SpecMER: Fast Protein Generation with K-mer Guided Speculative Decoding

Thomas A. Walton
Georgia Institute of Technology
twalton42@gatech.edu

Darin Tsui
Georgia Institute of Technology
darint@gatech.edu

Aryan Musharaf
Georgia Institute of Technology
amusharaf3@gatech.edu

Amirali Aghazadeh
Georgia Institute of Technology
amiralia@gatech.edu

Abstract

Autoregressive models have transformed protein engineering by enabling the generation of novel protein sequences beyond those found in nature. However, their sequential inference introduces significant latency, limiting their utility in high-throughput protein screening. Speculative decoding accelerates generation by employing a lightweight draft model to sample tokens, which a larger target model then verifies and refines. Yet, in protein sequence generation, draft models are typically agnostic to the structural and functional constraints of the target protein, leading to biologically implausible outputs and a shift in the likelihood distribution of generated sequences. We introduce SpecMER (Speculative Decoding via k-mer Guidance), a novel framework that incorporates biological, structural, and functional priors using k-mer motifs extracted from multiple sequence alignments. By scoring candidate sequences in parallel and selecting those most consistent with known biological patterns, SpecMER significantly improves sequence plausibility while retaining the efficiency of speculative decoding. SpecMER achieves 24–32% speedup over standard autoregressive decoding, along with higher acceptance rates and improved sequence likelihoods.

1 Introduction

Designing proteins with enhanced or novel biological functions is an important problem with wide-ranging applications in therapeutics, sustainability, and drug discovery [1]. Protein design is challenging due to the astronomically large space of possible protein sequences, of which only a tiny fraction is likely to exhibit the desired functions [2]. Generative models trained on the vast landscape of evolved protein sequences have emerged as powerful tools for navigating this combinatorial sequence space. Among them, autoregressive models have been especially successful in the design of proteins with functional properties comparable to natural proteins [3]. Their autoregressive nature integrates well with textual information [4], ensures training stability [5], enables variable-length sequence generation [5], and effectively captures long-range dependencies that encode complex amino acid interactions [4]. Despite these advantages, autoregressive models struggle with pathological repetition [6–8], early termination [7], out-of-distribution generalization [8], and most consequential, high computational cost during inference, resulting in slow sequence generation. This computational cost is a debilitating limitation for high-throughput protein screening, where thousands of protein sequences need to be generated and screened to create large-scale libraries of functional and structurally stable proteins [9]. For instance, generating 20,000 protein sequences of length 200 amino acids using ProGen2-XL, a 6.4-billion-parameter transformer-based autoregressive model, takes approximately 65 hours using a single NVIDIA A6000 GPU. Such slow sequence generation can

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

https://doi.org/10.52202/085713-0570

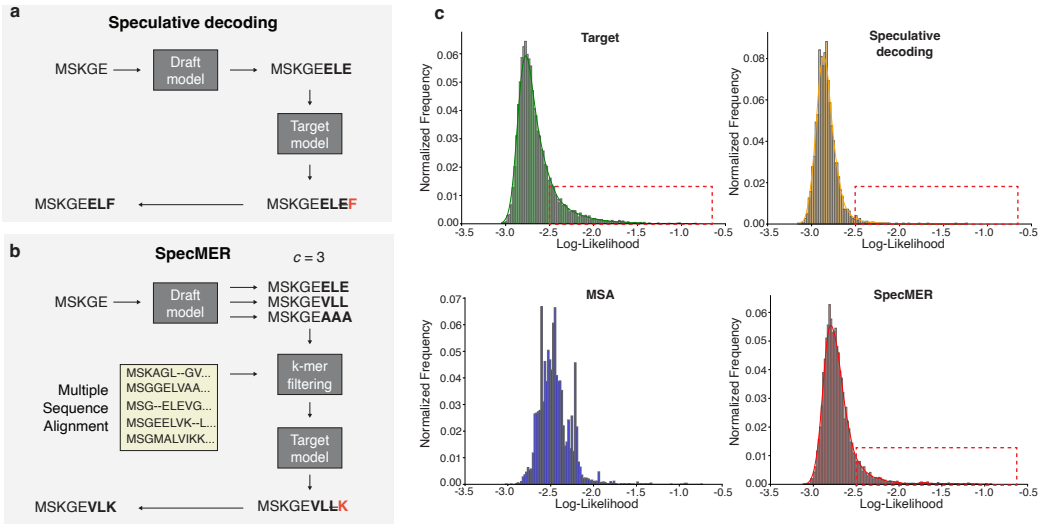


Figure 1: **Overview of speculative decoding and SpecMER. a**, Given a context sequence, speculative decoding uses a draft model to sample tokens. The target model verifies generated tokens, accepting or rejecting them. In this example, two tokens were accepted from the draft model, and the token "E" was corrected to "F". **b**, SpecMER augments the drafting process, generating multiple sequences which are then filtered by k-mers at near-zero cost. K-mers are formed through a multiple sequence alignment, capturing evolutionary patterns through homologous protein sequences. In this example, "VLL" was selected from the proposed candidates, where two tokens were accepted and the token "L" was corrected to "K". **c**, Speculative decoding undersamples high log-likelihood sequences, missing the tail distribution of the target model. K-mers act as guidance for SpecMER, resulting in a distribution of generated sequences that more adequately samples high-likelihood, high-plausibility protein sequences. Additionally, SpecMER retains nearly the same generation speed as speculative decoding, with only a 3% overhead observed at $c = 3$.

delay high-throughput design workflows by days, not accounting for the additional time required for iterative design processes.

Recently, speculative decoding [10] has proven effective in accelerating autoregressive generation in natural language processing. Speculative decoding utilizes a lightweight *draft* model to propose tokens, which are then verified by a larger, more expressive *target* model. This speculate-then-verify framework of generation has demonstrated wall-time speedup factors exceeding $2\times$ by selecting outputs consistent with the target model distribution [11].

In protein sequence generation, however, target-consistent outputs may not align with the most biologically plausible proteins, particularly those that best satisfy structural or functional constraints. This misalignment is amplified in protein design workflows, where satisfying biochemical constraints is essential. While draft models may capture broad structural patterns, small deviations in local structure can lead to misfolded or nonfunctional proteins. Among target-consistent outputs, some may result in proteins with more plausible biological properties; however, such distinctions are not considered during speculative decoding. To address this, incorporating biologically meaningful local priors offers a principled way to guide generation. Notably, k-mers, or short contiguous amino acid subsequences, encode local structural motifs that correlate with both folding and function, and can be extracted with minimal computational cost.

We hypothesized that incorporating structural cues from k-mers could guide speculative decoding toward more biologically plausible proteins, while retaining or even enhancing generation speed. To test this hypothesis, we develop speculative decoding via k-mer guidance (SpecMER), a novel decoding framework that leverages structural motifs from homologous protein sequences to improve both generation speed and quality. By incorporating structural priors into the decoding process, SpecMER provably selects tokens that are not only consistent with the target model but also biased toward completions likely to yield structured, biologically plausible proteins. This motif-informed

Algorithm 1 Token-level maximal coupling [11]

Input: Distributions p, q, Draft sample $X \sim p$.
Compute the residual distribution p_{res} where $\forall x \in \mathcal{V}$, $p_{\text{res}}(x) = \frac{q(x)-\min\{p(x),q(x)\}}{1-\sum_{x' \in \mathcal{V}} \min\{p(x'),q(x')\}}$.
Sample $\eta \sim U(0,1)$
if $\eta \leq \min\left(1, \frac{q(X)}{p(X)}\right)$ **then**
  Return $Y = X$. {Accept the draft token.}
else
  Return $Y \sim p_{\text{res}}$. {Correct the token by sampling from the residual distribution.}
end if

guidance increases the likelihood that generated sequences will fold into stable structures. Our contributions are as follows:

- We develop the first speculative decoding framework for protein generation. Using ProGen2-S as the draft model and ProGen2-M as the target model, we demonstrate an average increase in generation speed by 32%.
- We develop SpecMER, a novel framework that incorporates k-mer frequencies from natural proteins to guide draft token selection. SpecMER produces sequences more likely to resemble natural proteins while retaining increased generation speed from speculative decoding, enabling fast generation of plausible proteins. Software for SpecMER is available at `https://github.com/amirgroup-codes/SpecMER.git`.
- We establish theoretical bounds on wall-time speedup and characterize how guidance from k-mers improves draft token selection, offering practical guidance for estimating speedups under varying system configurations and sequence constraints.

2 Preliminary

Notation. Let draft model P and target model Q be autoregressive models with probability density functions p and q. A protein sequence $x \sim P$ is denoted by sequence $x(i), x(i+1), \ldots, x(j)$, where $x(i)$ is the i^{th} token sampled from p. Tokens are defined by the shared vocabulary $\mathcal{V}$ of P and Q, consisting of amino acids and special tokens.

2.1 Speculative Decoding

Speculative decoding has recently emerged as a highly effective technique for accelerating autoregressive generation. Speculative decoding employs a smaller draft model to propose candidate tokens, which are then refined by a larger, more expressive target model [10, 11]. This approach generates tokens consistent with the target model distribution while significantly reducing decoding latency. Various extensions to speculative decoding have explored architectural modifications [12], tree-based verification [13], distillation of the draft model [14], and optimized scheduling or reference-free inference [15].

Given a context sequence $x(t)$ and a draft model P, speculative decoding efficiently samples L tokens, $\tilde{x}(t+1), \ldots, \tilde{x}(t+L)$. Speculative decoding can be broken down into three steps: 1) Draft construction, 2) Conditional probability computation, and 3) Draft selection (Figure 1a).

1. **Draft construction.** The draft model first samples L tokens, $\tilde{x}(t+1), \ldots, \tilde{x}(t+L)$. For every token $i < L$, we use the draft model to compute the conditional probability of the next token y, $P(y \mid x(t), \tilde{x}(t+1:t+i))$, where $y \in \mathcal{V}$.
2. **Conditional probability computation.** For every token $i < L$, we use the target model Q to compute the conditional probability of the next token $Q(y \mid x(t), \tilde{x}(t+1:t+i))$.
3. **Draft selection.** We validate the first L' out of L tokens based on the conditional probabilities of the draft and target models. We set $x(t+i) = \tilde{x}(t+i)$ for $i \leq L'$. For the rejected token $\tilde{x}(t+L'+1)$, sample from a residual distribution and correct it.

After steps 1 through 3, we use $x(t+L'+1)$ as the next context and continue the speculative decoding process iteratively. Algorithm 1 [11] details the acceptance and rejection process for each

https://doi.org/10.52202/085713-0570

token generated by the draft model. Rejected tokens are corrected by sampling from the residual distribution $p_{res}(x)$.

The correctness guarantees of speculative decoding are formally defined for ancestral sampling, where outputs are drawn directly from the full target distribution. In protein language models, however, ancestral sampling is rarely used in practice, as it often produces implausible or repetitive sequences [8]. Instead, decoding with nucleus sampling (top-p) is common to balance plausibility and diversity, and is utilized with both ProGen and ProGen2 [4, 8, 16, 17]. When paired with speculative decoding, nucleus sampling truncates the low-probability tail of the target distribution, as observed in Figure 1. Since this tail accounts for a small fraction of probability mass ($p = 0.95$ in this study), outputs remain closely aligned with the target model, with differences confined to rare tail events.

Speculative decoding has demonstrated impressive speedups in natural language generation, where structural constraints are minimal. However, existing methods are agnostic to domains like protein design, where sequences must satisfy intricate biochemical and structural requirements. In this setting, small changes can disrupt folding, stability, or function, and distributional drift between the draft and target models can accumulate over time, degrading generation quality. To the best of our knowledge, no prior work in speculative decoding incorporates sequence-level structural information to guide draft generation, limiting its applicability in this domain. This work addresses that gap by introducing a structure-aware drafting strategy that leverages local sequence motifs to guide speculative decoding toward more plausible protein sequences.

Which tokens are optimal? Assume that the acceptance ratio of token $x(t+i)$ sampled from p is given by $\beta(t+i)$. If we assume each $\beta(t+i)$ is i.i.d., then $\mathbb{E}[\beta] = \alpha$, where α indicates the overall acceptance ratio for a given configuration of speculative decoding [10]. It follows from [11] that α is the complement to the expected total variation distance between p and q. Therefore, α describes exactly how well p approximates q. If the generation cost coefficient $c_e = \frac{M_p}{M_q}$ is known, where M_p and M_q represent the time it takes for P and Q to generate one full-length sequence, the wall-time speedup for drafts of length γ can be computed as:

$$\frac{1-\alpha^{\gamma+1}}{(1-\alpha)(\gamma c_e+1)}. \tag{1}$$

Therefore, an effective speculative decoder should aim to maximize α or minimize c_e. Importantly, the set of tokens that increase α is not limited to the highest likelihood token under the target model. Rather, there exists a set of completions E which satisfy the requirements of Algorithm 1. While speculative decoding excels at proposing any given token from E, only a subset of these completions may align with biologically meaningful constraints. For instance, two candidate sequences "EEL" and "VLL" (as in Figure 1) might both pass the maximal coupling acceptance check, yet "VLL" may correspond to a conformation with higher predicted structural stability. Current speculative decoding frameworks lack a mechanism to prioritize such candidates, as they do not account for auxiliary objectives, which are essential in protein design but not explicitly modeled during generation.

2.2 K-mers

K-mers are contiguous substrings of length k extracted from biological sequences. In proteins, k-mers often correspond to recurring structures such as alpha-helices or beta-strands, capturing local features that are critical to folding and function. Compared to analogous representations in other domains, protein k-mers encode a dense combination of physicochemical properties, structural motifs, and functional sites, all at minimal computational cost. In contrast, n-grams in natural language primarily reflect syntactic or statistical patterns without necessarily encoding physical or functional constraints. Due to their high information content, k-mers have been used across many applications in computational biology, including tasks such as classification, motif discovery, or alignment-free sequence comparison [18]. Notably, Tranception [19] utilizes k-mer attention to better capture structural dependencies during the generation process.

3 Methods

3.1 SpecMER: K-mer Guided Protein Generation

SpecMER leverages k-mer statistics to guide protein sequence generation. Figure 1 details the modified sampling procedure of SpecMER. Instead of sampling one completion at a time from

the draft model, SpecMER batch generates multiple completions, which we refer to as candidate sequences. A k-mer scoring function then assesses each length L candidate sequence, providing a score based on the selected values of k. The highest-scoring candidate is then scored by the draft model and verified by the target model. To summarize, SpecMER decodes protein sequences using the following steps, which repeat until generation is complete:

1. **Candidate construction.** The draft model first samples L tokens c times, producing a batch with dimensions $c \times L$ of candidate sequences.
2. **K-mer scoring.** A k-mer scoring function then selects the candidate sequence $[\tilde{x}^{(1)}(t+1), \ldots, \tilde{x}^{(c)}(t+L)]$ which yields the highest score.
3. **Conditional probability computation.** For every token $i < L$, the selected candidate sequence is scored using the draft and target models.
4. **Draft selection.** Each proposed token in the candidate sequence is accepted or rejected in accordance with Algorithm 1. The new context is set to $x(t+i) = \tilde{x}(t+i)$, where i is the number of accepted tokens to add.

3.2 K-mer Scoring

After the draft model samples c candidate sequences, each sequence is evaluated based on its k-mer frequency. K-mers are computed prior to generation using a multiple sequence alignment (MSA) constructed for a target wild-type protein sequence. MSA data consists of homologous sequences identified based on similarity to the wild-type protein. For each wild-type sequence, we retrieve its corresponding MSA from ProteinGym [20]. The resulting alignment encapsulates evolutionary and functional constraints, making MSA-derived data particularly well-suited for k-mer analysis.

We extract k-mers by applying a sliding window of size k across all sequences in the MSA. We selected k values of $1, 3$, and 5, in accordance with [19]. We chose not to exceed $k = 5$ as the number of possible k-mers can grow exponentially relative to the vocabulary size, increasing access cost during inference. K-mers are tracked over all sequences in the MSA as they appear, enabling better memory and computational efficiency during retrieval. The resulting k-mer frequencies serve as a proxy for biological plausibility and are normalized to form a probability distribution. During the decoding process, k-mers are used to score candidate sequences s:

$$\text{Score}(s) = \frac{1}{L} \sum_{k \in K} \sum_{i=0}^{L-k} \mathbb{P}_k(s(i : i+k)) \,, \tag{2}$$

where L is the length of the candidate sequence, K is the set of k-mers to evaluate, and $\mathbb{P}_k$ is the probability of a k-mer given the normalized distribution from the MSA. The scoring is additive rather than multiplicative to avoid a zero score in the case of unseen k-mers and to promote exploration of sequences with partially formed motifs. The candidate with the highest k-mer score is selected to continue the decoding process. A significant advantage of k-mer scoring is that it can be pre-computed and is easy to access during generation, contributing negligible computational overhead. Consequently, SpecMER has the same computational complexity as speculative decoding: $\mathcal{O}(L^2)$.

3.3 Performance Gain from K-mers

Vanilla speculative decoding drafts individual sequences and verifies each token in the sequence independently. The maximal coupling acceptance algorithm (Algorithm 1) ensures that selected tokens do not diverge from the target model distribution, increasing generation speed with no sacrifice to the quality of the generated sequences. However, there could be many tokens that are consistent with the target distribution at each verification step, passing the acceptance criteria $\eta \leq \min(1, \frac{Q(x)}{P(x)})$. To quantify the looseness of this criterion and the efficacy of external scoring functions, we describe the conditions for selecting tokens consistent with the target model and optimal under the scoring function.

Proposition 4.4. *The expected batch-and-select acceptance satisfies:*

$$\mathbb{E}[A^*] = 1 - (1-\alpha)^m - \epsilon \,, \tag{3}$$

where $\mathbb{E}[A^*]$ is the acceptance ratio of the speculative decoder, α is the acceptance ratio of vanilla speculative decoding, and m is the number of batch generated candidate sequences.

https://doi.org/10.52202/085713-0570

Proof. Let P be the draft model and Q the target as before. For a drafted (candidate) sequence s, let $M(s)$ be an indicator of the acceptance or rejection of s. That is, $M(s) = 1$ if Q accepts s, and $M(s) = 0$ otherwise. For a given context sequence x,

$$\mathbb{P}[M(s) = 1 \mid s] = r(s) = \min\left(1, \frac{Q(s|x)}{P(s|x)}\right). \tag{4}$$

If s is a single draw, as in vanilla speculative decoding, then the acceptance rate α is defined as $\mathbb{P}_{s\sim P}[M(s) = 1] = \mathbb{E}_{s\sim P}[r(s)]$. In the case of batch generation, m [1] i.i.d. candidates are selected for each draft: $s_1, \ldots, s_m \sim P$. Following drafting, the scoring function S selects the highest scoring candidate $s^* = \text{argmax}_i S(s_i)$. Let acceptance indicator $A^* = M(s^*)$, and define $E = \{\exists i : M(s_i) = 1\}$ as the event in which at least one s_i is an acceptable candidate under Q. The probability that an acceptable candidate s_i exists but is not selected by S is defined by the misranking loss $\epsilon = \mathbb{P}[E \wedge (A^* = 0)]$. Consider the case where no candidates satisfy the acceptance criteria of Q, $\mathbb{P}[E^c] = \mathbb{P}[M(s_1), \ldots, M(s_m)] = (1-\alpha)^m$. Intuitively, $(1-\alpha)^m$ is the probability that among m candidates, none pass the criteria of Q. The probability that a selected candidate will be rejected is then defined by $\mathbb{P}[A^* = 0] = \mathbb{P}[E^c] + \mathbb{P}[E \wedge (A^* = 0)] = (1-\alpha)^m + \epsilon$. The overall acceptance of the decoder is then:

$$\mathbb{E}[A^*] = 1 - \mathbb{P}[A^* = 0] = 1 - (1-\alpha)^m - \epsilon\,. \quad \square \tag{5}$$

In practice, $\mathbb{E}[A^*]$ is measured empirically as the average acceptance ratio of the speculative decoder, and α is the average acceptance ratio of vanilla speculative decoding (no external scoring). ϵ is the probability that the scoring function chooses the wrong candidate, given that there was an acceptable candidate to choose. A good scoring function should yield $\epsilon \ll 1$, indicating that it excels in selecting candidates that will be accepted by Q. Additional bounds on expected speedups for given hardware configurations can be found in Appendix A.

4 Experiments

In this section, we detail experiments across seven different proteins. We tested SpecMER with ProGen2 [4], an autoregressive protein language model. For each experiment, we conditioned generation on a fixed context length from a given wild-type protein as detailed in Table 1. We selected proteins with varying molecular functions and lengths to ensure robustness of testing. Each experiment consisted of generating 200 sequences on an NVIDIA RTX A6000 GPU. Sequences were generated up until the length of the wild-type protein or until a stop token was generated. We swept across a set of hyperparameters for each decoding method to determine the configuration with the highest acceptance ratio and generation speed. Results of a full hyperparameter sweep are located in Appendix B.3.

4.1 Datasets

We focus on the conditional generation task, where the objective is to generate starting from a context sequence. We selected seven proteins with varying functions and collected their MSA from ProteinGym [20]. For each protein, we set the context length to roughly 10% of the wild-type sequence, balancing exploration of sequence space while avoiding common pitfalls in autoregressive generation such as pathological repetition. Table 1 details each protein and its respective molecular function.

Table 1: Summary of proteins and context length used.

Protein	Description	Molecular Function	Length	Context	MSA Sequences	Citation
GFP	Green fluorescent protein	Fluorescence	238	20	396	[21]
RBP1	RalA-binding protein 1	Stability	52	10	135922	[22]
ParD3	Antitoxin ParD3	Growth enrichment	93	15	38613	[23]
GB1	IgG-binding domain of protein G	Binding	56	10	44	[24]
Bgl3	β-glucosidase	Enzyme function	501	50	105913	[25]
ADRB2	Beta-2 adrenergic receptor (GPCR)	Receptor activity	413	40	204722	[26]
CBS	Cystathionine beta-synthase	Growth	551	50	19563	[27]

[1] We opt to use m in place of c to avoid confusion with set complement.

4.2 Experimental Setup

Hyperparameters. Each experiment consisted of generating 200 protein sequences given the starting context sequence (Table 1). We swept over the following hyperparameters: draft tokens $\gamma \in \{5, 10, 15\}$, temperatures $T \in \{0.7, 1, 1.4\}$, and k-mers $k \in \{(1), (3), (1, 3), (1, 3, 5)\}$. Sequences were sampled using nucleus (top-p) sampling, setting $p = 0.95$. We opted for a large value of p to enable diverse sampling, controlling how deterministic the output is instead with temperature.

Evaluation metrics. We measured four categories across all experiment configurations: acceptance ratio, negative log-likelihood (NLL), top-20 NLL, and top-5 NLL. Acceptance ratio α is defined as:

$$\alpha = \frac{\text{No. accepted tokens}}{\text{No. accepted tokens + No. rejected tokens}} \ , \tag{6}$$

where a higher acceptance ratio is better. After generation, we computed NLL, top-20 NLL, and top-5 NLL under the target model, normalizing for length. We assessed the top-20 and top-5 NLL, as protein design experiments often create a library of sequences and filter for only the most plausible candidates. Sequences with higher likelihoods are more likely to resemble natural proteins, making likelihood a proxy for sequence quality [28, 29]. In addition to likelihood, we computed predicted local-distance difference test (pLDDT) scores [30] using ESMFold [31]. The pLDDT score measures how confident the structure prediction is for a given protein sequence, and has been demonstrated to correlate with the likelihood that a protein sequence will fold into a stable structure [30]. We evaluated biological plausibility using both sequence likelihood and pLDDT, under the assumption that sequences achieving high scores on both metrics are more likely to adopt stable, protein-like structures consistent with natural biological constraints. Sequence likelihoods and pLDDT scores for each wild-type sequence can be found in Appendix B.3.

We generated embeddings for both MSA and sampled sequences ($c = 1, 2, 3, 5$) using ESM2 [31], and used PCA to qualitatively assess alignment between natural and generated sequences. For each decoding method, we selected the top three configurations by average NLL, further filtering for the 100 sequences with the highest pLDDT scores. Bgl3, ADRB2, and CBS were excluded from the pLDDT score analysis due to sequence length limitations of ESMFold.

Generation speed experiments were conducted using ProGen2-S and ProGen2-M as baselines. We tested vanilla speculative decoding ($c = 1$) and SpecMER ($c = 2, 3, 5$) against these baselines, tracking tokens per second. We swept over a range of hyperparameters to find the fastest configuration, averaging over 20 generated sequences.

4.3 Results

Acceptance and log-likelihood. The best results per experimental configuration are reported in Table 2. We observed that acceptance ratios were marginally higher on average for SpecMER compared to speculative decoding and increased with c. Sequences generated with SpecMER had significantly lower NLL, including a lower top-20 and top-5 NLL. That is, while k-mer scoring increased acceptance for some configurations, it nearly unanimously produced protein sequences with higher likelihood.

We observed that the acceptance ratio was sensitive to T and k. Specifically, k strongly influenced the acceptance ratio for some proteins, while others were relatively unaffected. For example, GFP exhibited the best performance with $k = \{1, 3\}$, whereas RBP1 had no preference toward k. A full discussion on important hyperparameters can be found in Appendix B.3.

Generation of plausible sequences. SpecMER yielded sequences with higher likelihoods compared to speculative decoding, indicating that sequences generated by this framework are more likely to be biologically plausible. To corroborate this claim, we computed pLDDT scores, a measure that correlates with the likelihood that a protein will fold into a stable conformation. Table 3 details the average pLDDT scores across 300 sequences per configuration. We observed that on average, SpecMER produces sequences with higher pLDDT scores. GFP was an exception to this trend, exhibiting higher scores for vanilla speculative decoding. However, top-5 pLDDT scores for GFP demonstrated that top-end performance was comparable between all decoding methods (Table 10 in Appendix D). The pLDDT scores did not increase linearly with c; instead, the best value of c varied across proteins. This suggests that the optimal value of c reflects a trade-off between exploration of diverse sequence space and adherence to the likelihood distribution of the target model. Furthermore, this trade-off may depend on structural characteristics of the target protein.

 https://doi.org/10.52202/085713-0570

Table 2: Decoding results using ProGen2. Each metric is averaged over 200 generated sequences. The value of c chosen for SpecMER experiments represents the number of batch-generated candidate sequences, with $c = 1$ indicating speculative decoding. For each method, a sweep over hyperparameters is conducted as described in Section 4. The best results in each category are reported. A full summary of results is located in Appendix B.3.

Decoding Method	Protein	Accept Ratio ↑	NLL ↓	Top-20 NLL ↓	Top-5 NLL ↓
Speculative Decoding	GFP	0.911 ± 0.029	2.45 ± 0.42	1.38 ± 0.32	0.98 ± 0.11
	RBP1	**0.938** ± 0.043	2.73 ± 0.19	2.43 ± 0.19	2.01 ± 0.44
	ParD3	0.902 ± 0.036	1.93 ± 0.59	0.80 ± 0.21	**0.50** ± 0.09
	GB1	**0.927** ± 0.040	2.79 ± 0.15	2.60 ± 0.25	2.28 ± 0.33
	Bgl3	0.852 ± 0.025	0.91 ± 0.11	0.76 ± 0.06	0.67 ± 0.08
	ADRB2	0.866 ± 0.079	1.90 ± 0.65	1.18 ± 0.30	0.78 ± 0.16
	CBS	**0.910** ± 0.051	2.42 ± 0.42	2.06 ± 0.44	1.43 ± 0.38
SpecMER ($c = 3$)	GFP	0.937 ± 0.076	1.23 ± 0.68	0.44 ± 0.04	0.35 ± 0.04
	RBP1	0.926 ± 0.038	2.58 ± 0.32	2.08 ± 0.39	1.48 ± 0.21
	ParD3	0.919 ± 0.093	1.56 ± 0.49	0.75 ± 0.12	0.54 ± 0.06
	GB1	0.924 ± 0.043	2.71 ± 0.18	2.43 ± 0.26	2.06 ± 0.57
	Bgl3	**0.869** ± 0.030	0.81 ± 0.14	0.65 ± 0.07	0.54 ± 0.08
	ADRB2	0.857 ± 0.062	1.33 ± 0.50	0.77 ± 0.13	0.61 ± 0.04
	CBS	0.902 ± 0.081	2.17 ± 0.66	1.47 ± 0.50	0.84 ± 0.16
SpecMER ($c = 5$)	GFP	**0.945** ± 0.084	**1.09** ± 0.64	**0.41** ± 0.07	**0.33** ± 0.02
	RBP1	0.926 ± 0.047	**2.41** ± 0.40	**1.72** ± 0.30	**1.25** ± 0.12
	ParD3	**0.942** ± 0.063	**1.33** ± 0.41	**0.67** ± 0.12	0.52 ± 0.05
	GB1	0.925 ± 0.038	**2.61** ± 0.27	**2.20** ± 0.31	**1.74** ± 0.15
	Bgl3	0.867 ± 0.022	**0.80** ± 0.17	**0.63** ± 0.11	**0.46** ± 0.05
	ADRB2	**0.869** ± 0.082	**1.03** ± 0.60	**0.57** ± 0.11	**0.43** ± 0.04
	CBS	0.908 ± 0.066	**1.87** ± 0.68	**1.14** ± 0.32	**0.74** ± 0.07

Table 3: Average pLDDT scores across four different proteins. Sequences are collected from the three best configurations for each decoding method, determined by the highest average log-likelihood under the target model (ProGen2-M).

Protein	Speculative Decoding ($c = 1$)	SpecMER ($c = 2$)	SpecMER ($c = 3$)	SpecMER ($c = 5$)
GFP (↑)	**0.493** ± 0.156	0.449 ± 0.160	0.479 ± 0.191	0.426 ± 0.176
RBP1 (↑)	0.571 ± 0.116	0.664 ± 0.127	0.700 ± 0.115	**0.740** ± 0.105
ParD3 (↑)	0.638 ± 0.206	**0.650** ± 0.202	0.584 ± 0.201	0.519 ± 0.176
GB1 (↑)	0.465 ± 0.084	0.464 ± 0.071	0.477 ± 0.075	**0.504** ± 0.09

When compared to speculative decoding ($c = 1$), SpecMER ($c = 5$) generated sequences that more closely aligned with homologous sequences from an MSA, while also exploring beyond the MSA centroid. Figure 2a illustrates this result for RBP1, where SpecMER captures both MSA-like sequences and novel variants. Notably, many of these divergent sequences exhibited high likelihood under the target model and achieved higher pLDDT scores on average (Figure 2 and Table 3), indicating a greater degree of biological plausibility. Beyond structural plausibility, we also assessed sequence diversity and found that both speculative decoding and SpecMER generated novel sequences far from the wild-type while maintaining high inter-sequence diversity (see Appendix D.1). Taken together, these results suggest that k-mer guidance preserves evolutionary plausibility while facilitating exploration of high-quality regions of sequence space.

To investigate the role of MSA-derived k-mers, we conducted two ablation experiments (see Appendix C for full details). First, we tested SpecMER under cross-protein mismatches, conditioning on GFP while using GB1-derived k-mers to select continuations, and conditioning on GB1 while using Bgl3-derived k-mers. In both cases, likelihoods dropped relative to protein-specific k-mers, indicating that observed gains depend on the correct evolutionary context. Second, we reduced the

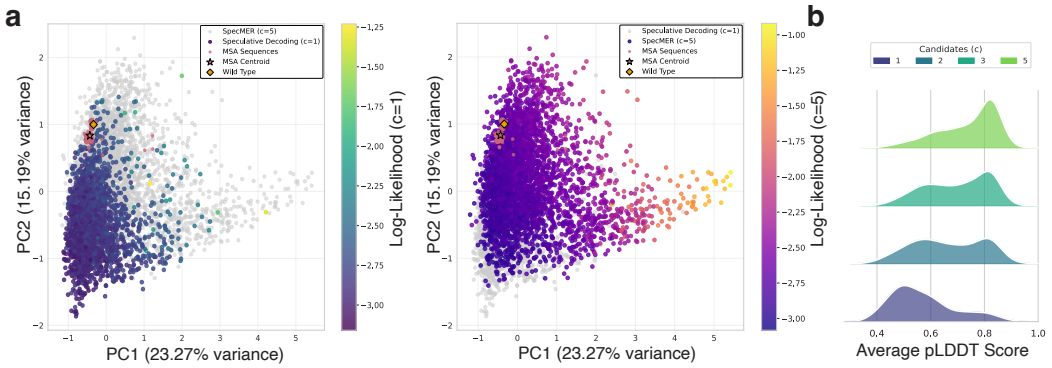


Figure 2: Impact of candidates c on biological plausibility of sequences generated by vanilla speculative decoding ($c = 1$) versus SpecMER for RBP1. **a**, PCA of embeddings from sequences generated by speculative decoding ($c = 1$) (highlighted on the left plot) and SpecMER ($c = 5$) (highlighted on the right plot), compared to embeddings from the MSA of RBP1. Each sequence is shaded by its likelihood under the target model (ProGen2-M). Sequences generated by SpecMER cluster more closely to the MSA embeddings and demonstrate a higher likelihood, indicating a higher degree of biological plausibility. **b**, pLDDT distributions for different configurations of SpecMER, with $c = 1$ representing vanilla speculative decoding. SpecMER demonstrates higher structural confidence scores, indicating that generated sequences are more likely to fold.

depth of the Bgl3 MSA from ∼105k to 1k sequences. The resulting likelihoods steeply declined, supporting our conclusion that MSA quality impacts the performance of SpecMER.

While vanilla speculative decoding ($c = 1$) routinely undersamples the high-likelihood tail of the target distribution (Figure 1c; see Section 2.1), our experiments demonstrate that SpecMER overcomes this limitation. By using k-mers extracted from the MSA to bias the draft model toward structurally grounded completions, SpecMER achieves improved coverage of biologically plausible sequences, often exceeding the likelihoods obtained by target-only decoding. Table 4 compares the top-20 NLL values from SpecMER ($c = 5$) compared to those from the target model, highlighting the improved coverage of high-likelihood sequences.

Table 4: Comparison of top-20 NLL from sequences generated by the target model (ProGen2-M) versus SpecMER using the same temperature values. Sequences are scored using ProGen2-M.

Method	**Bgl3**	**GFP**	**RBP1**	**GB1**	**ParD3**
Target	0.78 ± 0.02	0.51 ± 0.04	1.62 ± 0.15	2.27 ± 0.24	0.69 ± 0.11
SpecMER ($c = 5$)	0.63 ± 0.11	0.41 ± 0.07	1.72 ± 0.30	2.20 ± 0.31	0.67 ± 0.12

Wall-time speedups. We observed an increase in sequence likelihoods as the number of candidates increased (Table 2a). This improvement, however, incurred a slight cost to generation speed, indicating a trade-off between sequence quality and computational efficiency. As demonstrated in Table 5, generating more candidates slowed decoding, though the decline in speed was minimal. Speculative decoding ($c = 1$) achieved the fastest generation speed with an average speedup of 32%, while SpecMER ($c = 5$) yielded the slowest configuration, achieving a modest 24% speedup. Notably, $c = 3$ yielded the largest increase in sequence log-likelihood for the smallest decrease in speed. We noticed an increase in variance as c increased due to batch generation, which we discuss further in Appendix B.1.

We also tested the generation speed with ProGen2-XL as the target model and ProGen2-M as the draft, finding that with $c = 3$, SpecMER demonstrated a speed increase of 38% over target-only decoding. In reference to the example from the introduction, SpecMER ($c = 3$) would have taken 40 hours to generate 20,000 protein sequences of length 200 amino acids, saving an entire day of computing.

https://doi.org/10.52202/085713-0570

Table 5: Generation speed (tokens/sec) over the best configuration per method. Results are averaged over GFP, RBP1, and GB1. The draft and target models are ProGen2-S and ProGen2-M, and the baseline is speculative decoding ($c = 1$). Speedup is calculated with respect to the target model.

-	**Draft**	**Target**	**Baseline**	**SpecMER** ($c = 2$)	**SpecMER** ($c = 3$)	**SpecMER** ($c = 5$)
Toks/sec	74.11	31.48	41.62 ± 0.88	41.43 ± 1.24	40.57 ± 2.83	39.11 ± 4.50
Speedup	-	-	32%	32%	29%	24%

While increasing the number of candidates decreased tokens per second, it resulted in selecting better candidates the majority of the time. SpecMER ($c = 5$) had the lowest misranking error (Equation 3 and Figure 3), selecting an acceptable sequence from a pool of candidates 92% of the time. Of the set of tokens consistent with the target distribution, SpecMER was able to correctly discern between tokens that increased acceptance and tokens that increased both acceptance and biological plausibility. To summarize the trade-off space, increasing c decreases ϵ, tokens per second, and negative log-likelihood, while increasing acceptance ratio and pLDDT scores. Further discussion on this trade-off space can be found in Appendix B.2.

5 Conclusion

Discussion. In this study, we developed the first speculative decoding framework for protein generation. Using ProGen2-S as the draft model and ProGen2-M as the target model, we achieved an average speedup of 32% across functionally diverse proteins. Building on this foundation, we developed SpecMER, a novel framework that incorporates k-mer guidance derived from multiple sequence alignments to guide generation toward biologically meaningful regions of sequence space.

SpecMER drafts multiple candidate sequences and uses evolutionary patterns from k-mers to select completions that are more likely to fold into stable, biologically plausible proteins. As illustrated in Figure 1, speculative decoding often undersamples high-likelihood sequences, despite proposing tokens consistent with the target model distribution. SpecMER addresses this deficiency by guiding generation toward sequences that both align with MSA-derived motifs and exhibit improved log-likelihood and pLDDT scores. In doing so, it preserves the speed advantages of speculative decoding while producing sequences that are more likely to fold into stable conformations, achieving the best of both efficiency and quality.

Beyond empirical gains, we also provide bounds on wall-time speedups and define the criteria for selecting tokens that conform to guidance and acceptance objectives. Furthermore, we offer practical guidance for tailoring hyperparameter configurations to maximize performance improvements for protein design workflows. As speculative decoding techniques continue to evolve and more protein data becomes available, we anticipate larger speedups in automated protein generation. SpecMER marks a step forward in accelerating protein design, paving the way for high-throughput design workflows and beyond.

Limitations. The effectiveness of SpecMER depends on the quality of the input alignment (MSA); performance may degrade when informative motifs are sparse or unavailable. For example, target proteins with extensive disordered regions or lacking discernible structural patterns may benefit less from k-mer guidance. While k-mers impose minimal computational overhead, increasing the number of drafted candidates c increases compute and therefore energy cost, which may limit scalability in certain settings. SpecMER utilizes batch generation to efficiently draft multiple candidates, though this is not strictly parallel. In practice, a fully parallel implementation could yield even greater speedups, but was not attainable given the limitations of our hardware setup.

Future work. While our method was developed for conditional generation of protein sequences, its principles could extend to natural language, where low-cost scoring functions can guide generation under constraints such as style or safety.

Acknowledgments and Disclosure of Funding

This work was supported by the Parker H. Petit Institute for Bioengineering and Biosciences (IBB) interdisciplinary seed grant, the Institute of Matter and Systems (IMS) seed grant, the National Science Foundation (NSF) Graduate Research Fellowship Program (GRFP), and Georgia Institute of Technology start-up funds.

References

[1] Xiangru Tang, Howard Dai, Elizabeth Knight, Fang Wu, Yunyang Li, Tianxiao Li, and Mark Gerstein. A survey of generative AI for de novo drug design: new frontiers in molecule and protein generation. *Briefings in Bioinformatics*, 25(4):bbae338, 07 2024. ISSN 1477-4054. doi: 10.1093/bib/bbae338.

[2] Pascal Notin, Nathan Rollins, Yarin Gal, Chris Sander, and Debora Marks. Machine learning for functional protein design. *Nature Biotechnology*, 42(2):216–228, February 2024. ISSN 1546-1696. doi: 10.1038/s41587-024-02127-0.

[3] Jung-Eun Shin, Adam J. Riesselman, Aaron W. Kollasch, Conor McMahon, Elana Simon, Chris Sander, Aashish Manglik, Andrew C. Kruse, and Debora S. Marks. Protein design and variant prediction using autoregressive generative models. *Nature Communications*, 12(1):2403, 2021.

[4] Erik Nijkamp, Jeffrey A. Ruffolo, Eli N. Weinstein, Nikhil Naik, and Ali Madani. ProGen2: exploring the boundaries of protein language models. *Cell Systems*, 14(11):968–978, 2023.

[5] Alexey Strokach and Philip M. Kim. Deep generative modeling for protein design. *Current Opinion in Structural Biology*, 72:226–236, 2022.

[6] Chloe Hsu, Robert Verkuil, Jason Liu, Zeming Lin, Brian Hie, Tom Sercu, Adam Lerer, and Alexander Rives. Learning inverse folding from millions of predicted structures. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 8946–8970. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/hsu22a.html`.

[7] Han Spinner, Aaron W Kollasch, and Debora Susan Marks. How well do generative protein models generate? In *ICLR 2024 Workshop on Generative and Experimental Perspectives for Biomolecular Design*, 2024. URL `https://openreview.net/forum?id=gH3QEMNRe1`.

[8] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=rygGQyrFvH`.

[9] Zachary Wu, S. B. Jennifer Kan, Russell D. Lewis, Bruce J. Wittmann, and Frances H. Arnold. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proceedings of the National Academy of Sciences*, 116(18):8852–8858, April 2019. ISSN 1091-6490. doi: 10.1073/pnas.1901979116.

[10] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR, 2023.

[11] Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, and Felix Yu. SpecTr: Fast speculative decoding via optimal transport. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[12] Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR, 2024.

 https://doi.org/10.52202/085713-0570

[13] Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. SpecInfer: Accelerating large language model serving with tree-based speculative inference and verification. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, page 932–949. ACM, April 2024. doi: 10.1145/3620666.3651335.

[14] Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. DistillSpec: Improving speculative decoding via knowledge distillation. In *International Conference on Learning Representations (ICLR)*, 2024.

[15] Nan Yang, Tao Ge, Liang Wang, Binxing Jiao, Daxin Jiang, Linjun Yang, Rangan Majumder, and Furu Wei. Inference with reference: Lossless acceleration of large language models, 2023. URL `https://arxiv.org/abs/2304.04487`.

[16] Ali Madani, Ben Krause, Eric R. Greene, Subu Subramanian, Benjamin P. Mohr, James M. Holton, Jose Luis Olmos, Caiming Xiong, Zachary Z. Sun, Richard Socher, James S. Fraser, and Nikhil Naik. Large language models generate functional protein sequences across diverse families. *Nature Biotechnology*, 41(8):1099–1106, January 2023. ISSN 1546-1696. doi: 10.1038/s41587-022-01618-2. URL `http://dx.doi.org/10.1038/s41587-022-01618-2`.

[17] Jeremie Theddy Darmawan, Yarin Gal, and Pascal Notin. Sampling protein language models for functional protein design. In *NeurIPS 2023 Generative AI and Biology (GenBio) Workshop*, 2023. URL `https://openreview.net/forum?id=JPOW9FToYX`.

[18] Camille Moeckel, Manvita Mareboina, Maxwell A. Konnaris, Candace S.Y. Chan, Ioannis Mouratidis, Austin Montgomery, Nikol Chantzi, Georgios A. Pavlopoulos, and Ilias Georgakopoulos-Soares. A survey of k-mer methods and applications in bioinformatics. *Computational and Structural Biotechnology Journal*, 23:2289–2303, December 2024. ISSN 2001-0370. doi: 10.1016/j.csbj.2024.05.025.

[19] Pascal Notin, Mafalda Dias, Jonathan Frazer, Javier Marchena-Hurtado, Aidan N. Gomez, Debora Marks, and Yarin Gal. Tranception: Protein fitness prediction with autoregressive transformers and inference-time retrieval. In *Proceedings of the International Conference on Machine Learning*, pages 16990–17017. PMLR, 2022.

[20] Pascal Notin, Aaron Kollasch, Daniel Ritter, Lood van Niekerk, Steffanie Paul, Han Spinner, Nathan Rollins, Ada Shaw, Rose Orenbuch, Ruben Weitzman, Jonathan Frazer, Mafalda Dias, Dinko Franceschi, Yarin Gal, and Debora Marks. Proteingym: Large-scale benchmarks for protein fitness prediction and design. In *Advances in Neural Information Processing Systems*, volume 36, pages 64331–64379. Curran Associates, Inc., 2023.

[21] Karen S. Sarkisyan, Dmitry A. Bolotin, Margarita V. Meer, Dinara R. Usmanova, Alexander S. Mishin, George V. Sharonov, Dmitry N. Ivankov, Nina G. Bozhanova, Mikhail S. Baranov, Onuralp Soylemez, Natalya S. Bogatyreva, Peter K. Vlasov, Evgeny S. Egorov, Maria D. Logacheva, Alexey S. Kondrashov, Dmitry M. Chudakov, Ekaterina V. Putintseva, Ilgar Z. Mamedov, Dan S. Tawfik, Konstantin A. Lukyanov, and Fyodor A. Kondrashov. Local fitness landscape of the green fluorescent protein. *Nature*, 533(7603):397–401, May 2016. ISSN 1476-4687. doi: 10.1038/nature17995.

[22] Kotaro Tsuboyama, Justas Dauparas, Jonathan Chen, Elodie Laine, Yasser Mohseni Behbahani, Jonathan J. Weinstein, Niall M. Mangan, Sergey Ovchinnikov, and Gabriel J. Rocklin. Mega-scale experimental analysis of protein folding stability in biology and design. *Nature*, 620 (7973):434–444, 2023. doi: 10.1038/s41586-023-06481-2.

[23] David Ding, Ada Y. Shaw, Sam Sinai, Nathan Rollins, Noam Prywes, David F. Savage, Michael T. Laub, and Debora S. Marks. Protein design using structure-based residue preferences. *Nature Communications*, 15(1):1639, 2024.

[24] C. Anders Olson, Nicholas C. Wu, and Ren Sun. A comprehensive biophysical description of pairwise epistasis throughout an entire protein domain. *Current Biology*, 24(22):2643–2651, 2014. doi: 10.1016/j.cub.2014.09.072.

[25] Philip A. Romero, Tuan M. Tran, and Adam R. Abate. Dissecting enzyme function with microfluidic-based deep mutational scanning. *Proceedings of the National Academy of Sciences*, 112(23):7159–7164, 2015. doi: 10.1073/pnas.1422285112.

[26] Eric M Jones, Nathan B Lubock, AJ Venkatakrishnan, Jeffrey Wang, Alex M Tseng, Joseph M Paggi, Naomi R Latorraca, Daniel Cancilla, Megan Satyadi, Jessica E Davis, M Madan Babu, Ron O Dror, and Sriram Kosuri. Structural and functional characterization of g protein–coupled receptors with deep mutational scanning. *eLife*, 9, October 2020. ISSN 2050-084X. doi: 10.7554/elife.54895. URL http://dx.doi.org/10.7554/eLife.54895.

[27] Song Sun, Jochen Weile, Marta Verby, Yingzhou Wu, Yang Wang, Atina G. Cote, Iosifina Fotiadou, Julia Kitaygorodsky, Marc Vidal, Jasper Rine, Pavel Ješina, Viktor Kožich, and Frederick P. Roth. A proactive genotype-to-patient-phenotype map for cystathionine beta-synthase. *Genome Medicine*, 12(1), January 2020. ISSN 1756-994X. doi: 10.1186/s13073-020-0711-1. URL http://dx.doi.org/10.1186/s13073-020-0711-1.

[28] Noelia Ferruz, Steffen Schmidt, and Birte Höcker. ProtGPT2 is a deep unsupervised language model for protein design. *Nature Communications*, 13(1), July 2022. ISSN 2041-1723. doi: 10.1038/s41467-022-32007-7.

[29] Zhidian Zhang, Hannah K. Wayment-Steele, Garyk Brixi, Haobo Wang, Dorothee Kern, and Sergey Ovchinnikov. Protein language models learn evolutionary statistics of interacting sequence motifs. *Proceedings of the National Academy of Sciences*, 121(45), October 2024. ISSN 1091-6490. doi: 10.1073/pnas.2406285121.

[30] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A. A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, July 2021. ISSN 1476-4687. doi: 10.1038/s41586-021-03819-2.

[31] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, March 2023. ISSN 1095-9203. doi: 10.1126/science.ade2574.

 https://doi.org/10.52202/085713-0570

Appendix

A Bounding Speedups

Definition A.1. Let c_e be the cost coefficient for one iteration of SpecMER. Then,

$$c_e = \frac{cM_p + M_k}{M_q}, \tag{7}$$

where M_k is the time it takes the k-mer scoring function to score c candidates. Assuming M_p can batch generate candidates, then the cost of M_p grows sublinearly with respect to c. Additionally, $M_k \ll M_p$, and it follows that:

$$c_e = \frac{\xi M_p}{M_q}, \tag{8}$$

where $1 \leq \xi < c$ is the batch generation cost. Under true parallelism, $\xi = 1$ and the expected improvement factor remains the same as in Equation (1).

Proposition A.2. *The expected (batch) wall-time speedup factor of batch generation is*

$$S(\gamma) \approx \frac{1-\alpha^{\gamma+1}}{(1-\alpha)[c_e+1]}. \tag{9}$$

Proof. Let $T_{\text{iteration}}(\gamma)$ be the time it takes for one iteration of draft selection and evaluation to complete on γ draft tokens. Similarly, let $T_p(\gamma)$, $T_q(\gamma)$, and T_k be the time it takes for the draft model (single candidate), target model, and k-mer scoring function to complete one iteration.

Since $T_p(\gamma)$ can batch generate, $T_p(\gamma) = \xi T_p(\gamma)$, whereas in the serial case, $T_p(\gamma) = cT_p(\gamma)$. The total time for an iteration is then $T_{\text{iteration}}(\gamma) = [\xi T_p(\gamma) + T_k] + T_q(\gamma)$, with an expected speedup of:

$$S(\gamma) = \frac{1-\alpha^{\gamma+1}}{(1-\alpha)(\frac{T_{\text{iteration}}(\gamma)}{T_q(\gamma)})}. \tag{10}$$

Let us define the cost coefficient $c_{\text{draft}}(\gamma) = \frac{\xi T_p(\gamma)+T_k}{T_q(\gamma)}$. By Equation (8), $c_{\text{draft}}(\gamma) \approx c_e$, simplifying the ratio between total iteration time and target iteration time $\frac{T_{\text{iteration}}(\gamma)}{T_q(\gamma)} \approx c_e + 1$. Consequently,

$$S(\gamma) \approx \frac{1-\alpha^{\gamma+1}}{(1-\alpha)[c_e+1]}. \tag{11}$$

In the case that generation is serial, the wall-time speedup results in a similar equation.

Corollary A.3. *The expected serial wall-time speedup factor is:*

$$S(\gamma) \approx \frac{1-\alpha^{\gamma+1}}{(1-\alpha)[\frac{c}{\xi}c_e+1]}. \tag{12}$$

We note that $T_p(\gamma)$ and $T_q(\gamma)$ are functions of γ as the time it takes to decode a sequence of length γ does not scale linearly. In practice, generating short sequences approximately scales linearly with γ, in which near exact equality is achieved in Equation (9) and Equation (12). Increasing the acceptance ratio leads to large gains in time saved across all configurations of γ.

B Hyperparameters

B.1 Implementation Details

Code for SpecMER is publicly available here: `https://github.com/amirgroup-codes/SpecMER.git`.

ProGen2 [4] is a decoder-only Transformer architecture with four different parameter sizes: small - 151M; medium/base - 764M; large - 2.7B; xlarge - 6.4B. No model training is performed in this work; inference is run on a server with eight NVIDIA A6000 GPUs.

We conducted extensive testing to find the correct parallel implementation of candidate sequence generation. Batch generation, while not exactly parallel, yielded the best results in regard to tokens per second. We estimate that batch generation took about $1.2 - 1.3\times$ longer to generate $c = 5$ sequences as compared to $c = 1$ sequences, incurring additional latency due to larger batch operations. This discrepancy in generation time resulted in a larger variance in tokens per second as c increased. When a speculated token is rejected, decoding takes longer, and the added cost of batch generation appears in the variance. Due to the non-deterministic nature of sampling, some generations have higher acceptance rates than others; therefore, occasional rises and dips in acceptance have a larger impact on tokens per second than producing a single sequence at a time. We implemented SpecMER on top of the sample method implementation provided by ProGen2. We considered attempting a fully parallel implementation of batched sequence generation; however, we ultimately decided to defer to the default implementation. This was due to hardware limitations and practical considerations. A fully parallel multi-GPU system, while likely to yield a higher tokens per second, is significantly more demanding for practitioners who wish to use SpecMER. We implemented SpecMER with single GPU instances in mind.

In addition to testing configurations for parallel implementations, we tested KV caching versus full rescoring during each iteration of SpecMER. In a normal setting, KV caching would almost always be preferable. However, due to the maximal coupling acceptance logic, full rescoring versus KV caching depended on the expected performance of the decoder. We stored KV caches for the draft model and target model, updating at each iteration; however, if any token was rejected and resampled using the logic in Algorithm 1, that token would require a full forward pass. Since correction tokens are resampled from the residual distribution, the corresponding logits must be recomputed with a full forward pass. This poses a tradeoff: either compute a full uncached forward pass for both the draft and target model at each iteration, or compute the cached forward pass at the risk of incurring an extra forward pass for every rejected token. We observed that as long as the speculative decoder achieved an acceptance ratio of $\approx 80\%$ or higher, the KV cache implementation was faster. For this reason, we chose to use a custom implementation of KV caching for SpecMER. We recognize that our implementation of SpecMER is not fully optimized and could be improved to realize even faster speedups. We leave this task for future work, or for specialized implementations with access to better computing hardware.

B.2 Trade-off Space

Selecting the number of candidates to draft c comes at the cost of a trade-off between generation speed and generation quality. Figure 3 details how c influences generation speed and misranking error ϵ. Setting $c = 3$ yielded the best tradeoff in this space, demonstrating the largest gain in likelihood improvements for the least decrease in tokens per second. We observed an improvement in misranking error for SpecMER, indicating that structural cues from SpecMER not only increase structural plausibility but also help close the distribution gap between the target and draft model. The target model in protein generation often exhibits higher correlation with functional and structural benchmarks, learning these auxiliary objectives implicitly from millions of protein sequences. This correlation increases as a function of the number of parameters. SpecMER illustrates that this auxiliary gap between the target and draft can be supplanted with external functional or structural information.

B.3 Results From Hyperparameter Sweep

This section details how each hyperparameter impacted acceptance and likelihood. As stated in the main text, we swept over the following hyperparameters: draft tokens (γ), temperature (T), and k-mers (k). We tested every combination of the following configurations: $\gamma \in \{5, 10, 15\}$, $T \in \{0.7, 1, 1.4\}$, $k \in \{(1), (3), (1, 3), (1, 3, 5)\}$. We chose this consolidated list to encompass realistic hyperparameters used for protein generation. For each configuration, we generated 200 protein sequences, stopping generation at max sequence length (length of the wild-type for the context being used) or upon generation of a stop token ("2" in the case of ProGen2). Depending on the protein selected, testing a single configuration could last anywhere between $2 - 30$ minutes to generate 200 full-length sequences. The final hyperparameter set used to report results in Table 2 is listed in Table 6.

 https://doi.org/10.52202/085713-0570

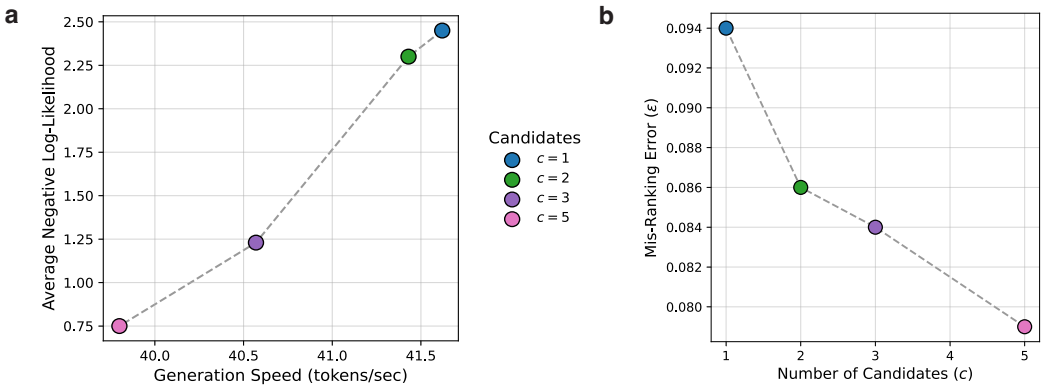


Figure 3: Trade-off space between number of candidates, misranking error, and negative log-likelihood (NLL). **a**, Generation speed versus NLL. As c increases, NLL drastically increases at the penalty of reduced generation speed. **b**, Number of candidates versus misranking error ϵ. As c increases, misranking error decreases, indicating the influence of k-mers on selecting candidates that maximize acceptance while also maximizing the k-mer score.

Table 6: Final hyperparameter configurations for each protein corresponding to results in Table 2.

Protein	**Temperature**	**Draft Tokens**	k **value**	**Candidates**
Bgl3	1.0	5	3	5
GFP	0.7	5	1, 3	5
RBP1	1.0	10	3	5
GB1	1.4	10	1, 3, 5	5
ParD3	1.0	5	1, 3, 5	5
CBS	0.7	5	1, 3, 5	5
ADRB2	0.7	5	1, 3	5

We observed variance in generated sequence lengths, largely dependent on temperature. Sharper temperatures ($T = 0.7$) led to longer sequences on average, shifting probability mass away from producing a stop token early. Sequence likelihoods were heavily influenced by T, with likelihood increasing on average when temperature decreased. This is not an unexpected result: decreasing temperature with top-p sampling results in a more deterministic sampling procedure. With this in mind, we chose to generate sequences over a range of temperature values, as increased sequence likelihood does not always result in more plausible structures.

We noticed that each protein had subtle preferences toward the selection of γ. Shorter proteins, such as ParD3 and GB1, preferred $\gamma = 5$, while longer proteins, such as GFP, saw improvements with $\gamma = 15$. Bgl3 and RBP1 showed no significant preference toward this hyperparameter. Plots pertaining to the effect of T, k, and c are available at the conclusion of the appendix.

For additional context to the main results, we include the pLDDT and NLL measurements for each protein in Table 7.

Table 7: Negative log-likelihood (NLL) and pLDDT scores for each wild-type sequence. Missing values indicate proteins for which structural predictions were not available.

Protein	**NLL**	**pLDDT**
CBS	0.75	–
Bgl3	0.92	–
ADRB2	1.31	–
ParD3	2.11	0.79
GB1	2.52	0.82
RBP1	2.63	0.83
GFP	2.93	0.42

C MSA Ablations

We performed two ablation studies to test the efficacy of k-mers in SpecMER.

Cross-protein k-mers. We tested SpecMER using mismatched evolutionary signals: (1) generate conditioned on GFP with GB1-derived k-mers, and (2) generate conditioned on GB1 with Bgl3-derived k-mers. The results are shown in Table 8. In both cases, we observed a drop in likelihoods, both on average and for the top-20 highest-likelihood sequences, compared to the protein-specific results in Table 4. This suggests that protein-specific k-mers are responsible for the increase in likelihood observed with SpecMER.

Table 8: Cross-protein k-mer ablation results.

Condition	Mean NLL	Top-20 NLL
GFP + GB1 k-mers	2.52 ± 0.27	1.78 ± 0.23
GB1 + Bgl3 k-mers	2.79 ± 0.10	2.59 ± 0.11

MSA depth. We next tested the importance of MSA depth by generating Bgl3 proteins using only 1,000 sequences from the MSA instead of the full set of $\sim$130k sequences. This reduction limited the number of k-mers available to guide generation. The top-20 sequences yielded an average NLL of 1.56 ± 0.20, compared to 0.63 ± 0.11 for SpecMER ($c = 5$) with the full-depth MSA. These results support our conclusion that SpecMER's performance can degrade when informative motifs are sparse or unavailable.

D Structural Scores and Embeddings

This section describes experiments pertaining to structural scores, diversity metrics, and embeddings for generated protein sequences. For each section, metrics are computed over all tested configurations of vanilla speculative decoding and SpecMER.

D.1 Sequence Diversity

Novelty is a key metric for determining how well generative models sample from the space of learned protein sequences. We computed two diversity metrics to assess how SpecMER affects sequence novelty: WT Hamming distance and inter-sequence Hamming distance. WT Hamming distance illustrates how many edits away the generated protein is from the wild-type. Inter-sequence illustrates how similar sequences generated under the same configuration are. Results for both metrics are reported in Table D.1.

Table 9: Wild-type (WT) distance and inter-sequence (Inter-Seq) distance across proteins. SpecMER generates sequences further from the WT while maintaining inter-sequence diversity, thereby exploring sequence space while producing plausible protein sequences.

Protein	WT Dist. (SpecMER)	WT Dist. (Spec. Dec.)	Inter-Seq (SpecMER)	Inter-Seq (Spec. Dec.)
GFP	208.35 ± 5.76	208.49 ± 4.91	181.78 ± 27.14	184.56 ± 27.14
RBP1	41.27 ± 3.48	42.81 ± 3.43	42.60 ± 3.87	44.88 ± 4.00
ParD3	75.97 ± 3.01	78.68 ± 2.41	67.39 ± 6.87	70.00 ± 5.41
GB1	44.70 ± 3.25	45.27 ± 3.27	46.47 ± 4.46	46.99 ± 3.86
Bgl3	324.88 ± 30.88	333.12 ± 33.96	261.02 ± 42.28	284.97 ± 40.44
CBS	378.64 ± 140.60	431.00 ± 106.05	291.84 ± 161.60	457.27 ± 40.98
ADRB2	263.80 ± 120.22	290.18 ± 109.01	270.82 ± 93.46	340.28 ± 47.66

We observed that SpecMER followed the same trends as speculative decoding for both WT and inter-sequence distance. Both methods generate many dissimilar designs far away from the WT, ensuring that sequence generation explores novel sequence space, a key advantage of autoregressive PLM generation.

https://doi.org/10.52202/085713-0570

D.2 Embeddings

We assessed embeddings of sequences generated from vanilla speculative decoding and SpecMER. Embeddings were generated using ESM-2 8M [31] with mean pooling and unaligned sequences. ESM-2 is a protein language model that embeds sequences that are functionally or structurally similar closer together. For each protein used as context for conditional generation, we computed embeddings for its respective MSA. Sequences from the MSA are expressed in nature, providing an anchor point for which sequences are likely to be biologically plausible. We used PCA to compare embeddings over two principal components, finding that sequences generated by SpecMER were closer in proximity to the MSA sequences than vanilla speculative decoding. We deduced from this analysis that k-mers formed from the MSA did guide generation toward biologically plausible sequences. However, this analysis alone does not provide enough evidence to support this claim, as embedding distance and PCA are approximate metrics for sequence plausibility. To further corroborate claims of plausibility, we computed structural confidence scores as detailed in the next section. Embedding plots for each protein can be found at the conclusion of the appendix.

D.3 pLDDT Scores

We assessed pLDDT [30] scores as a structural confidence measure. pLDDT ranges from $0 - 100$ at each amino acid site, with 0 indicating a structure very unlikely to fold, and 100 indicating a structure very likely to fold. We computed pLDDT scores using ESMFold [31], a structure prediction model that provides per-residue pLDDT scores. To assess the overall structural confidence of generated proteins, we averaged the pLDDT scores across all amino acid positions. Figure 2b details pLDDT scores from top configurations of vanilla speculative decoding and SpecMER for different c values, with generation conditioned on RBP1. For each candidate c, we identified the three best configurations for each method, determined by average log-likelihood. Furthermore, we selected the top-100 most likely sequences from these three configurations, resulting in a set of 300 data points for each c value. This was done to ensure diversity in the set of sampled sequences. We observed that as c increased, so did the average pLDDT score. This indicates a clear relationship between increasing the influence of k-mers and generating sequences with higher biological plausibility.

We computed average (Table 3) pLDDT scores for all proteins with the exception of Bgl3. This is due to the sequence length, which exceeds 400, and is therefore incompatible with ESMFold. For the remaining proteins, we selected the top 300 sequences across the three best configurations for each method as described before. We observed that sequences generated using SpecMER yielded the highest pLDDT scores, with the exception of GFP, which demonstrated the highest pLDDT scores under vanilla speculative decoding. This was a surprising observation, as conditional generation using GFP as the context sequence yielded higher log-likelihoods when using SpecMER. However, top-5 statistics indicated higher-end scores were comparable for GFP. A summary of top-5 pLDDT scores can be found in Table 10. We observed a similar trend with top-5 and average pLDDT scores. That is, increasing c led to higher pLDDT scores, with the exception of GFP, which was competitive across all values of c.

Table 10: Top-5 pLDDT scores across four different proteins. Sequences are collected from the three best configurations for each decoding method, determined by the highest average log-likelihood under the target model (ProGen2-M).

Protein	**Speculative Decoding** ($c = 1$)	**SpecMER** ($c = 2$)	**SpecMER** ($c = 3$)	**SpecMER** ($c = 5$)
GFP	0.892 ± 0.020	0.838 ± 0.062	0.888 ± 0.008	0.850 ± 0.035
RBP1	0.864 ± 0.019	0.884 ± 0.016	0.876 ± 0.016	0.888 ± 0.011
ParD3	0.912 ± 0.004	0.918 ± 0.008	0.922 ± 0.004	0.926 ± 0.005
GB1	0.790 ± 0.030	0.704 ± 0.010	0.760 ± 0.059	0.794 ± 0.019

E Multiple Sequence Alignment and K-mer Generation

We collect MSAs of target proteins from ProteinGym [20]. As described in Section 3.2, k-mers are formed by applying a k length sliding window over all sequences in the MSA of a respective protein. Gap characters "-" are ignored.

We observed that the best k value for each protein varied substantially. Some proteins demonstrated little to no preference toward k, such as RBP1 and GB1, whereas GFP was heavily influenced by $k = \{1, 3\}$ and $k = \{1, 3, 5\}$. It is possible that for proteins comprised of large disordered regions, SpecMER would show reduced effectiveness, or even be detrimental to generation. Plots describing how k values affected log-likelihood can be found at the conclusion of the appendix.

 https://doi.org/10.52202/085713-0570

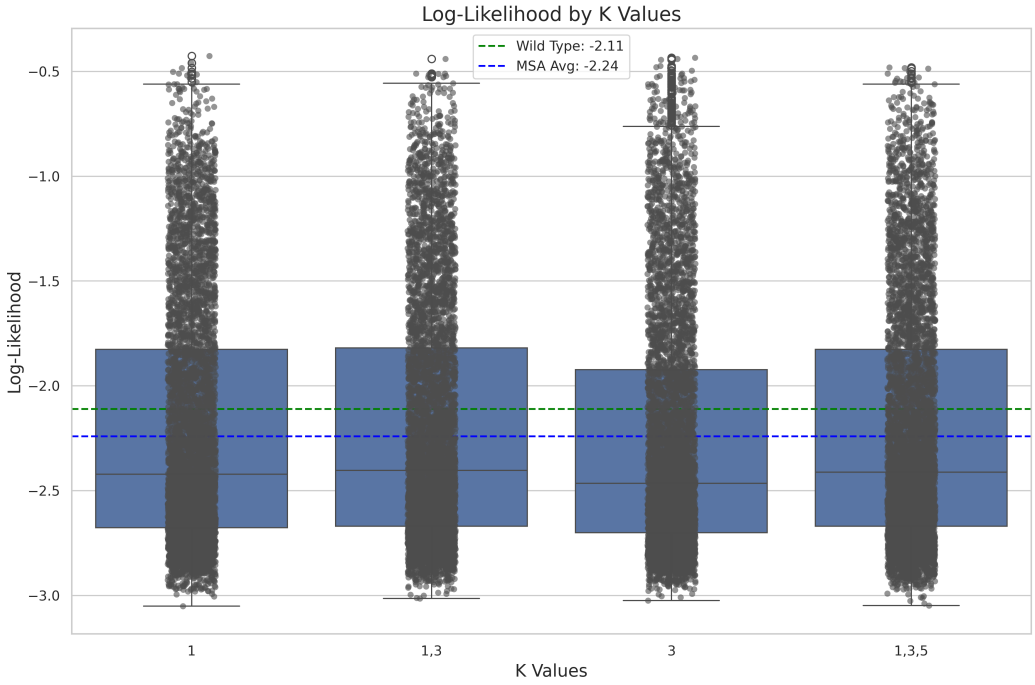


Figure 4: Log-likelihood versus selection of k for ParD3.

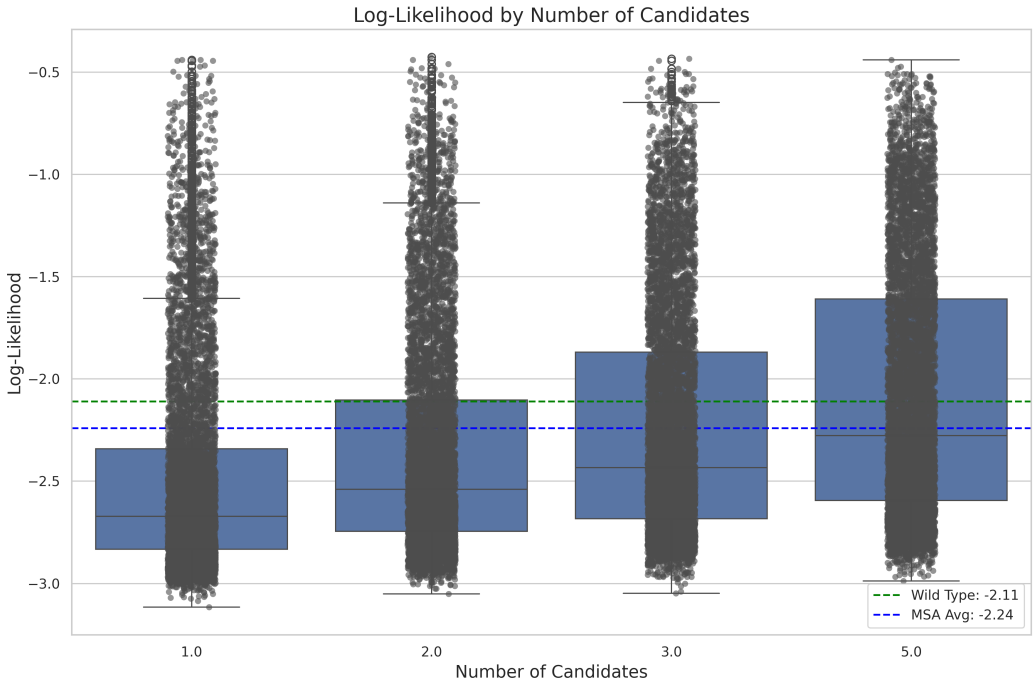


Figure 5: Log-likelihood versus selection of candidates c for ParD3.

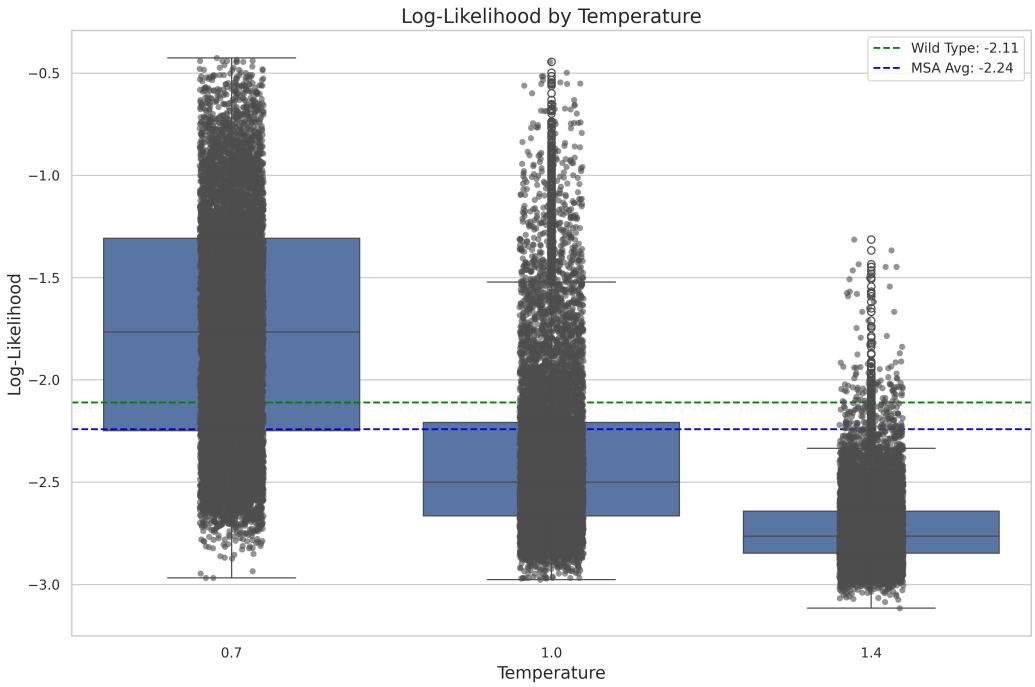


Figure 6: Log-likelihood versus selection of temperature for ParD3.

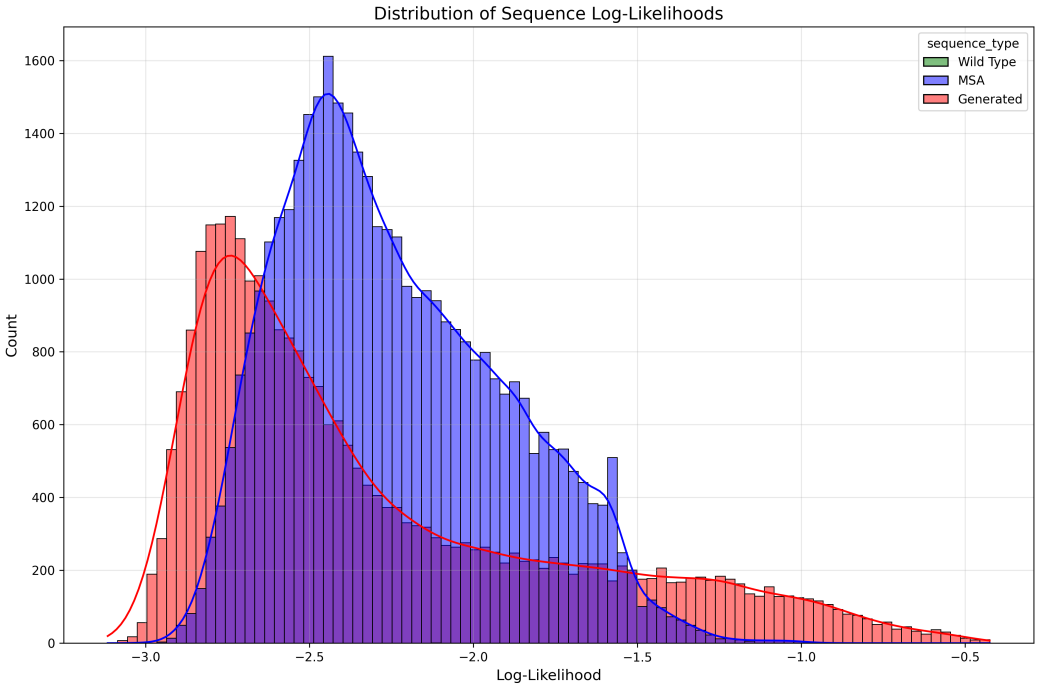


Figure 7: Likelihood distribution of ParD3 sequences generated using SpecMER compared to the likelihood distribution of its collected MSA (scored using ProGen2-M).

https://doi.org/10.52202/085713-0570

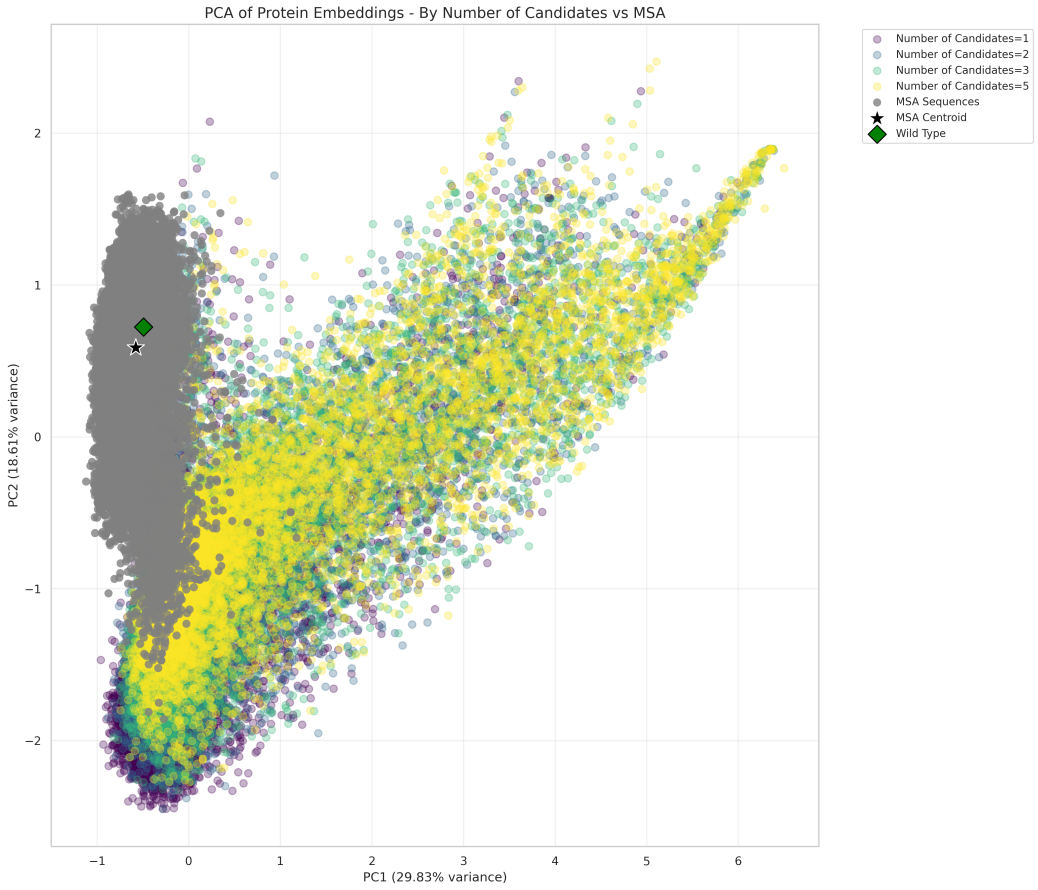


Figure 8: PCA plot for sequence embeddings generated with varying c, with $c = 1$ denoting speculative decoding, and $c > 1$ denoting SpecMER. Sequences are generated using the context of ParD3.

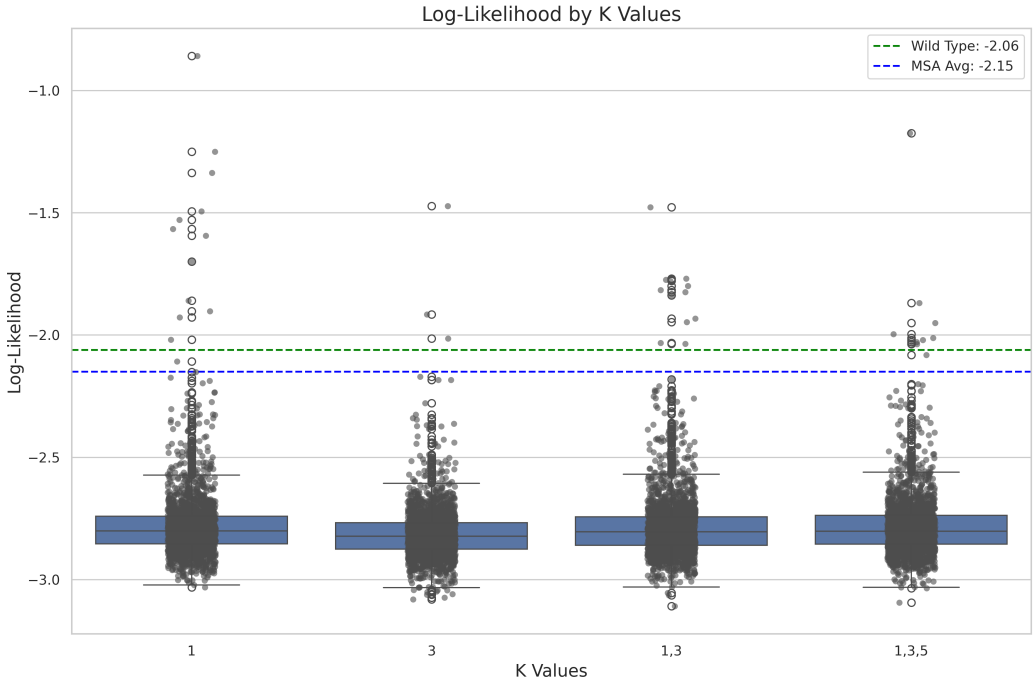


Figure 9: Log-likelihood versus selection of k for GB1.

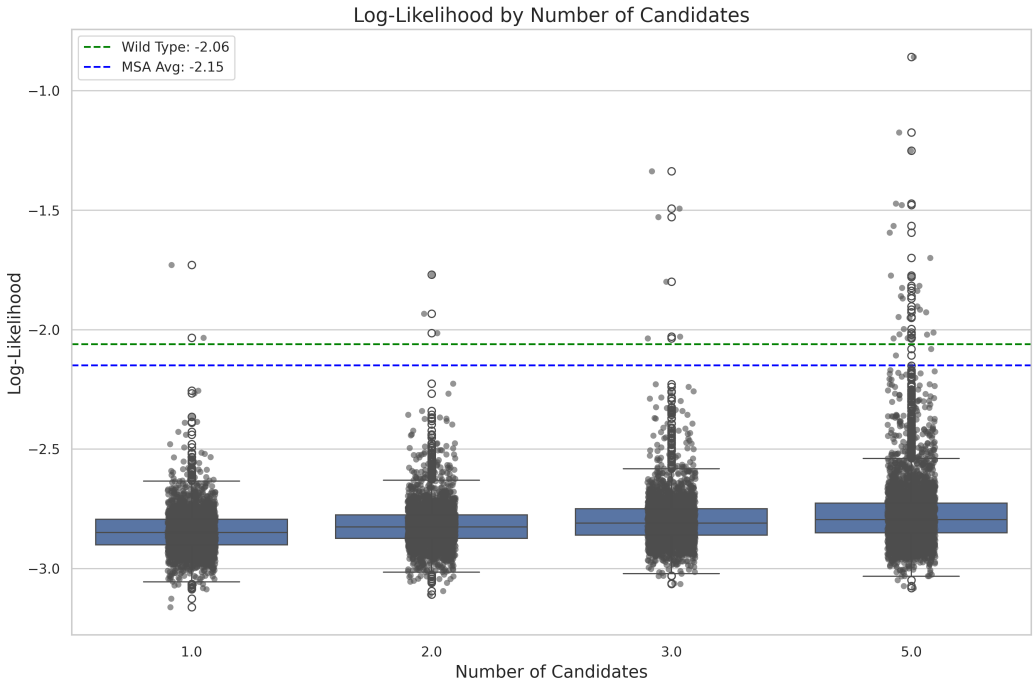


Figure 10: Log-likelihood versus selection of candidates c for GB1.

https://doi.org/10.52202/085713-0570

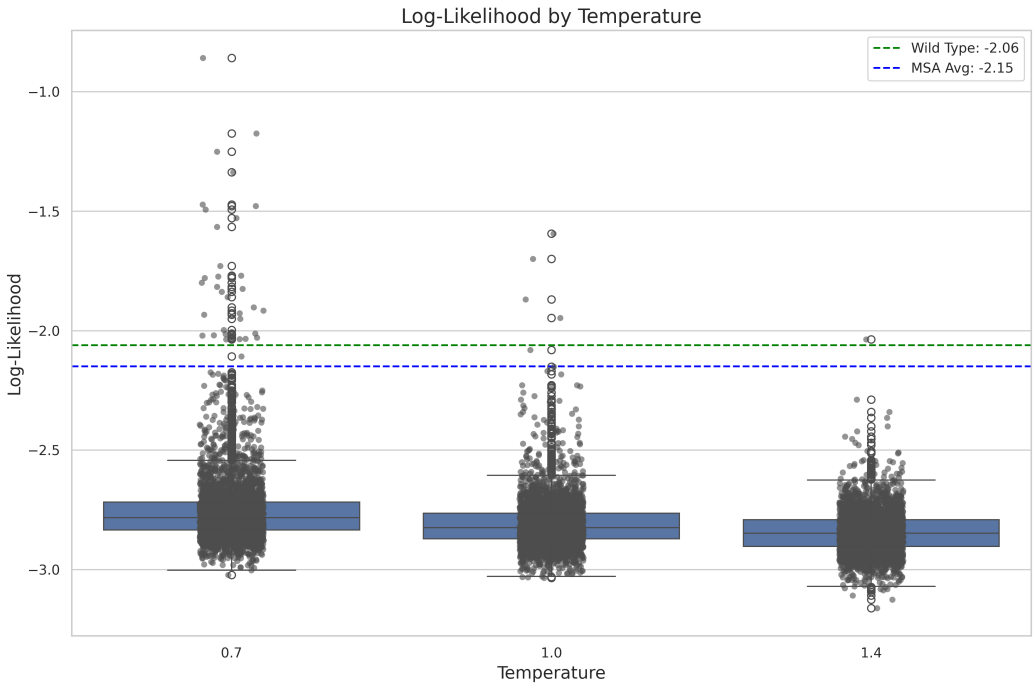


Figure 11: Log-likelihood versus selection of temperature for GB1.

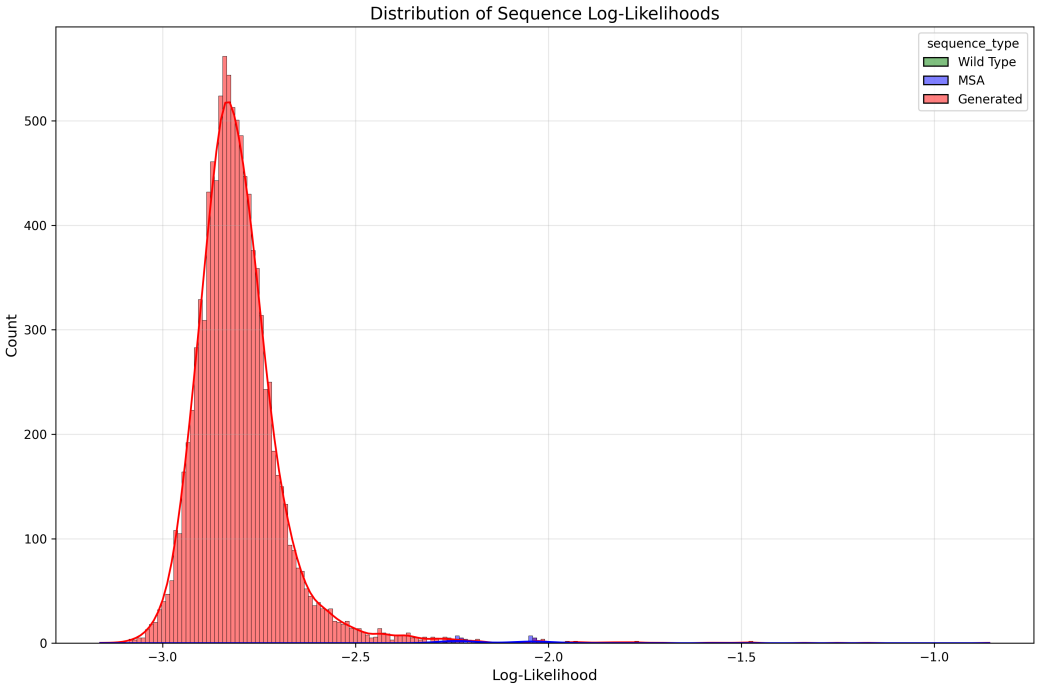


Figure 12: Likelihood distribution of GB1 sequences generated using SpecMER compared to the likelihood distribution of its collected MSA (scored using ProGen2-M).



Figure 13: PCA plot for sequence embeddings generated with varying c, with $c = 1$ denoting speculative decoding, and $c > 1$ denoting SpecMER. Sequences are generated using the context of GB1.

https://doi.org/10.52202/085713-0570

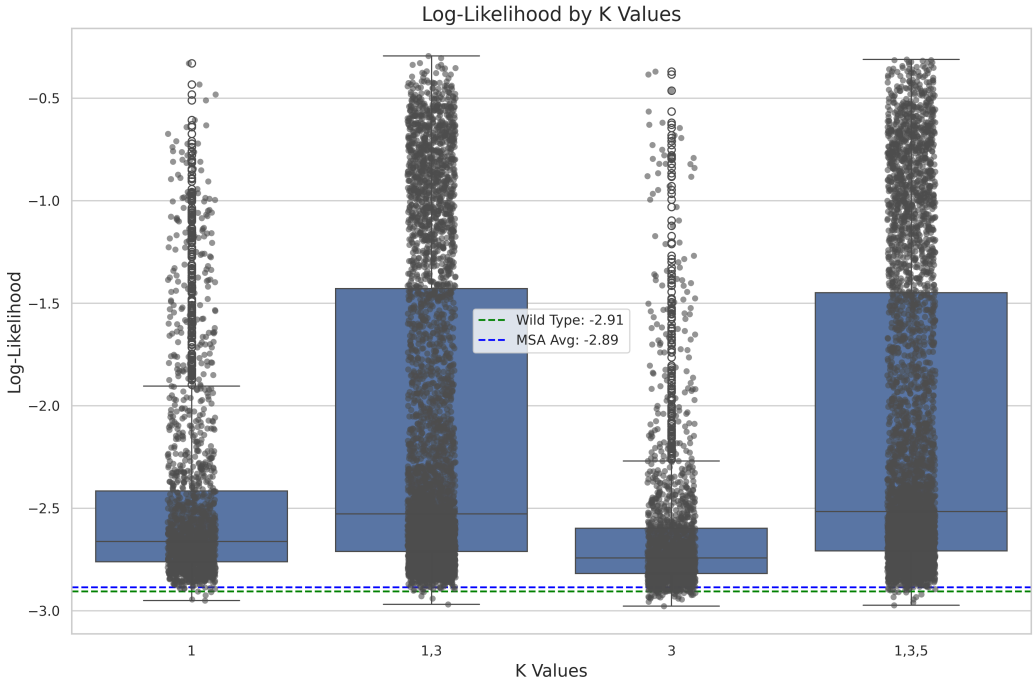


Figure 14: Log-likelihood versus selection of k for GFP.

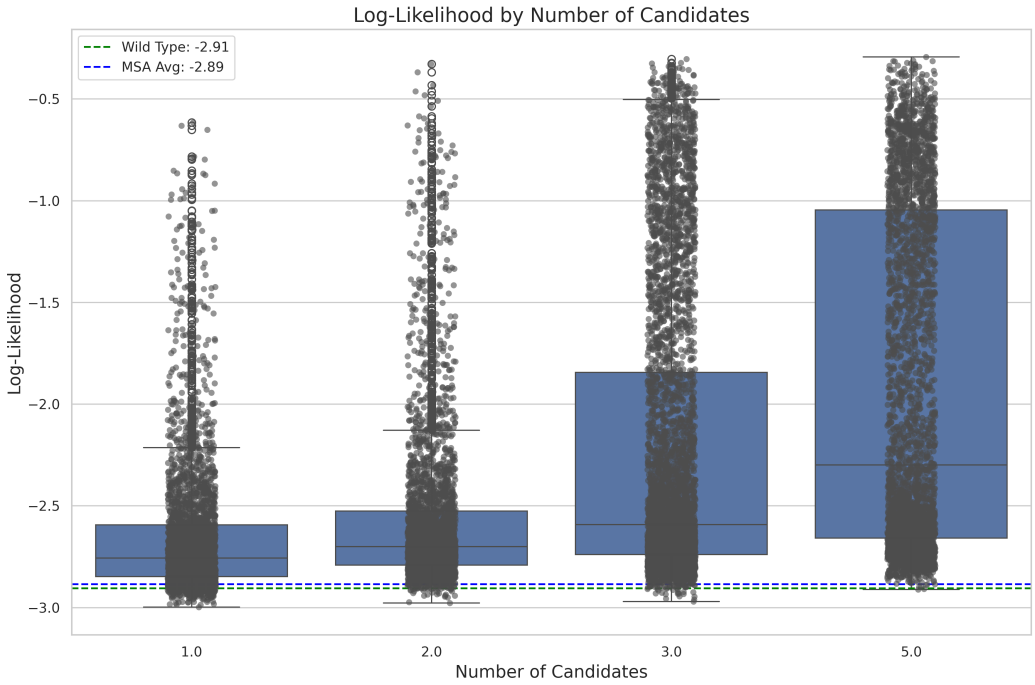


Figure 15: Log-likelihood versus selection of candidates c for GFP.

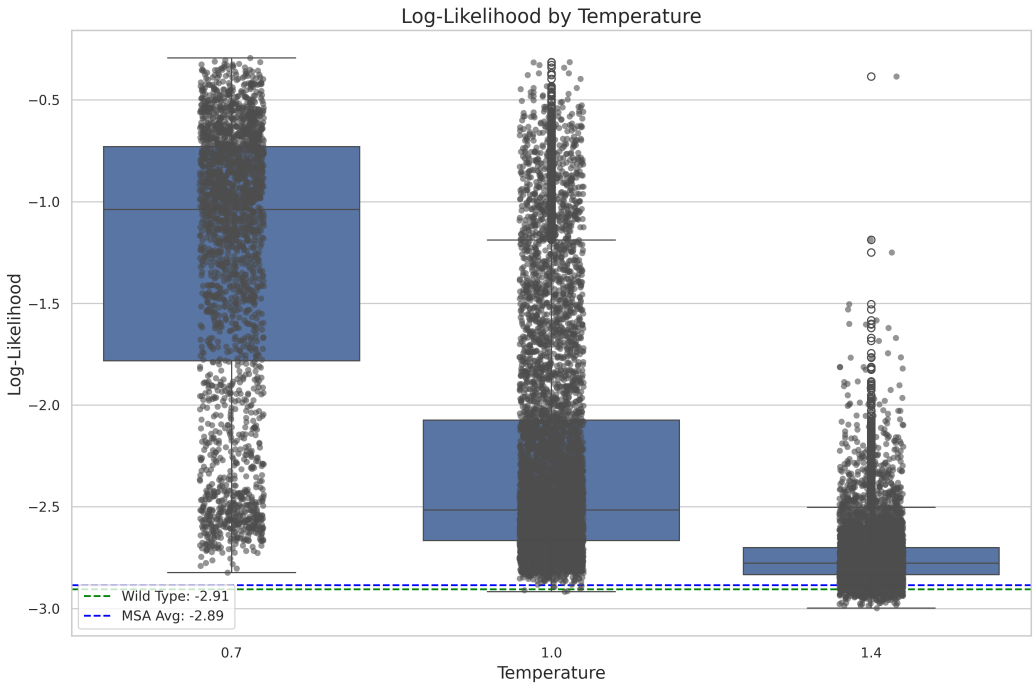


Figure 16: Log-likelihood versus selection of temperature for GFP.

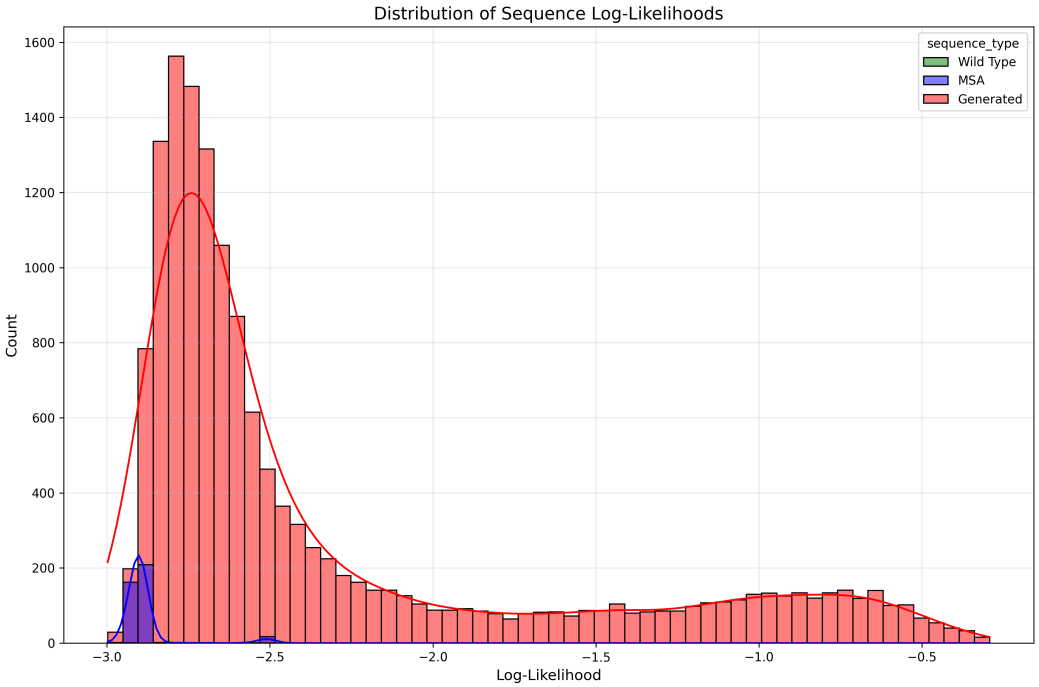


Figure 17: Likelihood distribution of GFP sequences generated using SpecMER compared to the likelihood distribution of its collected MSA (scored using ProGen2-M).

https://doi.org/10.52202/085713-0570

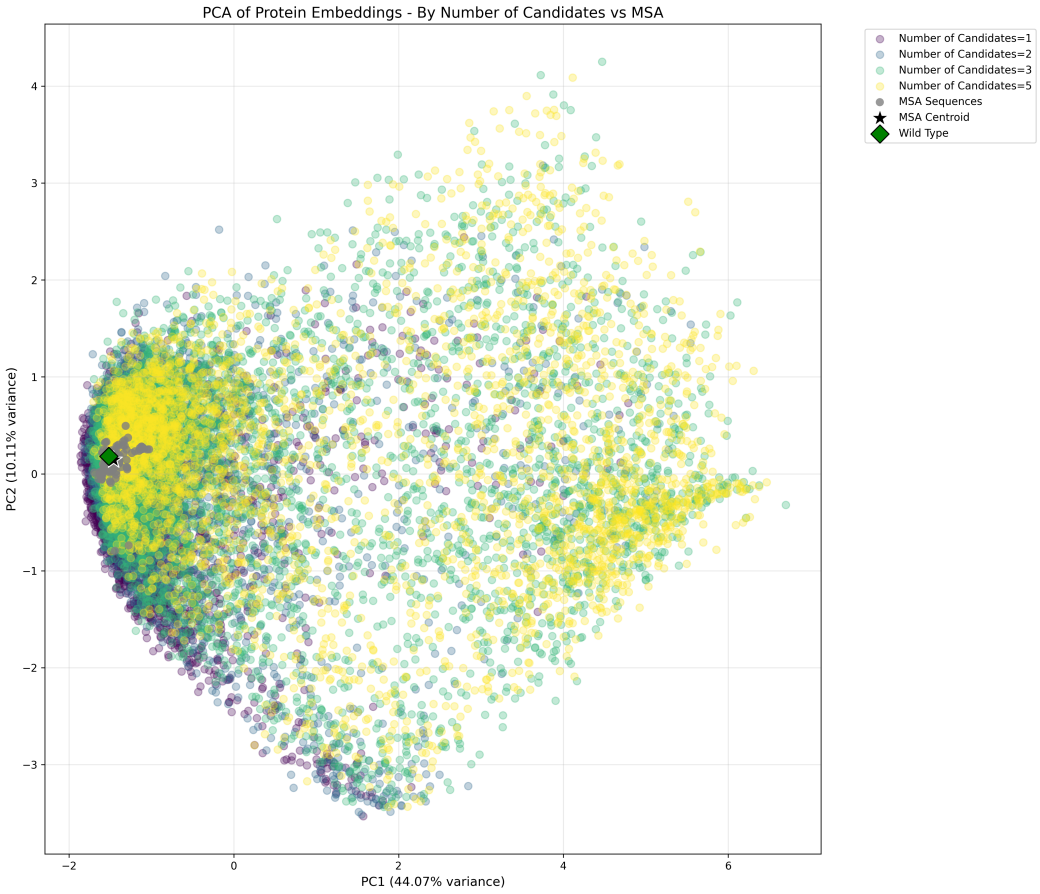


Figure 18: PCA plot for sequence embeddings generated with varying c, with $c = 1$ denoting speculative decoding, and $c > 1$ denoting SpecMER. Sequences are generated using the context of GFP.

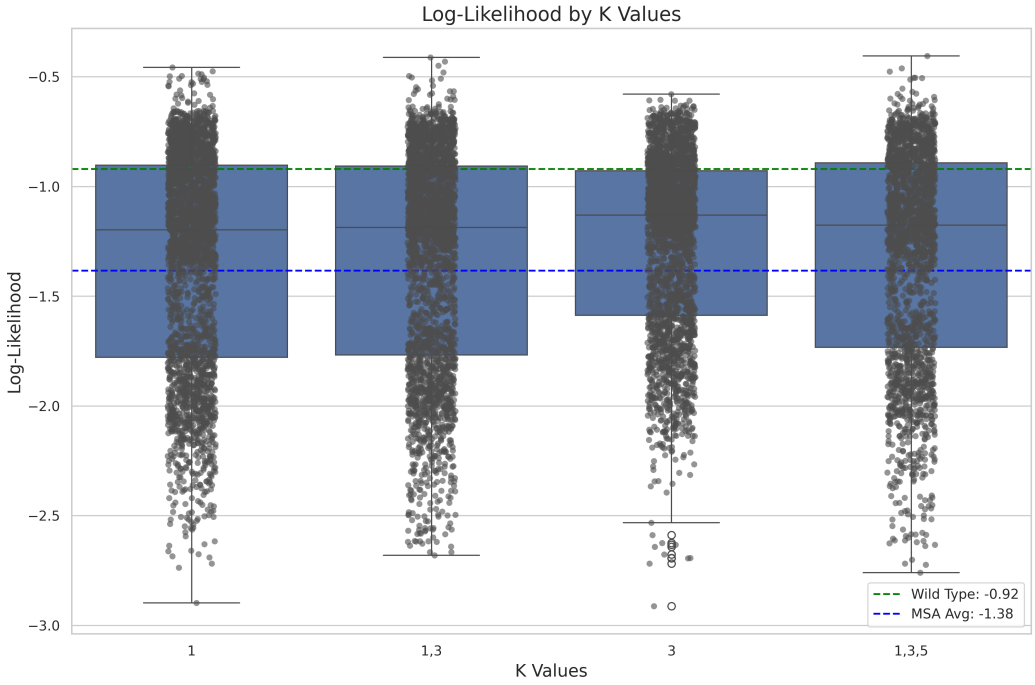


Figure 19: Log-likelihood versus selection of k for Bgl3.

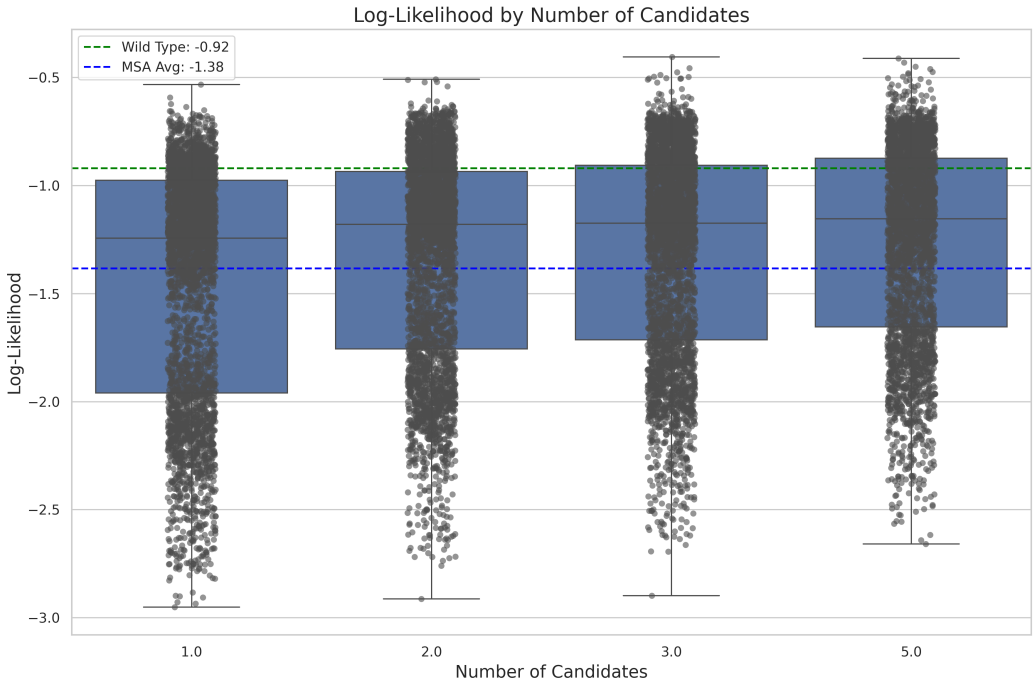


Figure 20: Log-likelihood versus selection of candidates c for Bgl3.

https://doi.org/10.52202/085713-0570

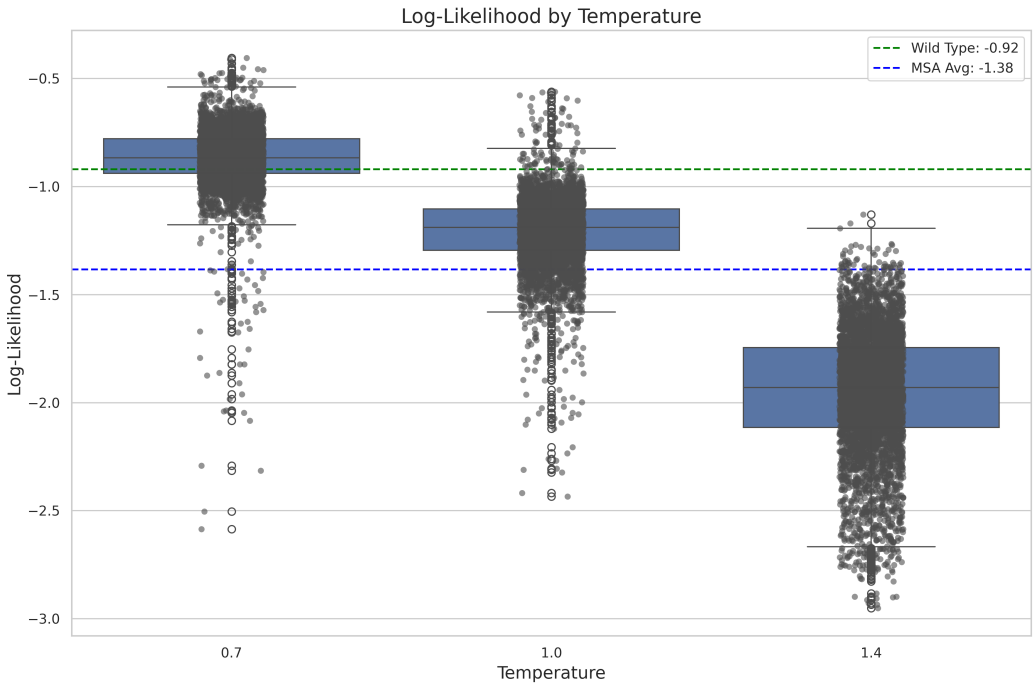


Figure 21: Log-likelihood versus selection of temperature for Bgl3.

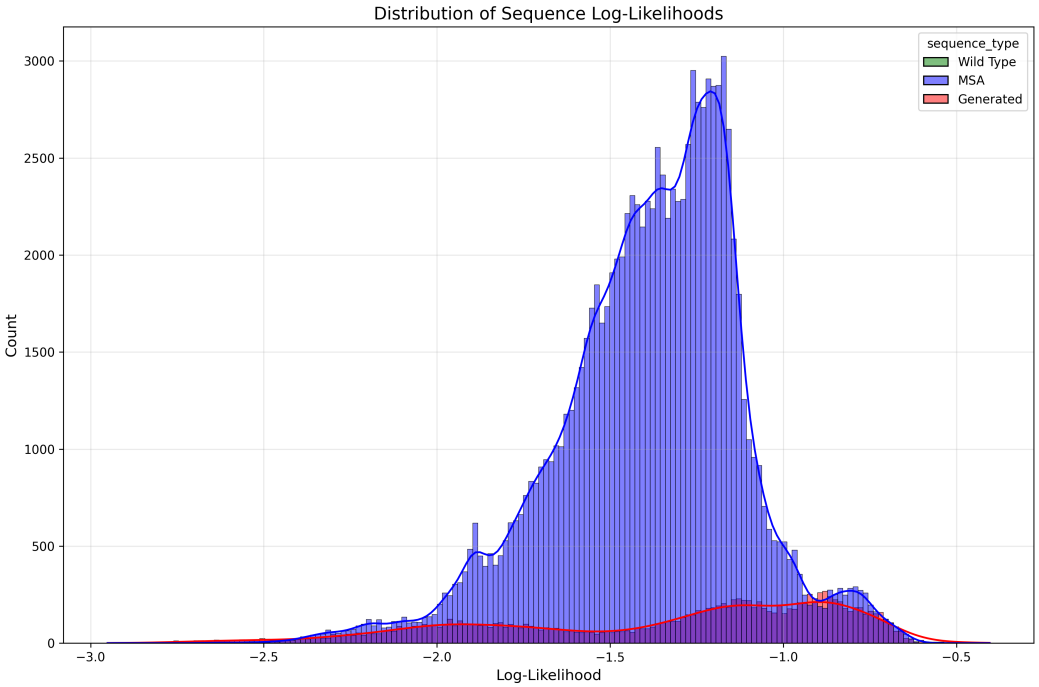


Figure 22: Likelihood distribution of Bgl3 sequences generated using SpecMER compared to the likelihood distribution of its collected MSA (scored using ProGen2-M).

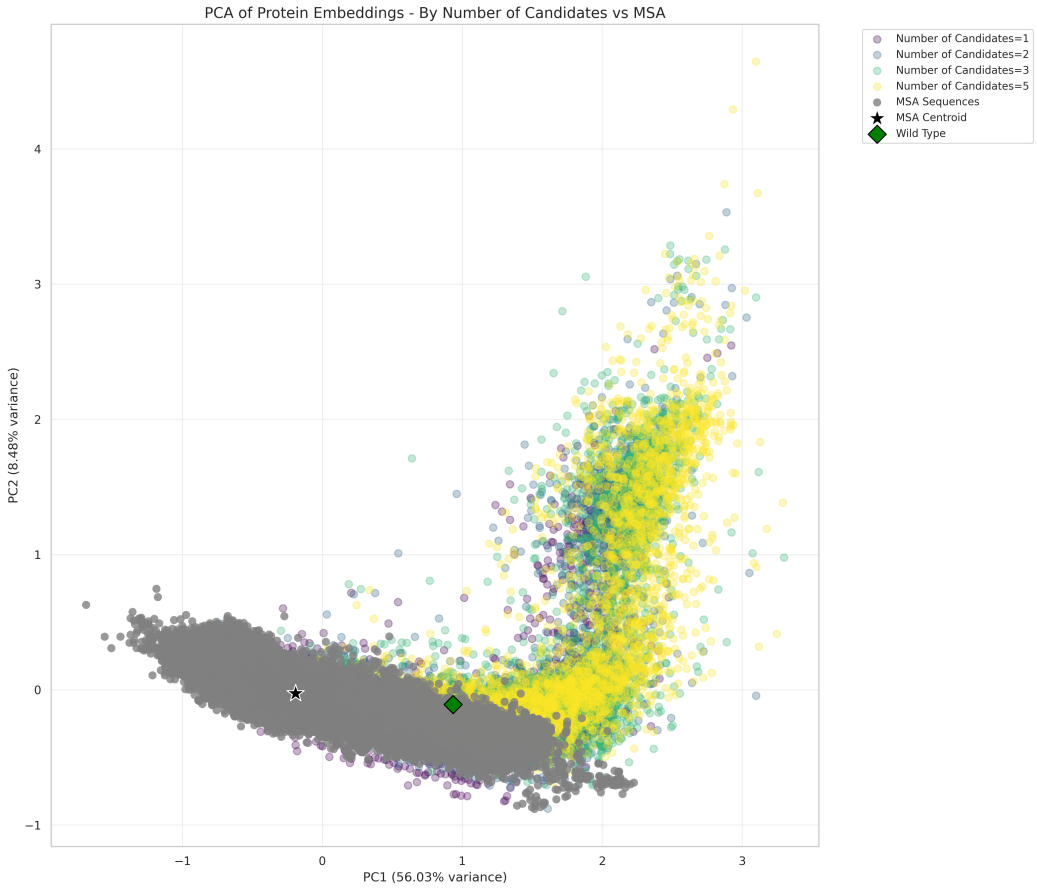


Figure 23: PCA plot for sequence embeddings generated with varying c, with $c = 1$ denoting speculative decoding, and $c > 1$ denoting SpecMER. Sequences are generated using the context of Bgl3.

https://doi.org/10.52202/085713-0570

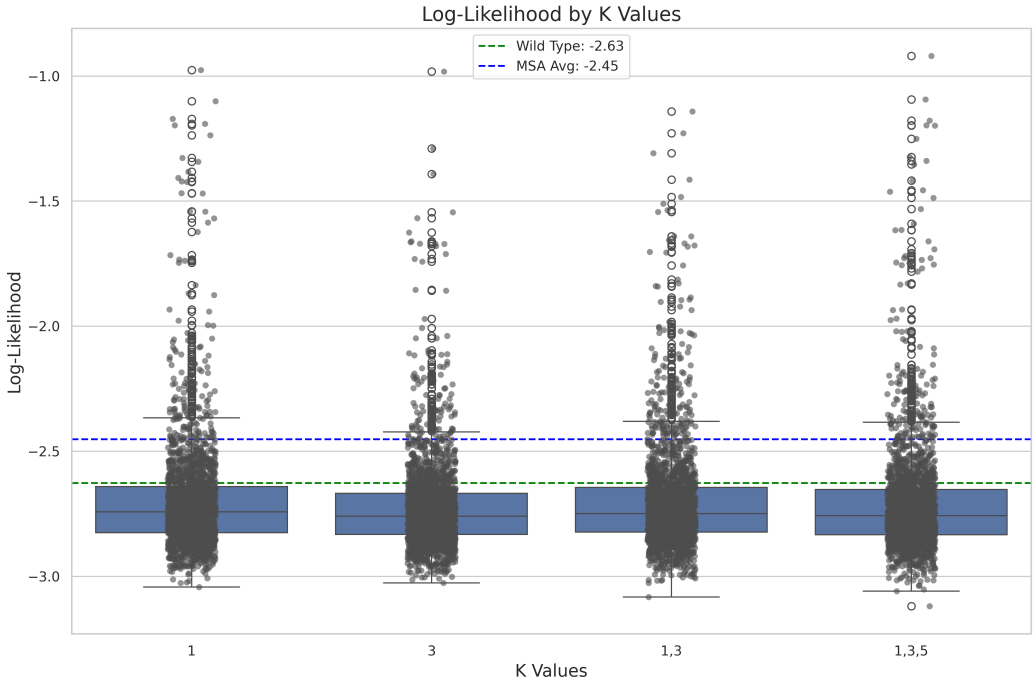


Figure 24: Log-likelihood versus selection of k for RBP1.

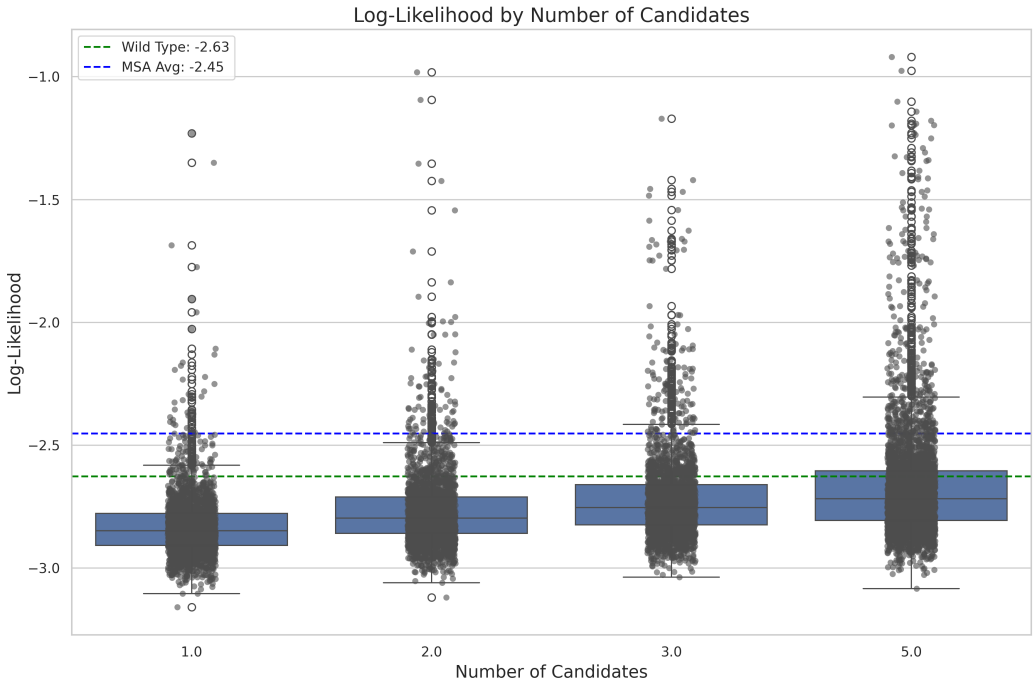


Figure 25: Log-likelihood versus selection of candidates c for RBP1.

https://doi.org/10.52202/085713-0570

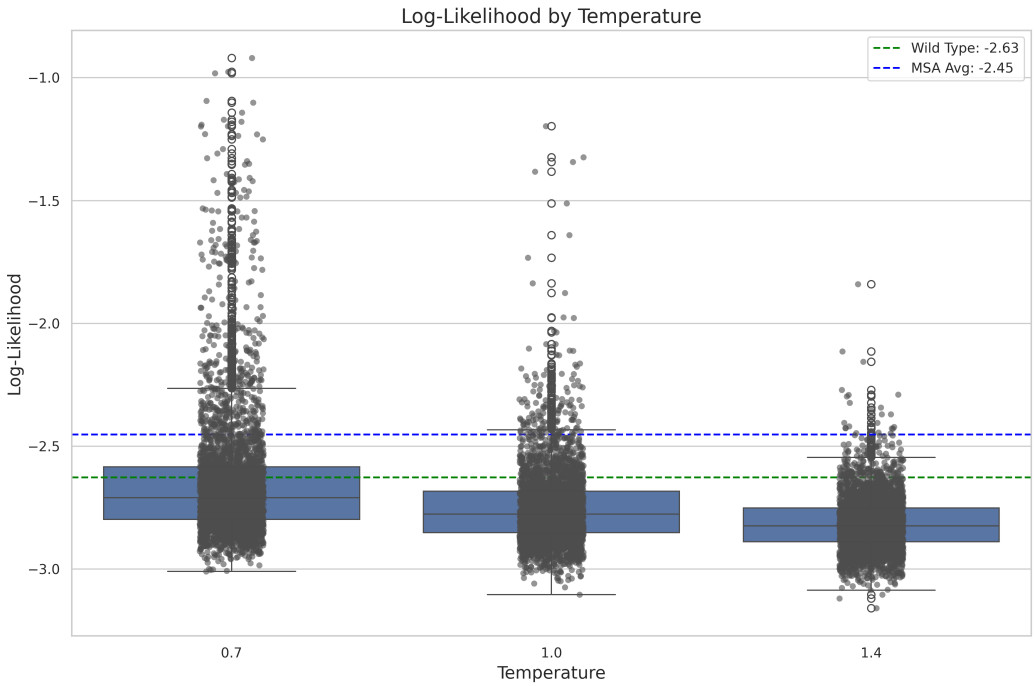


Figure 26: Log-likelihood versus selection of temperature for RBP1.

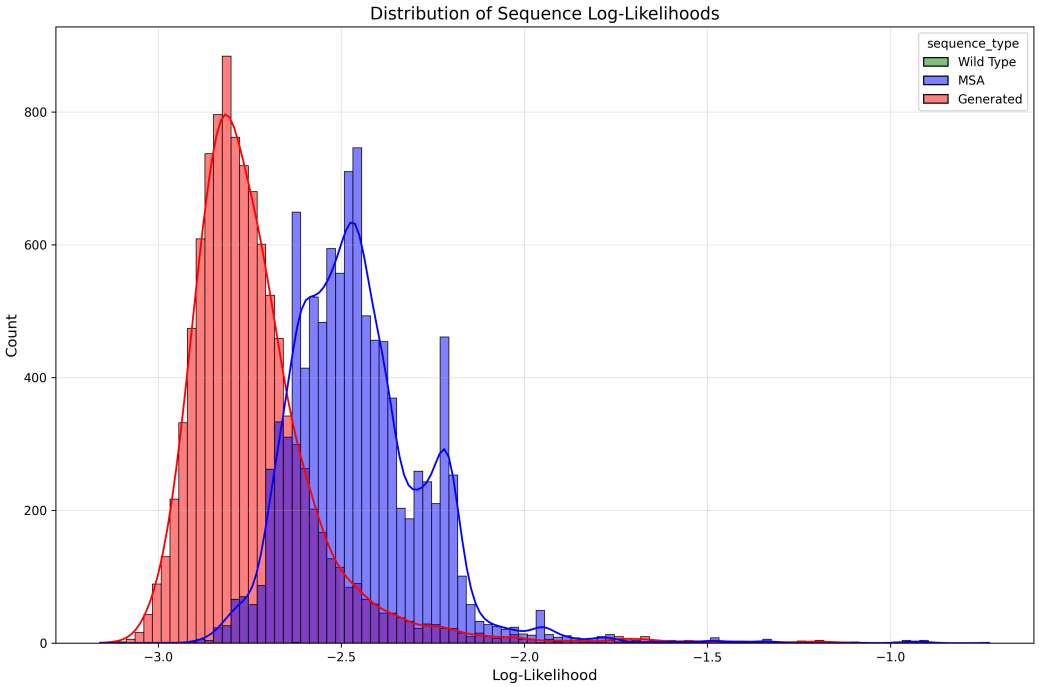


Figure 27: Likelihood distribution of RBP1 sequences generated using SpecMER compared to the likelihood distribution of its collected MSA (scored using ProGen2-M).

https://doi.org/10.52202/085713-0570

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Generation speed claims are supported in Section 4.3 and Tables 2 and 5. Claims of sequence plausibility are corroborated in Section 4.3, Figure 2, and Tables 2 and 3. Theoretical results are proven in Section 3.3, with additional results proven in Appendix A. The scope of claims is centered around protein generation, which is supported by over five different functionally diverse protein sequences.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed thoroughly in the conclusion (Section 5), addressing data, scalability, and hardware limitations.

   Guidelines:

   - The answer NA means that the paper has no limitations, while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Assumptions are described in Section 3.3 as well as Section 2. Proofs can be found in Section 3.3 and Appendix A. Additional assumptions are supported by related works and are properly attributed credit.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The presented algorithm implementation details are described in Section 2, Section 3, Figure 1, and Algorithm 1. Code is released for exact implementation details.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

 https://doi.org/10.52202/085713-0570

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code is made available on GitHub, and can be found in Section 1 and Appendix B.1. Workflow details are available in the GitHub repository README file.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Training details and evaluation metrics can be found in Section 4. Additional details required to reproduce results are located in Appendix B.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Standard deviation for experimental results are presented in each table. No figures were presented that required error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Hardware and compute details are presented in Section 4. Expected times for reproducing results can be ascertained from Section 4.3.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This work conforms to the guidelines outlined in the code of ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

https://doi.org/10.52202/085713-0570

Justification: While the broader field of protein design has clear societal implications, this work presents a methodological contribution that is tangential to applications with direct positive or negative societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This work does not pose any risks that require safeguards.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We use the publicly available ProGen2 model and existing protein sequence datasets. We cite the original works for each of these assets and follow usage terms as provided by these works.

Guidelines:

- The answer NA means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Code for new assets is released publicly and includes extensive documentation both within this work as well as the repository where the code is stored.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This work does not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This work does not involve crowdsourcing or human subjects.

https://doi.org/10.52202/085713-0570

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: Method development did not involve LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.