https://doi.org/10.1038/s41587-024-02123-4

Designing proteins with language models

Jeffrey A. Ruffolo & Ali Madani

Protein language models learn from diverse sequences spanning the evolutionary tree and have proven to be powerful tools for sequence design, variant effect prediction and structure prediction. What are the foundations of protein language models, and how are they applied in protein engineering?

Proteins are composed of a linear chain of residues, with 20 canonical amino acids making up the vocabulary of most natural proteins. The ordering of these amino acids determines the tertiary structure of proteins in their environment and subsequently enables their specific function. Understanding the relationships between protein sequence, structure, and function is a major focus of biological research. In this Primer, we focus on a class of machine learning models that operate only on sequences yet capture the structural and functional properties of proteins. Protein language models (PLMs) are trained on vast datasets of protein sequences spanning the evolutionary tree of life. From these sequences, PLMs learn the underpinnings of protein structure and function, enabling a wide range of protein modeling and design tasks.

Evolution of protein sequences

Advances in high-throughput DNA sequencing technology enabled the collection of billions of protein sequences from a wide variety of sources. The growth in the number of observed protein sequences (billions) has outpaced the rate of structural data collection (hundreds of thousands). As we observe greater numbers of protein sequences, we can begin to identify the patterns underlying the evolutionary process. From a structural perspective, certain mutations may cause larger disruptions to secondary or tertiary structure by breaking α-helices, introducing unsatisfied hydrogen bonds, or burying charged atoms. In general, none of these mutations are entirely restricted, but the bias toward maintenance of the existing fold is sufficiently strong to imprint upon the evolutionary process. From a functional perspective, particular amino acids (or residues) must be carefully coordinated to carry out the biological role of a protein. For example, in the case of zinc finger motifs, several residues must be present and properly oriented to bind the metal ion (Fig. 1a). As with structural constraints, violation of these functional arrangements can occur and give rise to new functionalities. However, such events are exceedingly improbable, meaning we will typically observe few changes at key functional positions, and if such changes occur, they are often compensated by changes at other positions that together define the functional arrangement.

These soft constraints on protein sequences are often referred to as coevolution. Exploitation of this coevolutionary information has enabled advances in protein modeling, most notably for protein structure prediction. With language models, we aim to explicitly model the interdependencies between residues in proteins.

Foundations of protein language models

Fundamentally, protein language models aim to predict how likely we are to observe a particular protein sequence S given all the protein sequence data collected thus far. We denote a protein sequence $S = (s_1, s_2, \ldots s_N)$, where s_i represents the amino acid at position i in the sequence. As a first approximation, we might consider the probability of observing a protein as the joint probability of observing each of its constituent amino acids. Under this model, referred to as unigram, we calculate the probability of a sequence S as

$$P(S) = \prod_i^N P(s_i)$$

In practice, to compute $P(S)$, we simply tabulate the frequency of each amino acid occurring in our sequence database and multiply the probabilities for the specific sequence S. However, proteins are not unordered collections of amino acids. Rather, the specific order in which we observe the amino acids is a critical determinant of structure and function. To capture this order dependency, we can use the preceding residues to inform the probability of the next amino acid. In an n-gram model, we multiply these contextualized probabilities to form the overall probability of the sequence:

$$P(S) = \prod_i^N P(s_i | s_{i-(n-1)}, \ldots, s_{i-1})$$

In the above equation, the initial residues of the sequence may have fewer than n preceding residues (or none, for s_1). When $n = 2$, this model is called a bigram, and we can tabulate the frequency of each amino acid occurring after the preceding amino acid in our sequence dataset to calculate $P(S)$.

Bigram models may begin to capture the patterns of secondary structure, which display varying amino acid propensities, but are insufficient to model dependencies separated by long stretches of sequence, as exemplified by the zinc finger domain in Fig. 1a. To capture long-range dependencies, we could simply increase the number of preceding residues considered by the model. However, in practice, as the model is extended to consider more context, the number of sequences necessary to inform the statistical measurements grows exponentially. For the zinc finger domain in Fig. 1a, the active site spans 21 consecutive residues, meaning the specific arrangement we observed is one of 20^{21} possibilities. To address this challenge, modern language models typically use a neural network architecture called Transformer, capable of learning sequence dependencies over arbitrary-length contexts from data rather than tabulation.

Transformers incorporate the entire sequence context

The Transformer model was first proposed for machine translation of natural languages – for example, translation of English text to German. The original Transformer model included an encoder to summarize the source text and a decoder to produce text in the target language. However, for many applications in natural language processing and

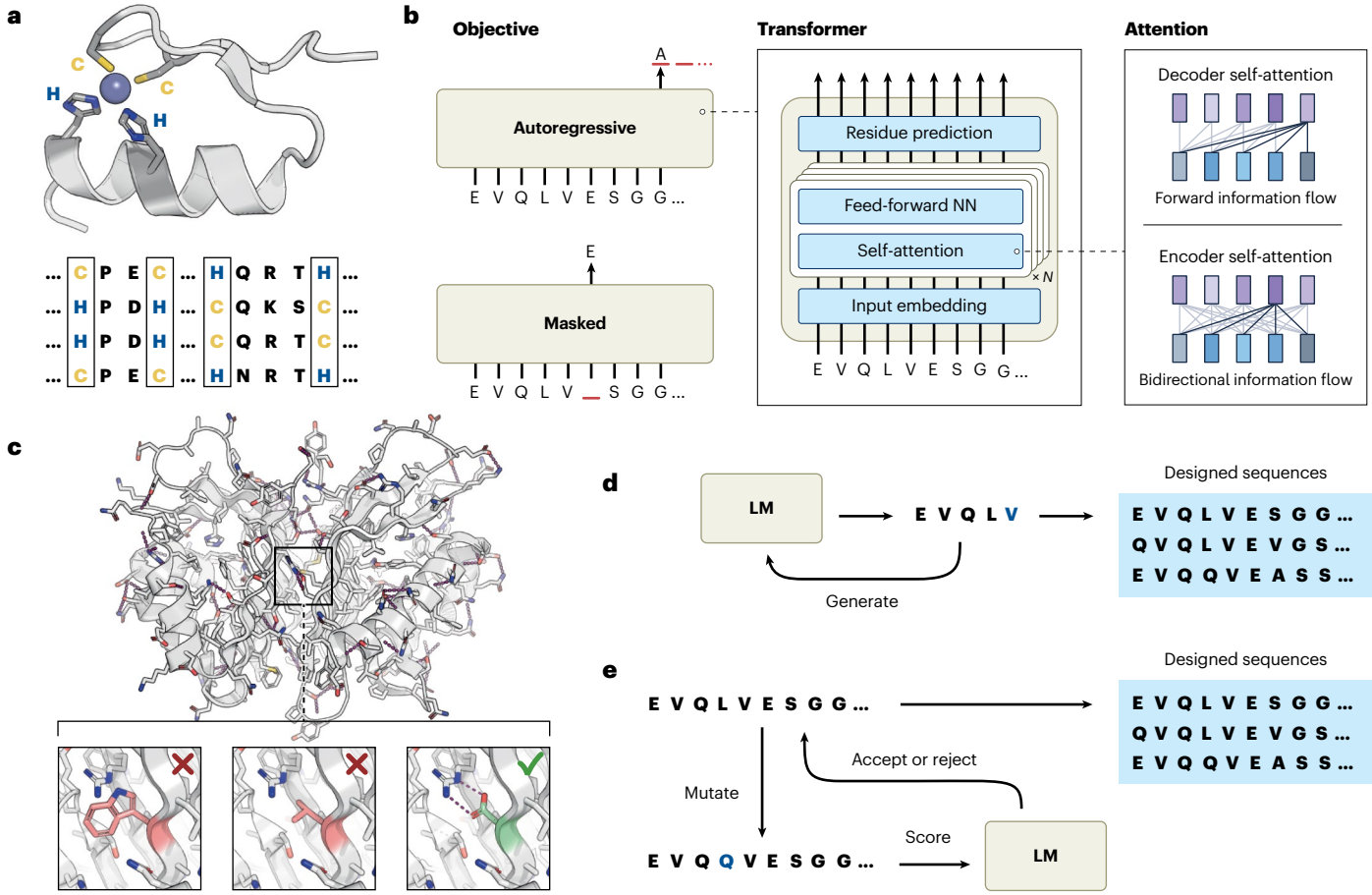

Fig. 1 | Application of language models to protein design. a, The co-evolving zinc finger functional residues are localized in spatial proximity but discontinuous in sequence. **b**, Autoregressive and masked language models are typically based on the Transformer architecture but use different types of attention for their respective objectives. NN, neural network. **c**, An NTF2 protein dimer is shown with side chain polar interactions indicated by red lines. In the magnified panels, three potential amino acids are shown for one position. The most favorable amino acid for this position is an asparagine forming two hydrogen bonds with a nearby residue. With atomically precise tertiary understanding, models can recapitulate such interactions. **d**, Schematic for designing protein sequences through autoregressive generation with a language model (LM). **e**, Schematic for optimizing sequences through Markov chain Monte Carlo sampling with an LM.

protein sequence modeling, these components are employed individually as encoder-only and decoder-only language models. The network architecture is largely the same for both types of models (Fig. 1b). First, the input sequence of amino acids is projected to a 'hidden' or latent sequence by an input embedding layer. Next, a series of repeating attention layers (see below) and feed-forward networks process the sequence representation. Finally, a residue prediction layer projects the processed sequence representation back to a predicted distribution over amino acids. Ultimately, the model is trained to fill in missing amino acids either at the end of a sequence (decoder-only) or in the middle of the sequence (encoder-only).

Among the key innovations of the Transformer is the use of attention to model global dependencies across sequences. Intuitively, the attention mechanism enables models to learn which parts of sequence context are relevant for a given prediction, much as a human might pay attention to specific portions of an essay more than other portions when asked a reading comprehension question. When a sequence representation is passed to an attention layer, each position emits a set of query and key vectors. If the query from one position i matches the key from another position j (typically measured by a dot product), the network assigns high attention from i to j. The attention values for all pairs of positions are collected into an attention matrix with dimension $N \times N$. Each position also emits a value vector. To update the sequence representation, we calculate a weighted sum for each position i according to the attention from i to all other positions j and their respective value vectors. This process can be summarized using matrix multiplication as

$$\text{Attention}(Q, K, V) = \text{softmax}(QK^{\mathrm{T}})V$$

where Q, K and V are matrices in which the ith rows are the query, key and value vectors for each sequence position, respectively, and softmax is a normalizing operation to ensure that the attention from each position sums to 1; K^{T} denotes the transpose of K. In practice, each attention layer typically performs several independent attention operations in parallel (referred to as multi-head attention), allowing the model to learn multiple distinct connectivity patterns.

In practice, encoder-only and decoder-only models trained on protein sequences have proven quite useful. Encoder-only models are

generally employed for learning representations of sequences, which are then adapted to various downstream tasks, while decoder-only models are used for generation and scoring of protein sequences.

Autoregressive language models generate and score proteins

Decoder models are sometimes referred to as autoregressive language models because they are trained in a manner that allows them to generate sequences by iteratively predicting the next residue based on their previous output (see also the Primer by Hsu et al.). They are trained using a next-token prediction objective where the probability of the next amino acid is informed by the entire preceding sequence:

$$P(S) = \prod_i^N P(s_i|s_{<i})$$

We train the autoregressive model on a database of sequences to predict $P(s_i|s_{<i})$. To facilitate this task, causal masking is used to restrict the attention operations throughout the model such that information flows only from earlier positions to later, and not the reverse (Fig. 1b, decoder self-attention). Prominent examples of autoregressive models for protein sequences include UniRep, ProGen and ProtGPT2. Autoregressive models can generate diverse sequences adopting a wide variety of folds, and the predicted $P(S)$ has also been shown to correlate with the functional fitness of proteins. Sequences are generated by iteratively sampling the next residue from the predicted distribution $P(s_i \mid s_{<i})$, with each sampled residue being appended to the sequence to inform the following prediction. In a similar fashion, sequences can be scored by computing the likelihood of a sequence $P(S)$ according to the model, which can be considered the likelihood of a given sequence being produced by the evolutionary forces that gave rise to the training data. These models have been trained on a wide variety of datasets, including genomic, metagenomic and immune-repertoire sequences. By modifying the training data composition, we can vary the types of sequences generated by the model, as well as learn better fitness predictors. This alignment between the training data and the intended application of the model is a critical consideration that has large impacts on performance.

Masked language models learn generalizable representations

For encoder models, the training objective is modified to predict the identities of residues throughout the sequence. Specifically, a subset of residues is randomly selected and replaced with a special mask token, and the model (termed a masked language model) is tasked with predicting their identities. This objective can be expressed as $P(S_{-M}) = \prod_{i \in M} P(s_i|S_{-M})$, where M is a set of masked positions and $-M$ is the remaining unmasked positions. Unlike autoregressive models, masked language models use bidirectional attention and consider all residues throughout the sequence to make predictions (Fig. 1b, encoder self-attention). Prominent examples of masked language models for protein sequences include the ESM and ProtTrans families of models.

To perform well on the masked language modeling objective, models must learn a broad set protein features. For example, to predict the identity of a masked residue, models are implicitly encouraged (that is, without supervision) to build up secondary and tertiary structure representations (Fig. 1c). The attention matrices from masked language models have also been shown to directly encode protein structure, in the form of residue–residue contact maps. Beyond structural features, masked protein language models capture biophysical properties, evolutionary context and alignment within families. Since they learn generalizable representations, masked language models are often used to encode a given protein for a multitude of downstream sequence prediction tasks such as prediction of functional activity or interactions.

Generating and optimizing functional proteins

Madani et al. (2023) used a language model to generate functional protein enzymes. An autoregressive language model with over 1 billion parameters was trained on over 280 million protein sequences from more than 19,000 families. The training was augmented with tags derived from a given protein's associated metadata to enable efficient learning and primarily provide a method for controllable – that is, conditional – sequence generation from desired input arguments (for example, to generate a library of artificial sequences that likely reside within a predefined protein family). A library of over a million artificial sequences was generated by iteratively sampling the next amino acid with the previously sampled residue context fed as input to the model (Fig. 1d). A variety of decoding strategies for language models have been developed to improve diversity and quality of sequences, including beam search, top-k sampling and nucleus sampling. Each of these techniques reshapes the probability distribution at each step of autoregressive decoding, balancing computing cost against the diversity and quality of generated sequences.

In contrast to de novo sequence generation, most protein engineering efforts aim to optimize the functionality of a protein if access to a high-fidelity assay is available. In this scenario, the starting point, or parent sequence, is known and iteratively optimized through directed evolution. Language models can be trained in a supervised setting with sequence-label pairs derived from experimental data. Biswas et al. (2021) used as little as 24 functionally assayed mutant sequences to train a fitness predictor with a supervised language model. A Markov chain Monte Carlo procedure was used to optimize the sequences of green fluorescent protein and β-lactamase (Fig. 1e). In Markov chain Monte Carlo modeling, random mutations are generated, the likelihood of the resulting protein is scored by the PLM, and the proposed mutation is accepted or rejected with a probability based on the likelihood. These in silico designed sequences have been shown to have improved functionality in the wet lab.

Protein language models have proven effective at generating functional proteins and facilitating optimization of a given protein. Looking forward, controllable generation of functionally specified protein sequences remains an area of great promise. Current techniques require fine-tuning on a curated set of natural proteins, which can be challenging to assemble for poorly represented families or impossible for novel functions. Removing this constraint may enable on-demand generation of functional proteins.

Jeffrey A. Ruffolo & Ali Madani
Profluent Bio, Berkeley, CA, USA.
e-mail: ali@profluent.bio

Published online: 15 February 2024

Competing interests

J.A.R. and A.M. are employed by Profluent Bio, Inc.

Additional information

Peer review information *Nature Biotechnology* thanks the anonymous reviewers for their contribution to the peer review of this work.