbitary number of generators n is obtained by inclusion

$$F \to F_2, \, x_i \mapsto [x, y^i].$$

Remark D.1. Note that for a word $w \in F$ the magnitudes of coefficients in the matrix $\rho(w)$ depends exponentially on the length of w, making this embedding practically unfeasible. Nevertheless we tested an optimization on this embedding space to evaluate a potential of non-trainable embeddings on a small scale.

One of the advantages of matrix embeddings is that the homomorphisms $d_i : F \to F$, such that $\ker d_i = R_i$, as in Formula (8) can be reinterpreted as a matrix operation of nullifying i-th column denoted f_i, $i > 0$. At the same time, since ρ preserves the group operation, verifying whether $w \in R_i$ is equivalent to checking if $f_i \rho(w)$ is equal to the identity matrix I.

We investigated various optimization procedures on this feature space, however, regrettably, the method proved ineffective in generating any words for $n \geq 3$, even when considering the partial intersection $\cap_{i=1}^n R_i$.

Remark D.2. We have also considered posibility of hyperbolic embedding, i.e. embeddings of the (compact region of) Cayley graph of F into hyperbolic plane $\mathbb{H}$. We considered embedding by Sarkar algorithm (Sarkar, 2012) or as an orbit of i under $\rho : F \hookrightarrow SL_2(\mathbb{R})$ acting on the upper half-plane $\mathbb{C}_{\text{Im} \geq 0}$ by Mobius transformations.

Both embeddings, although not isometric, retain certain metric properties of F, where the metric structure of F is determined by the *word metric*. However, during our preliminary investigations, we discovered that the metric structure of F might not be the most suitable setting for optimization in our specific problem.

Activation Maximization While experimenting with neural based classifiers for elements in R_i, we investigated the feasibility of methods from the area of interpretable neural networks. Technique that we were studying, described in (Nguyen et al., 2019), involves maximizing the activation of specific neurons in a neural network to generate images that are most likely to activate those neurons. We tried to adopt this method to the free group's word generation as it was done for protein sequences in (Linder & Seelig, 2021).

Following (Killoran et al., 2017), (Linder & Seelig, 2021), we implemented the following approach. If the classifier input is given by sequence of length M over the alphabet of size N, then the latent variable z of size $N \times M$ is representing the probabilities of each token

on each position. Suppose that classifier is trained up to a sufficiently high accuracy, then sampling with the probabilities from z and feeding back the resulting words to the classifier, the relaxation procedure can be carried out in the latent space of z.

The appeal of this approach comes from the fact that it can be extended to an ensemble of classifiers. By maximizing the activation of all classifiers within the ensemble simultaneously using the same latent variable z, the approach can be used to generate words from the intersection $\cap_i R_i$ that maximize the activation of output neurons of all classifiers in the ensemble.

The appeal of this approach lies in its potential for extension to an ensemble of classifiers. By maximizing the activation of all classifiers within the ensemble simultaneously using the same latent variable z, the approach can be effectively applied across the entire ensemble.

In the end, the activation maximization method turned out to be incapable of generating at even short non-trivial words from a single R_i. Despite the continuous effort, we were unable to generate words from the intersection using the joint activation maximization of multiple generators.

SeqGAN   Another method that has been explored is generative adversarial networks (GANs) (Jabbar et al., 2020), specifically the SeqGAN (Yu et al., 2017) approach. GANs are a type of deep neural network that can generate realistic synthetic data, such as images or text. SeqGAN focuses on generating realistic sequences of data, such as sentences or musical compositions. However, while GANs have shown promising results in some applications (Jabbar et al., 2020) and can be used in similar circumstances (Yao et al., 2019), they did not yield promising results in our problem.

LSTM Ensemble   Before developing the general masking approach, based on multi-labels, as mentioned in Remark 4.3, we investigated its particular realization based on the ensemble of deep models. Our initial experiments were based on an ensemble of LSTM cells (Hochreiter & Schmidhuber, 1997), in which each generator G_j is trained on the subgroup R_j or some symmetric commutator subgroup.

In generating the word from the intersection, to sample the next token from the ensemble we used a *temperature sampling*, i.e. the probability of the next token y_k from a sequence y have a distribution

$$y_k \sim \text{Softmax}\left(\frac{1}{\tau}\sum_{i=1}^{N} w_i f_{\theta_i}(y)\right),$$

with weights w_i for each generator f_{θ_i}.

This approach allowed us to get first non-trivial elements from the full intersection. However, it was subsequently surpassed by a more general masking approach, which has also been demonstrated to be more scalable.

D.2. Training Dataset Properties

We used datasets that were generated online during the training process. For all of the experiments we used the same seed, 42. The tag version of the `freegroup` library we used is `icml2024-submission`. Here are the configurations for different cases of n, the dimension of the free group:

$n = 3$ Maximum depth of words from R_0 was equal to 10, maximum depth of words from R_i for $i > 0$ was equal to 30. Maximum number of arbitrary words from free group to increase the *noise* in data was equal to 1 and the length of this word at maximum 10. The support of lengths was from 1 to 200.

$n = 4$ Maximum depth of words from R_0 was equal to 10, maximum depth of words from R_i for $i > 0$ was equal to 30. Maximum number of arbitrary words from free group to increase the *noise* in data was equal to 1 and the length of this word at maximum 10. The support of lengths was from 1 to 400.

$n = 5$ Maximum depth of words from R_0 was equal to 7, maximum depth of words from R_i for $i > 0$ was equal to 30. Maximum number of arbitrary words from free group to increase the *noise* in data was equal to 1 and the length of this word at maximum 10. The support of lengths was from 1 to 400.

Moreover, we present the statistical properties of the datasets for $n = 3, 4, 5$ in Figure 5.

D.3. Translator Model

For a given word $w \in [F, F]$ there are multiple way to present it in form of a product of commutators. Although there are algorithms to find such presentations (Hall, 1950), (Bartholdi et al., 2015), the presentations of long words in terms of basic commutators or as a product of ordinary (not iterated) commutators are bulk and practically unusable. We are looking for short and concise presentations, such as given in Table 1.

To tackle this challenge, we employed an encoder-decoder model (Vaswani et al., 2017) as a translator from a reduced word presentation to the commutator presentation.

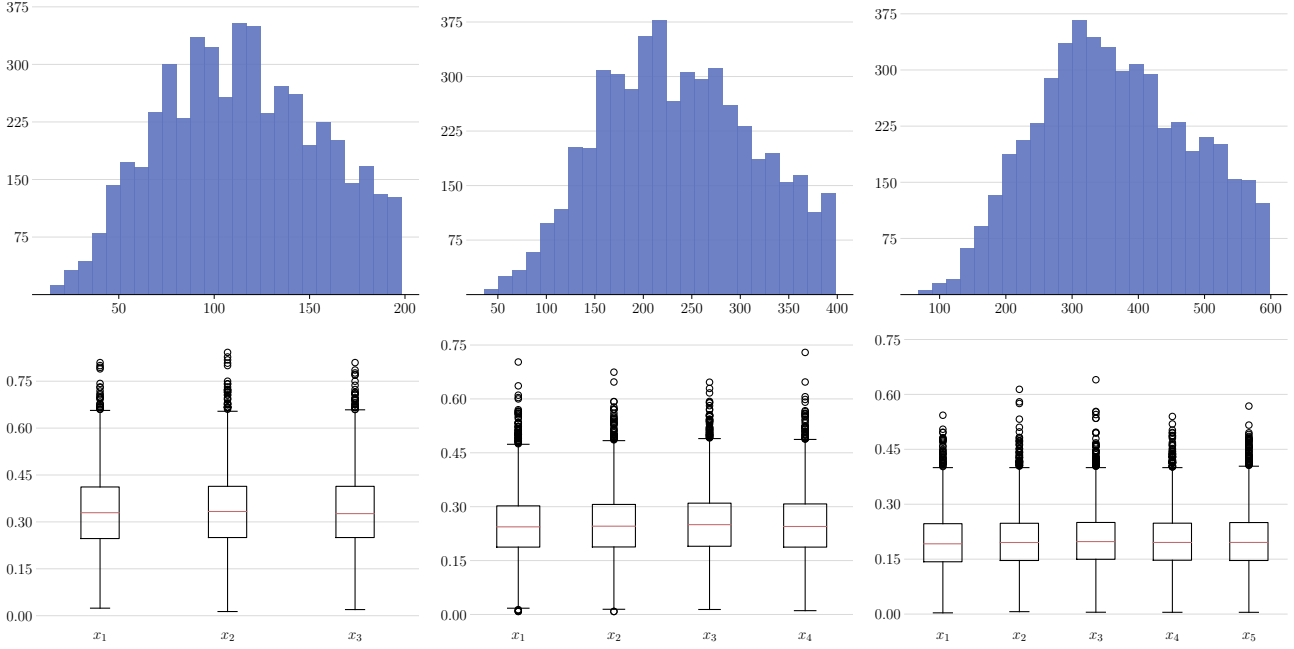The following procedure was implemented to generate the training and validation datasets:

Figure 5: Length distribution (top) and occurrence ratio of generators x_i (bottom) in the training dataset subsample for the $n = 3, 4, 5$.

1. A collection of random binary trees with a random number of leaves is generated.

2. The leaves of these trees are replaced with random words from a free group, ensuring that the word length remained within predefined bounds.

3. These modified trees (represented as strings) served as the labels for our model, representing a random commutators that we aim to translate.

4. Inputs, corresponding to the labels are given by reduced words, obtained from labels by opening all commutator brackets.

Subsequently, we employed a standard training method for the translation model. We used BERT (Devlin et al., 2019) with $2 \cdot 10^6$ parameters as an encoder as well as a decoder. We used AdamW with the learning rate $8 \cdot 10^{-5}$ and the batch size equal to 64.

As with the main models (see Section 5), the training and validation datasets are generated in online mode with the following parameters:

- number of generators $n = 4$,

- maximal length of a word $= 250$,

- maximal length of a leave $= 3$,

- maximal depth of a tree $= 7$.

D.4. Examples of Generated Words

We provide some examples of generated words for $n = 3, 4$ in commutator presentation for various deep models and inference methods, see Section 4.3 for additional details. The presentation is obtained using translator model described above.

- $n = 3$

$$[[y^2, [y^{-1}x^{-1}z^{-1}, [z^2, x]]], x] \qquad \text{(prompt + beam)}$$

$$\left.\begin{array}{l} [[x, y], [yz, x]] \\ [[xy^{-2}x^{-1}z, y], [z, xy]] \\ [[[z, xy], y^{-1}xyx], y] \end{array}\right\} \qquad \text{(ignore + sample)}$$

$$[[x, z], [x^{-1}, y][y, z]] \qquad \text{(mask + sample)}$$

Note that some of the generated words are not in the symmetric commutator subgroup.

- $n = 4$

$$\left.\begin{array}{c} [[xyzp,[y^4,[x,z]]],p] \\ [[(xyzp)^3,[[p^{-1},z^2],y]],x] \end{array}\right\} \quad \text{(ignore + beam)}$$

$$\left.\begin{array}{c} [[yzpx^{-3},x^{-1}],[z^2,[p^2ypy^{-1},y^4]]] \\ [[[y,p^{-1}zp],xzp^2z^{-1}x^{-1}p],[x^3,zyzpxpxyx^{-1}]] \end{array}\right\} \quad \text{(ignore + sample)}$$

$$\left.\begin{array}{c} [[[[z^{-1}y^{-1}x^{-1},p],y],x],z^3] \\ [[z^3,[(xyzp)^2,[y^2,x]]],p^3] \end{array}\right\} \quad \text{(mask + beam)}$$

$$[[p^2x^{-1}pxpx^{-1}px,[pzp^{-1}yzy^{-1},[y,x]]],xyzp] \quad \text{(mask + sample)}$$

$$\left.\begin{array}{c} [[[[p^{-1}z^{-1}y^{-1}x^{-1},z],y],x],p] \\ [[x^3,[[x^{-1},p^{-1}z^{-1}y^{-1}],[p,y]]],z^2] \end{array}\right\} \quad \text{(prompt + beam)}$$

$$\left.\begin{array}{c} [[[p^{-1}y^2p,z],p^{-1}xz^{-1}xzp],[xyzp,zp^3xpx^{-1}z^{-1}]] \\ [[[zpx,y],x^{-1}zx^2zx^{-1}z^{-1}],[[x^3yx,y^{-1}],p^3]] \end{array}\right\} \quad \text{(prompt + sample)}$$

D.5. Details on Greedy Algorithm

Suppose we have a given prefix $p = xyz$ and we aim to generate a word from the intersection of the normal closures $R_1 = \langle x \rangle^F$, $R_2 = \langle y \rangle^F$, and $R_3 = \langle z \rangle^F$. The corresponding stacks S_i created for each normal closure R_i are $S_1 = [y, z]$, $S_2 = [x, z]$, $S_3 = [x, y]$. In a list of candidates for the next token, the token z^{-1} is excluded because it would reduce the length of the prefix p.

Next, the algorithm assigns points to each token based on the two criteria mentioned in Section 4.2. The token y^{-1} would receive a point because it reduces the length of the S_3. The tokens x and x^{-1} would receive points because they do not increase the length of S_1. Similarly, y, y^{-1}, and z would receive points because they do not increase the length of S_2 and S_3 respectively. The algorithm then selects the token with the highest point value as the next token to add to the prefix. In this case, y^{-1} has the highest number of points, so it would be selected as the next token to add to the prefix.

In the case of a "many-token" normal closure, such as $R_0 = \langle xyz \rangle^F$, tokens receive points based on their ability to bring the top of the corresponding stack S_0 closer to a rotation of the normal closure. In particular, if token x_k is the top of S_0, then the token $x_{(k+1) \bmod (n+1)}$ will receive a point.
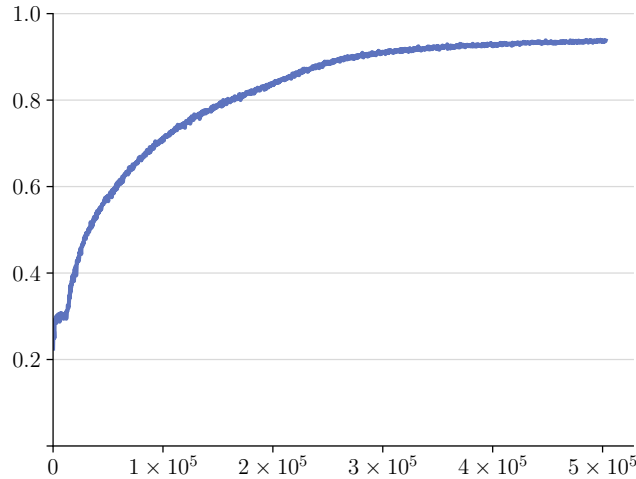

Figure 6: Illustration of commutator translator learning process: BLeU score on validation.

D.6. Auxilary Learning Curves

In Figures 7-9 we present additional graphs of learning process for various hyperparameters like inference method and prefix length. Note that variable prefix allows more variability in models output as well. Color coding is the same as in Figure 3: red for *negative baseline*, yellow for *prompt*, green for *masking*, blue for *ignore*.
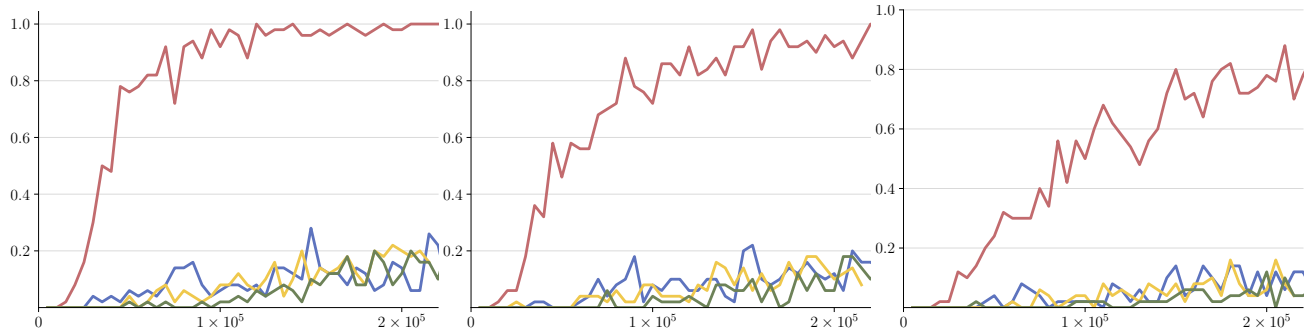

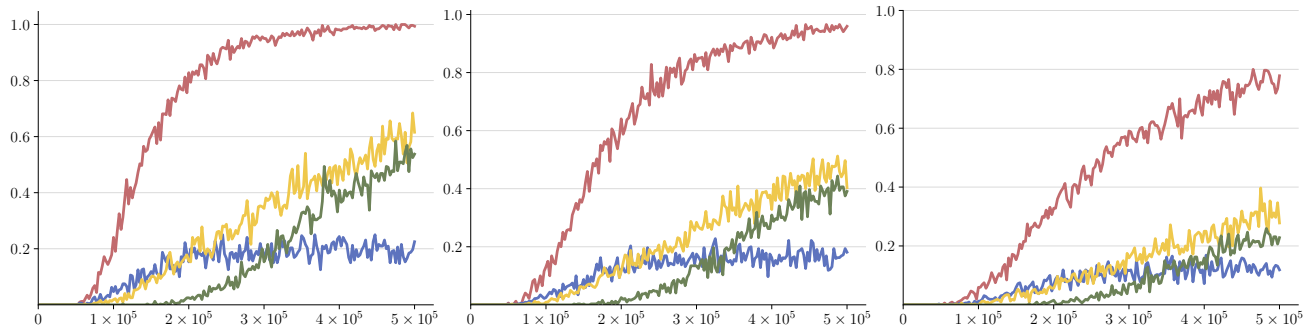Figure 7: Completion ratio with nucleus sampling for $n = 3$ and various length of prefixes $k = 5, \ 7, \ 10$.

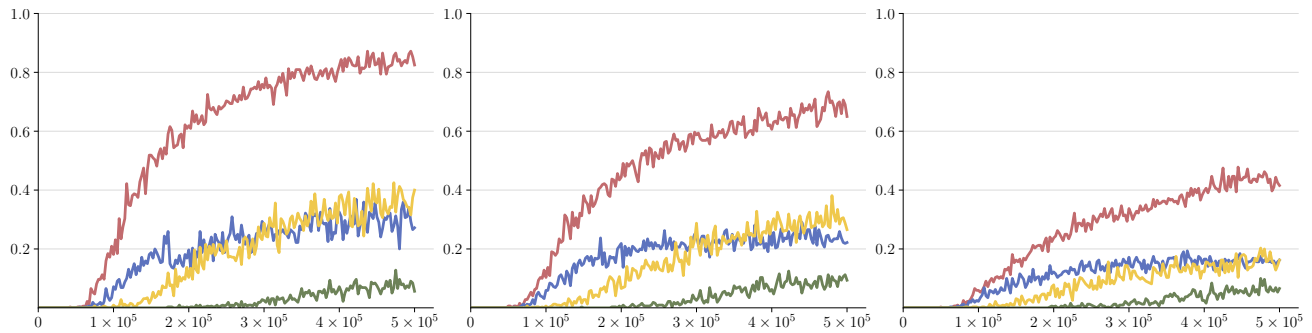Figure 8: Completion ratio with nucleus sampling for $n = 4$ and various length of prefixes $k = 5, \ 7, \ 10$.

Figure 9: Completion ratio with beam search for $n = 4$ and various length of prefixes $k = 5, \ 7, \ 10$.