PRISM: ENHANCING PROTEIN SEQUENCE DESIGN THROUGH FINE-GRAINED RETRIEVAL ON STRUCTURE-SEQUENCE MULTIMODAL REPRESENTATIONS

Sazan Mahbub[*1], **Souvik Kundu**[†2], **Eric P. Xing**[†1,3,4]

[1]Carnegie Mellon University, USA
[6]Intel Labs, USA
[2]Mohamed bin Zayed University of Artificial Intelligence, UAE
[4]GenBio AI, USA

ABSTRACT

Designing protein sequences that adopt a specific three-dimensional (3D) structure remains a significant challenge in computational biology. Existing methods usually rely on *static* learning approaches that, during inference, cannot dynamically integrate rich multimodal representations from larger datasets, specifically the combined information of 3D structure and 1D sequence. In this paper, we introduce PRISM, a novel retrieval-augmented generation (RAG) framework that enhances protein sequence design by dynamically incorporating fine-grained multimodal representations from a larger set of known structure-sequence pairs. Our experiments demonstrate that PRISM significantly outperforms state-of-the-art techniques in sequence recovery, emphasizing the advantages of incorporating fine-grained, multimodal retrieval-augmented generation in protein design.

1 INTRODUCTION

Proteins are fundamental to biological functions, with their three-dimensional (3D) structure governing their activity. This structure is determined by the amino acid sequence, making the inverse folding problem—designing a sequence that adopts a target structure—a central challenge in protein design. However, due to the intricate and non-deterministic nature of structure-to-sequence mapping, traditional computational approaches are often impractical.

Deep learning has emerged as the predominant approach for structure-conditioned sequence design, leveraging graph neural networks (GNNs) for 3D structural encoding (Jing et al., 2020; Dauparas et al., 2022; Ingraham et al., 2019) and generative approaches, including autoregressive (Dauparas et al., 2022), conditional masked language modeling (Zheng et al., 2023), and discrete diffusion (Wang et al., 2024a; Sun et al., 2024), for sequence generation. However, existing methods primarily learn in a *static* manner, with limited efforts to dynamically incorporate fine-grained, multimodal protein representations of training samples—specifically, the joint information of 3D structure and 1D sequence—during inference. Recent work by Wang et al. (2024b) explored retrieval-augmented generation (RAG) to leverage existing protein databases during inference for antibody design. However, their approach relies heavily on traditional full-structure similarity search, which is computationally expensive and lacks fine granularity. Moreover, their retrieval mechanism is predominantly structure-based, overlooking the integrated role of sequence information.

In this study, we introduce PRISM, a novel hierarchical RAG framework for protein inverse folding that dynamically retrieves and integrates fine-grained multimodal protein representations in a computationally efficient manner. PRISM fully leverages both 3D structural and sequence information, capturing rich local and global dependencies. Experimental results demonstrate that PRISM substantially outperforms the other state-of-the-art protein inverse folding methods in sequence recovery.

[*]smahbub@cs.cmu.edu
[†]Corresponding authors: souvikk.kundu@intel.com, epxing@cs.cmu.edu

2 METHOD

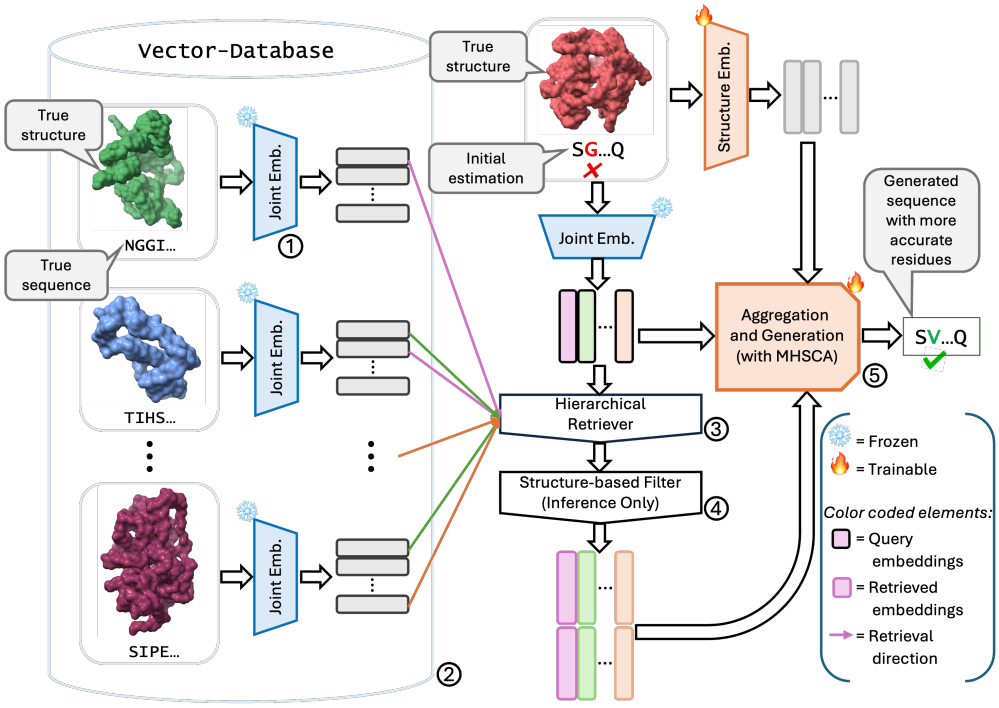

Figure 1: The overall pipeline of our proposed framework PRISM.

In this section we discuss our proposed RAG framework, namely PRISM. The overview of our framework is demonstrated in Figure 1.

2.1 STRUCTURE-SEQUENCE MULTIMODAL REPRESENTATION

We start with a function $\mathcal{G}$ capable of jointly encoding *two modalities* of data–the 3D structure $\mathcal{X}$ and 1D sequence S of any protein P (Figure 1, Point ①). Specifically,

$$\mathcal{E} = \mathcal{G}(P) = \mathcal{G}(\mathcal{X}, S) \in \mathbb{R}^{N \times d}, \tag{1}$$

where $\mathcal{E}$ is the joint embedding, N is the number of residues in the protein P, and d is the embedding dimension. Here $\mathcal{X}$ is the 3D structure of the protein, which can be represented as a nearest neighbor graph, $\mathcal{X}(V, E)$, where V and E respectively represent the nodes and the edges of this graph ($|V| = N$). Also $S = [S_1, S_2, \ldots, S_N]$ is the sequence of amino-acid residues S_j in P. The joint embedding matrix, $\mathcal{E} \in \mathbb{R}^{N \times d}$, consists of N vectors, each representing an amino acid in the protein P. Each vector, $\mathcal{E}_j \in \mathbb{R}^d$, captures the *context* of the 3D neighborhood surrounding the j-th residue as well as its long-range dependencies with other parts of the protein. This approach differs from many retrieval-augmented generation (RAG) methods, which segment sequences and represent each segment with a single vector, potentially losing granularity essential for accurate protein representation.

The sequence S can be effectively encoded using pre-trained protein language models (Sun et al., 2024; Nadav et al., 2023; Wang et al., 2024a), while the graph $\mathcal{X}(V, E)$ can be encoded with standard protein 3D structure encoders (Dauparas et al., 2022; Gao et al., 2022; Jing et al., 2020). Some methods jointly encode the structure and sequence of proteins, leveraging both pre-trained language models and structure encoders (Zheng et al., 2023; Wang et al., 2024a; Sun et al., 2024). In this study, we utilize AIDO.ProteinIF (Sun et al., 2024) as the multimodal encoding function $\mathcal{G}(.)$, however, we note that our proposed method is agnostic to any specific embedding model.

2.2 VECTOR-DATABASE

Here we will discuss how we create our vector-database (Vector-DB). We assume that we have access to the structures and sequences of a set of M proteins $\mathbb{P} = \{P^i : i \in [1, M]\} = \{(\mathcal{X}^i(V^i, E^i), S^i) :$

$i \in [1, M]\}$, ideally used as a training set. We embed each protein $P^i \in \mathbb{P}$ with the function $\mathcal{G}$ (Equation 1) and get a set of embedding matrices $\mathbb{E} = \{\mathcal{E}^i : i \in [1, M]\}$. Now we create our vector database as a set of *mappings*, $\mathbb{V} = \{\langle \mathcal{E}^i_j \Rightarrow f^i_j \rangle : i \in [1, M]$ *and* $j \in [1, |P^i|]\}$, where each element $\langle \mathcal{E}^i_j \Rightarrow f^i_j \rangle$ is a mapping from an embedding vector $\mathcal{E}^i_j$ to a 3D structure fragment f^i_j, consisting of r closest residues of the j-th residue in protein P^i (r is a hyperparameter). Later we use these fragments for structure-based filtering (discussed later in Section 2.4). Our vector-database is demonstrated in Figure 1, Point ②.

2.3 Hierarchical Retriever

For inverse folding, during inference we only know the structure $\mathcal{X}^q$ of a query protein P^q, which we use to generate a sequence. The query proteins should also be encoded by the same joint-embedding function $\mathcal{G}$ form Equation 1. However, since we do not know any sequence for our query protein *a priori*, we can estimate an initial version of the sequence with any off-the-shelf inverse folding method (Sun et al., 2024; Zheng et al., 2023; Wang et al., 2024a), which we can further improve with our RAG-based framework. We choose AIDO.ProteinIF (Sun et al., 2024) as our initial sequence estimator (our approach is agnostic to any other such method). The initial estimate $\hat{S}^q$ is then used to compute a *crude* representation of our query protein as $\hat{\mathcal{E}}^q = \mathcal{G}(\mathcal{X}^q, \hat{S}^q)$. For each vector $\hat{\mathcal{E}}^q_l$, we retrieve a set of the top K most similar entities $\mathbb{V}^{[q,l]} \in \mathbb{V}$ based on cosine-similarity, where K is a hyperparameter and $|\mathbb{V}^{[q,l]}| = K$. We note that $\hat{\mathcal{E}}^q$ is likely to be a less accurate representation of the protein P^q. However, from our analysis we find that our retriever can usually use this embedding to retrieve structurally similar fragments from other proteins in $\mathbb{V}$, as shown in an example in Appendix A, Figure 3. We also note that the complexity of search in a naive retrieval process would be $O\big(\sum_{i=1}^{M} |P_i|\big)$, which can be quite expensive for large M. Especially the search can become prohibitive if we want to use a very large training set (e.g., millions of proteins from AlphaFoldDB (Varadi et al., 2022; John et al., 2021)) or when computational resource is limited. To address this we take a hierarchal retrieval approach (Malkov & Yashunin, 2018). This allows us to retrieve entities form $\mathbb{V}$ in $O\big(\log(\sum_{i=1}^{M} |P_i|)\big)$ time, which is much faster than the naive approach. Our retriever module is shown as Point ③ in Figure 1. Future work will explore the potential of different types of retrieval techniques.

2.4 Structure-based Filtering

As discussed above, we retrieve relevant entities from our vector database based on a crude representation $\hat{\mathcal{E}}^q$ of our query protein P^q. However, we note that the perfomance of such retrieval will partly depend on the accuracy of the initial estimation $\hat{S}_p$. In order to relax such dependency, we further filter out a small subset of the retrieved entities in $\mathbb{V}^{[q,l]}$ using direct 3D structure similarity search, using a widely used tool named MASTER (Zhou & Grigoryan, 2015; 2020) (Figure 1, Point ④). Specifically, we first extract the structure fragment f^q_l that consists of r nearest neighbors of the l-th residue in our query protein P^q. Then, for each element (a mapping) $\langle \mathcal{E}^i_j \Rightarrow f^i_j \rangle \in \mathbb{V}^{[q,l]}$, we compute the structural similarity score between f^i_j and the query fragment f^q_l (root mean square distance of locally aligned structures). Based on these scores, we take the top $\tilde{K}$ entities from $\mathbb{V}^{[q,l]}$ and end up with a subset $\tilde{\mathbb{V}}^{[q,l]} \in \mathbb{V}^{[q,l]} \in \mathbb{V}$, where $\tilde{K} = |\tilde{\mathbb{V}}^{[q,l]}| < K$. This further refines the set of the entities we retrieve to be highly relevant to the query protein. Unlike Wang et al. (2024b), we apply traditional structure search only on the retrieved subset of fragments, which makes it computationally much efficient. Moreover, we apply structure-based filtering only during inference, which further reduces the total computational cost.

2.5 Aggregation and Generation

We aggregate the filtered entities $\tilde{\mathbb{V}}^{[q,l]}$ $\forall l \in [1, |\hat{S}^q|]$ to generate a new sequence $\tilde{S}^q$. We do this with a series of L consecutive learnable blocks, each consisting of one multihead self-attention layer (MHSA), one multihead cross-attention layer (MHCA), and two bottleneck multilayer perceptrons (Houlsby et al., 2019) (L is a hyperparameter). In the rest of this article, we refer to this hybrid block as multihead self-cross attention block (MHSCA).

As shown in Figure 1 (Point ⑤) and Figure 4 (in Appendix A), for $\forall l \in [1, |\hat{S}^q|]$ we first extract the embedding vectors $\{(\mathcal{E}^i_j)_k : k \in [1, \tilde{K}]\}$ from our retrieved entities in $\tilde{\mathbb{V}}^{[q,l]}$, where $(\mathcal{E}^i_j)_k$ is the *key*

of the mapping $\tilde{\mathbb{V}}_k^{[q,l]}$. We then merge them with the query vector $\mathcal{E}_l^q$ and linearly project the output to create a matrix $\mathcal{H}_l^q \in \mathbb{R}^{(\tilde{K}+1)\times d'}$ as,

$$\mathcal{H}_l^q = \text{concat}(\{(\mathcal{E}_j^i)_k : k \in [1, \tilde{K}]\} \cup \{\mathcal{E}_l^q\})\, W_{\mathcal{H}}, \tag{2}$$

where concat(.) performs a concatenation operation on the vectors in the union set, $W_{\mathcal{H}} \in \mathbb{R}^{d\times d'}$ is a learnable parameter, and d' is the output embedding dimension. Then for the whole query protein P^q we get a tensor $\mathcal{H}^q = [\mathcal{H}_1^q, \mathcal{H}_2^q, \ldots, \mathcal{H}_{|\hat{S}^q|}^q] \in \mathbb{R}^{|\hat{S}^q|\times(\tilde{K}+1)\times d'}$, which is used as *query*, *key*, and *value* of MHSA (see Vaswani (2017) for definitions). To ensure that the generator can effectively leverage any residual 3D structural information, we also encode the input structure $\mathcal{X}^q$ separately using a structural encoder, where *no sequence* information is provided. Similar to AIDO.ProteinIF (Sun et al., 2024), we leverage ProteinMPNN-CMLM (Zheng et al., 2023) for structure encoding, which is a variant of the original ProteinMPNN method (Dauparas et al., 2022) trained with conditional masked language modeling objective (Ghazvininejad et al., 2019). This generates structural encoding $\rho^q \in \mathbb{R}^{|\hat{S}^q|\times d^\rho}$. This encoding is then linearly transformed and merged with a linear projection of query encoding $\mathcal{E}^q$, creating a new representation matrix $\theta^q \in \mathbb{R}^{|\hat{S}^q|\times d'}$, where each element $\theta_l^q = \text{concat}(\{\rho_l^q W_\rho, \mathcal{E}_l^q W_{\mathcal{E}}\}) \in \mathbb{R}^{d'}$, with two learnable parameters $W_\rho \in \mathbb{R}^{d^\rho \times \frac{d'}{2}}$ and $W_{\mathcal{E}} \in \mathbb{R}^{d\times\frac{d'}{2}}$. For our MHCA blocks, we use θ^q as the *query*, and $\mathcal{H}^q$ as both the *key* and *value*. The motivation behind such design of MHSCA is, while MHSA layers can help jointly attend to multiple parts of the input protein as well as their corresponding retrieved embeddings, MHCA can help extract any kind of residual structural information needed to better decode the sequence. Moreover, since the MHCA here preserves the same dimension as θ^q, the output representation has $|\hat{S}^q|$ vectors which we can directly pass through another linear layer to generate the output logits $\mathcal{Y} \in \mathbb{R}^{|\hat{S}^q|\times d'}$. Sampling with $\mathcal{Y}$ provides us with a newly generated sequence $\tilde{S}^q$.

3 Results and Discussion

For our experiments, we leverage the widely used CATH-4.2 dataset (Orengo et al., 1997). This is a standard benchmark datatset where all the sequences are within 500 residues length. In Appendix A, in Figure 2 and Table 2 we show the distribution of sequence lengths and statistics on this dataset, respectively.

Table 1: **Evaluation of various protein inverse folding techniques**. Sequence recovery rates (SRR) of earlier methods are referenced from Wang et al. (2024a) and Sun et al. (2024). The best and the second best scores are shown in bold and italic fonts. Here "pLM" is the acronym for "protein language model".

Method Name	Uses pLM?	Uses RAG?	SRR
StructTrans (Ingraham et al., 2019)	×	×	35.82 %
GVP (Jing et al., 2020)	×	×	39.47 %
ProteinMPNN (Dauparas et al., 2022)	×	×	45.96 %
PiFold (Gao et al., 2022)	×	×	51.66 %
ProteinMPNN-CMLM (Zheng et al., 2023)	×	×	48.62 %
LM-Design (Zheng et al., 2023)	✓	×	54.41 %
DPLM (Wang et al., 2024a)	✓	×	54.54 %
AIDO.ProteinIF (Sun et al., 2024)	✓	×	*58.26* %
PRISM (Ours)	✓	✓	**60.29** %

In Table 1, we present a quantitative comparison among different state-of-the-art (SoTA) methods for protein inverse folding. We show that, with the help of our proposed RAG framework, we can significantly improve the sequence recovery rate on CATH-4.2 benchmark dataset. Specifically, PRISM achieves a recovery rate more than 2% higher compared to the next best performing method AIDO.ProteinIF (Sun et al., 2024).

4 Conclusion

In this study, we present PRISM, a multimodal retrieval-augmented generation framework aimed at improving protein inverse folding by dynamically incorporating finegrained structure-sequence

multimodal representations from a larger protein database, resulting in enhanced sequence recovery over current methods. Future work will involve testing on additional datasets and examining the generated sequences for notable insights.

REFERENCES

Justas Dauparas, Ivan Anishchenko, Nathaniel Bennett, Hua Bai, Robert J Ragotte, Lukas F Milles, Basile IM Wicky, Alexis Courbet, Rob J de Haas, Neville Bethel, et al. Robust deep learning–based protein sequence design using proteinmpnn. *Science*, 378(6615):49–56, 2022.

Zhangyang Gao, Cheng Tan, Pablo Chacón, and Stan Z Li. Pifold: Toward effective and efficient protein inverse folding. *arXiv preprint arXiv:2209.12643*, 2022.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. *arXiv preprint arXiv:1904.09324*, 2019.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

John Ingraham, Vikas Garg, Regina Barzilay, and Tommi Jaakkola. Generative models for graph-based protein design. *Advances in neural information processing systems*, 32, 2019.

Bowen Jing, Stephan Eismann, Patricia Suriana, Raphael John Lamarre Townshend, and Ron Dror. Learning from protein structure with geometric vector perceptrons. In *International Conference on Learning Representations*, 2020.

Jumper John, Evans Richard, Pritzel Alexander, Green Tim, Figurnov Michael, Ronneberger Olaf, Tunyasuvunakool Kathryn, Bates Russ, Žídek Augustin, Potapenko Anna, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 2021.

Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

Brandes Nadav, Goldman Grant, Wang Charlotte, H., Ye Chun, Jimmie, and Ntranos Vasilis. Genome-wide prediction of disease variant effects with a deep protein language model. *Nature Genetics*, 2023.

Christine A Orengo, Alex D Michie, Susan Jones, David T Jones, Mark B Swindells, and Janet M Thornton. Cath–a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109, 1997.

Ning Sun, Shuxian Zou, Tianhua Tao, Sazan Mahbub, Dian Li, Yonghao Zhuang, Hongyi Wang, Xingyi Cheng, Le Song, and Eric P Xing. Mixture of experts enable efficient and effective protein understanding and design. *bioRxiv*, pp. 2024–11, 2024.

Mihaly Varadi, Stephen Anyango, Mandar Deshpande, Sreenath Nair, Cindy Natassia, Galabina Yordanova, David Yuan, Oana Stroe, Gemma Wood, Agata Laydon, et al. Alphafold protein structure database: massively expanding the structural coverage of protein-sequence space with high-accuracy models. *Nucleic acids research*, 50(D1):D439–D444, 2022.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. Diffusion language models are versatile protein learners. *arXiv preprint arXiv:2402.18567*, 2024a.

Zichen Wang, Yaokun Ji, Jianing Tian, and Shuangjia Zheng. Retrieval augmented diffusion model for structure-informed antibody design and optimization. *arXiv preprint arXiv:2410.15040*, 2024b.

Zaixiang Zheng, Yifan Deng, Dongyu Xue, Yi Zhou, Fei Ye, and Quanquan Gu. Structure-informed language models are protein designers. In *International conference on machine learning*, pp. 42317–42338. PMLR, 2023.

Jianfu Zhou and Gevorg Grigoryan. Rapid search for tertiary fragments reveals protein sequence–structure relationships. *Protein Science*, 24(4):508–524, 2015.

Jianfu Zhou and Gevorg Grigoryan. A c++ library for protein sub-structure search. *bioRxiv*, pp. 2020–04, 2020.

A APPENDIX

Table 2: Statistics of CATH-4.2 dataset (Orengo et al., 1997). Here "St. Dev." stands for "standard deviation".

Data split	# of sequences	# of residues	Mean Length	Median Length	St. Dev. of Lengths
Train	18,024	3,941,775	218.7	204.0	109.93
Validation	608	105,926	174.22	146.0	92.44
Test	1,120	181,693	162.23	138.0	82.22
Combined	19,752	4,229,394	214.12	196.0	109.06

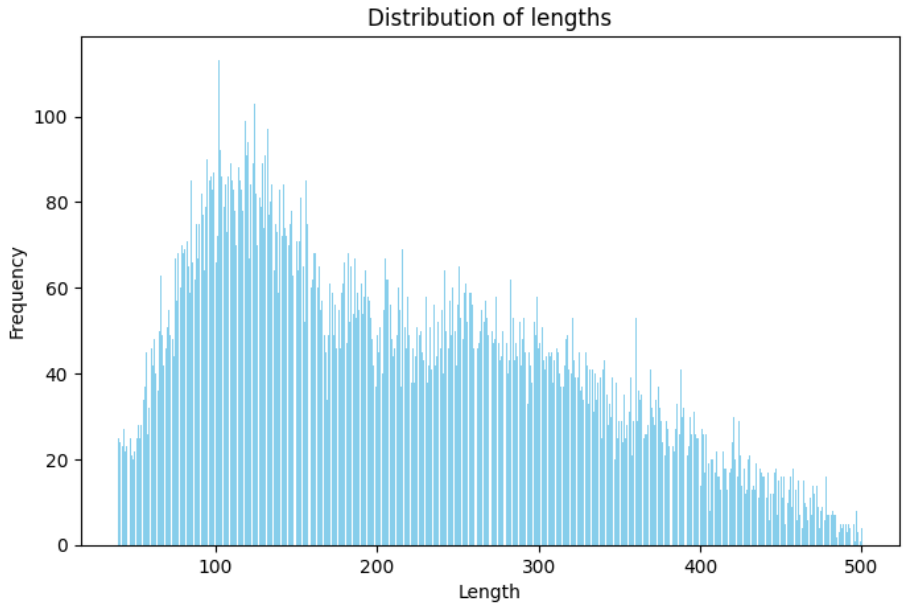

Figure 2: Distribution of lengths of the protein sequences in the benchmark dataset CATH-4.2 (Orengo et al., 1997).

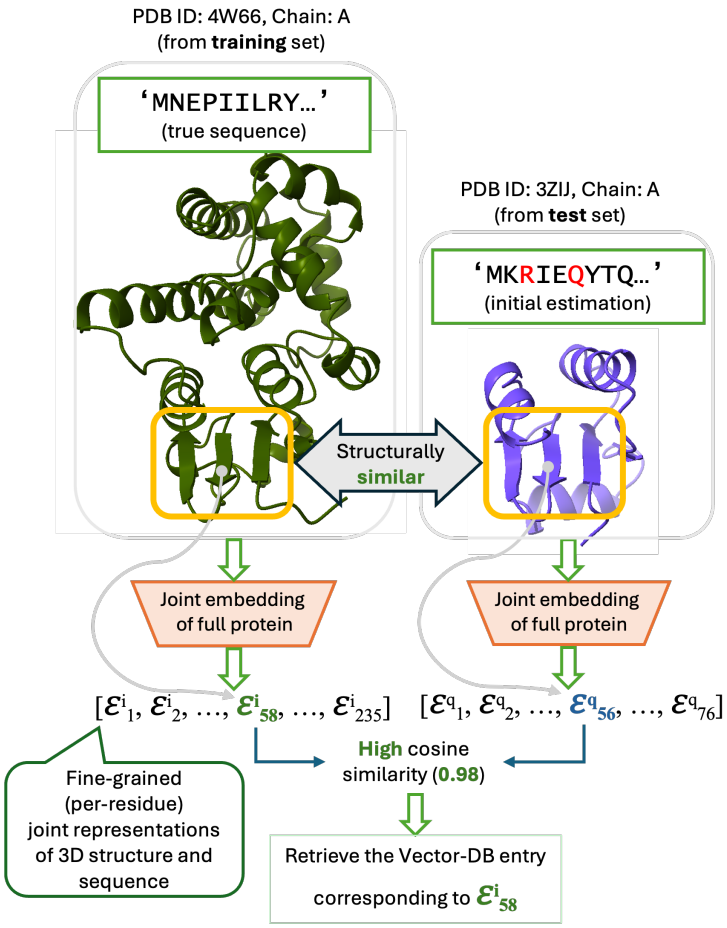

Figure 3: A example of how our retrieval process is actually aligning with true structure similarity search. This shows retrieval in such a way is indeed potential to extract rich information from already-existing larger protein databases. Here we leverage embedding $\mathcal{E}^i_j$ (fine-grained, at the very residue-level), which maps to a structure fragment f^i_j, consisting of the r-nearest neighbors (residues) of the j-th residue.

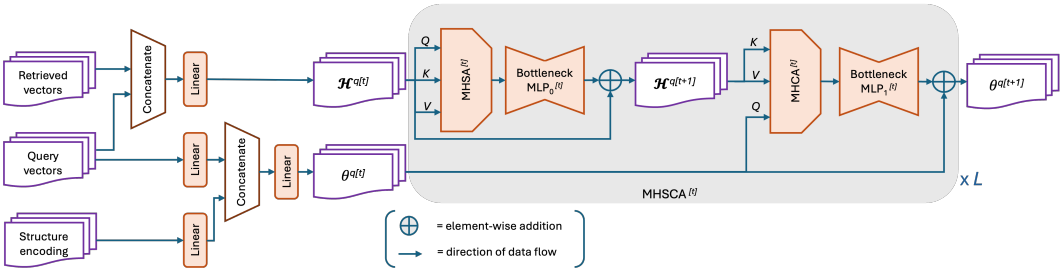

Figure 4: Our aggregation and generation module that uses L consecutive blocks of MHSCA. Here the super-script "$[t]$" corresponds to the *index* of the current MHSCA block (and also its components and their inputs).