PAIR: Boosting the Predictive Power of Protein Representations with a Corpus of Text Annotations

Anonymous Authors[1]

Abstract

Protein language models trained on raw amino acid sequences have demonstrated impressive success in various protein function prediction tasks. One explanation for this success is that language modeling for amino acid sequences captures the local evolutionary fitness landscape and, therefore, encourages the models to extract rich information about the structure and function of a protein. Yet, detecting distant evolutionary relationships from sequences alone is a challenge. In this work, we conduct a comprehensive study examining the effects of training protein models on nineteen types of expertly-curated function annotations in Swiss-Prot. We find that different annotation types had varying effects on the quality of the learned representations, with some even degrading the model's performance. However, by incorporating a carefully-selected subset of annotation types, we are able to improve the model's function prediction performance. Notably, unlike existing protein models, our approach either matches or outperforms the widely-used bioinformatics tool BLAST in annotating previously uncharacterized proteins.

1. Introduction

Advances in sequencing technologies have enabled the discovery of over 250 million protein sequences. Despite this wealth of sequence information, only a very small fraction - less than 0.25% of the UniProt Knowledgebase (Consortium, 2020) - have comprehensive and expertly-curated annotations describing their functions. Performing wet-lab experiments to obtain functional annotations on such a massive scale is simply not feasible due to prohibitive amounts of time, money, and resources it would consume. As a result, it has become important to develop computational techniques that can reliably transfer functional information from the relatively tiny annotated subset to the astronomically larger group of proteins whose functions remain uncharacterized.

The exponentially growing number of unannotated proteins has inspired the development of protein language models (PLMs) (Rives et al., 2021a; Lin et al., 2023a; Elnaggar et al., 2021b). Similar to the language modelling approach for human languages, PLMs are trained to learn a joint probability distribution over sequences of amino acids. This approach necessitates the models to learn the mutation patterns and evolutionary statistics of proteins. The underlying premise is that by doing so, the models can link an amino acid sequence to its three-dimensional structure and functional characteristics based on the fundamental biological rules of "sequence→structure→function". PLMs have demonstrated remarkable success across a wide array of protein-related tasks, including remote homology detection (Rives et al., 2021a), prediction of secondary structures (Lin et al., 2023a), enzyme commission number (Yu et al., 2023) and affinity maturation (Hie et al., 2024).

Despite this success, it remains unclear if training on sequence information only is the optimal strategy to learn generalizable biochemical principles that govern structure and function (Vu et al., 2023). The relationship between a protein's amino acid sequence and its biochemical function can be complex and multi-layered. Crucial factors like three-dimensional folding and molecular interactions can substantially influence a protein's functionality in ways that may not be immediately evident from the sequence itself. What's more, the pretraining objective of PLMs focuses on amino acids themselves, lacking broader contextual information about the protein as a whole. This "local-level" pretraining approach may not optimally align with tasks aimed at predicting protein function at a global level. Therefore, leveraging additional types of data other than the amino acid sequence could potentially enhance PLMs' ability to accurately predict protein functions.

Motivated by previous works that utilized function annotations to improve protein representations (Xu et al., 2023; Liu et al., 2023; Zhang et al., 2022; You et al., 2018), we conduct a comprehensive study examining the effects of

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review at ICML 2024 AI for Science workshop. Do not distribute.

training on nineteen types of expertly-curated functional annotations in Swiss-Prot. These annotations directly contain the information about the fundamental functional properties of proteins, such as their domain structures, family classifications and binding sites. Our results show that different types of annotations have highly variable impacts on the quality of the learned representations. Strikingly, we discovered that some annotation types actually hindered the model's performance. This implies that simply integrating all available annotation data would potentially lead to suboptimal performance, which underscores the importance of our study.

We then present a flexible fine-tuning framework, PAIR, to improve the quality of protein representations with our carefully-selected types of function annotations. PAIR employs a text decoder to direct the fine-tuning process of the encoder, enabling the learned protein representations to better align with the annotated functional information. Compared with the previous works, our model is more straightforward and adaptable, capable of handling any type of text annotations.

Our results have demonstrated that PAIR substantially improves the representational quality of PLMs for function predictions. Notably, we employed a temporal data split on UniProtKB-Swiss-Prot to evaluate the representations' predictive capabilities for uncharacterized proteins. We find that PAIR outperforms the base PLM by over 10% across a diverse range of nine function prediction tasks. Moreover, PAIR exhibits strong generalization on the tasks absent from its training data, which indicates PAIR learned generalizable representations across different aspects of protein function.

In a comparison against BLAST (Altschul et al., 1990), the *de-facto* sequence searching algorithm in bioinformatics, PAIR either matches or surpasses BLAST across all tasks, while all other PLMs we tested do worse. We also find that PAIR demonstrates more substantial advantages over BLAST when predicting the proteins with low sequence similarity to the training set. Moreover, PAIR is more computationally efficient than BLAST as we can pre-compute and store the representations of the training set. We envision that PAIR could be used as a useful computational tool to annotate uncharacterized proteins' functions and also inspire more effective pretraining strategies for protein language models.

2. Related Work

2.1. Computational techniques for protein labelling

Detecting protein sequence homology using sequence similarity is the standard approach to identifying functions that are common between proteins. A common way to do so is determining pairwise alignment across large sets of labeled sequences to retrieve similar protein sequences to a query using the Basic Local Alignment Search Tool (BLAST) algorithm (Altschul et al., 1990). BLAST is a carefully-designed heuristics-based algorithm that approximates the optimal alignment between sequences. Other methods include Hidden Markov Models (HMMs)(Söding, 2005) and integrate other information like protein family, amino acid composition and evolutionary information (Altschul et al., 1997; Chou, 2005; Steinegger et al., 2019; Wang et al., 2017). These methods are the state-of-the-art of classical representation methods. While they are reliable for proteins that have high sequence similarity ($> 25\%$), they struggle at remote homology detection – accurate classification of sequences that have low similarity to the current databases. This has motivated the emergence of deep learning efforts (Kulmanov et al., 2018; Hou et al., 2018; Almagro Armenteros et al., 2017; Li et al., 2018; Ryu et al., 2019).

2.2. Protein language models

Recently, numerous studies on self-supervised learning on large-scale protein sequences were reported in the literature (Rives et al., 2021b; Elnaggar et al., 2021a; Alley et al., 2019; Rao et al., 2019). Such works usually use existing language model architectures and training objectives from natural language processing (NLP), and are therefore frequently called protein language models (PLMs) (Unsal et al., 2022; Vu et al., 2023). The hope is that biochemical and physicochemical principles that underlie sequence–structure–function relationships can be captured and be used to predict the functional properties of proteins (Yu et al., 2023; Radivojac et al., 2013; Rao et al., 2019; Unsal et al., 2022; Xu et al., 2022).

2.3. Protein representation learning with text annotations

In an effort to enrich the pretrained protein representations, several works have augmented the training pipeline with modalities such as natural language descriptions. A pioneering work is DeepText2GO, which uses abstracts from the National Library of Medicine and homology information together to predict GO annotations of protein sequences (You et al., 2018). Another example is the OntoProtein model, which jointly optimizes the embedding of the GO knowledge graph, composed of text descriptions, and the protein sequence embedding using contrastive learning during pre-training (Zhang et al., 2022). Both works mentioned are trained using a single source of annotations.

The work most closely related to ours is ProtST (Xu et al., 2023). ProtST employs a combination of four training objectives: masked protein modeling, InfoNCE contrastive loss, protein-to-text masked language modeling, and text-to-protein masked language modeling. ProtST is trained

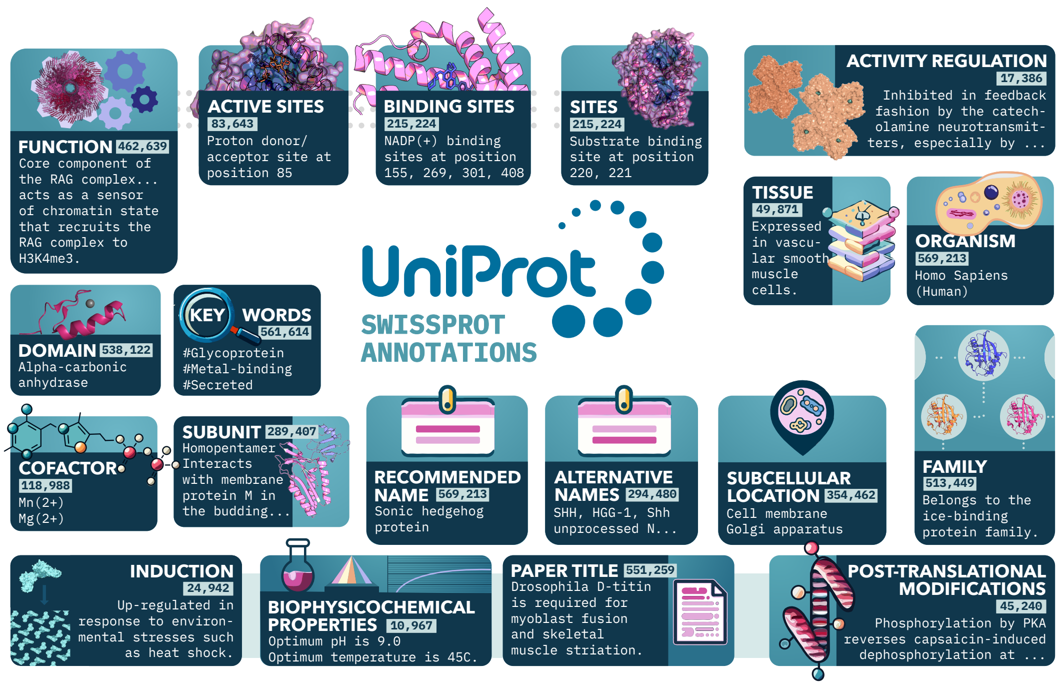

FUNCTION 462,639
Core component of the RAG complex... acts as a sensor of chromatin state that recruits the RAG complex to H3K4me3.
ACTIVE SITES 83,643
Proton donor/acceptor site at position 85
BINDING SITES 215,224
NADP(+) binding sites at position 155, 269, 301, 408
SITES 215,224
Substrate binding site at position 220, 221
ACTIVITY REGULATION 17,386
Inhibited in feedback fashion by the catecholamine neurotransmitters, especially by ...
TISSUE 49,871
Expressed in vascular smooth muscle cells.
ORGANISM 569,213
Homo Sapiens (Human)
UniProt
SWISSPROT ANNOTATIONS
DOMAIN 538,122
Alpha-carbonic anhydrase
KEY WORDS 561,614
#Glycoprotein
#Metal-binding
#Secreted
COFACTOR 118,988
Mn(2+)
Mg(2+)
SUBUNIT 289,407
Homopentamer Interacts with membrane protein M in the budding...
RECOMMENDED NAME 569,213
Sonic hedgehog protein
ALTERNATIVE NAMES 294,480
SHH, HGG-1, Shh unprocessed N...
SUBCELLULAR LOCATION 354,462
Cell membrane Golgi apparatus
FAMILY 513,449
Belongs to the ice-binding protein family.
INDUCTION 24,942
Up-regulated in response to environmental stresses such as heat shock.
BIOPHYSICOCHEMICAL PROPERTIES 10,967
Optimum pH is 9.0
Optimum temperature is 45C.
PAPER TITLE 551,259
Drosophila D-titin is required for myoblast fusion and skeletal muscle striation.
POST-TRANSLATIONAL MODIFICATIONS 45,240
Phosphorylation by PKA reverses capsaicin-induced dephosphorylation at ...

Figure 1. **Overview of annotation types in our training data.** We parsed nineteen types of annotations from Swiss-Prot February 2023. Each box contains the name of the annotation type, the number of proteins having that type, and an example underneath.

on four types of annotations: function, name, subcellular location and family. In comparison to ProtST, our approach offers three distinct advantages: 1) We leveraged the entire Swiss-Prot database and trained on a more extensive set of annotation types. 2) Our training loss formulation is significantly simpler, while obtaining superior results. 3) We conducted a more rigorous evaluation using a temporal data split and held-out proteins.

3. Data

In order to collect high-quality annotations to train PAIR, we created a database of canonical amino acid sequences and function annotations from Swiss-Prot. We downloaded the version of Swiss-Prot published in February 2023, which contains annotations for 569,213 proteins with varying text annotations, such as name, family and organism. In total, we then parsed 19 types of annotations from Swiss-Prot as shown in Figure 1. We deliberately held out the annotation types enzyme commission number (EC) and Gene Ontology (GO) from our training set to evaluate the model's performance on tasks it has not been trained on. The details of how we parsed each category can be found in Section E.

Dataset split. We then generated a train/validation data split based on sequence similarity, as random splitting could give falsely high results if the same sequence (from e.g. different organisms) exists in both training and validation, making it harder to detect overfitting. For this, we obtained the 2023-02 UniRef50 cluster centers with a member in Swiss-Prot 2023-02. Then, we ran MMseqs2[1] on the cluster centers using a similar search as in Lin et al. (2023b) with a sequence similarity threshold of 10%: `--min-seq-id 0.1 --alignment-mode 3 --max-seqs 300 -s 7 -c 0.8 --cov-mode 0`. We randomly selected 10% of the resulting UniRef50 reference sequences and generated the validation set by combining the members of those UniRef50 clusters that were present in Swiss-Prot 2023-02. The remaining 90% of reference sequences were used to create the training set in the same way. For the test set, we used a temporal split by taking the proteins from the Swiss-Prot January 2024 (2024-01) snapshot that had been added since Swiss-Prot 2023-02. This was to mimic the real-world setting of annotating newly discovered proteins.

4. Model

The core component of PAIR is a transformer-based encoder-decoder model (Rothe et al., 2020), where the encoder takes the amino acid sequence as the input and the decoder outputs the associated text annotation. The encoder can be initialized with pretrained protein encoders such as ESM (Rives et al., 2021a; Lin et al., 2023a) and ProtT5 (Elnaggar et al., 2021b), and the decoder can be initialized by language mod-

[1] https://github.com/soedinglab/MMseqs2

els trained on biomedical text (for example, SciBert (Beltagy et al., 2019a)). To model the relationship between the protein sequence and the function annotation, we then added cross-attention that attends to each amino acid from the text token. During training, the parameters of the encoder, decoder, and cross-attention are updated jointly so the model can learn to generate the function annotations of the input amino acid sequences.

Let $x = (x_1, \cdots, x_n)$ denote one input protein sequence and $y = (y_1, \cdots, y_m)$ denote the target text annotation. Each x_i is an amino acid and each y_i is a text token. The encoder first maps $(x_1, \cdots, x_n)$ to a sequence of continuous representations $V_N \in \mathbf{R}^{n \times d}$ through N self-attention blocks, where n is the sequence length and d is the representation dimension. V_N is then passed to the decoder to generate the text annotation y in an auto-regressive fashion.

Each attention block of the decoder consists of a self-attention layer followed by a cross-attention layer. In cross-attention, the queries of the attention come from the previous self-attention layer, and the keys and values come from the output of the encoder. Note that in the case where the encoder and the decoder output representation of different lengths, we add a linear projection head on the encoder side to map its representation to the same dimension as the decoder. After going through a stack of attention blocks, the decoder outputs a continuous representation r_k for the position k. We then project r_k with a linear head to compute the logits over the vocabulary of the decoder. At the end, we minimize the cross-entropy loss between this logit and one-hot encoding of the true token y_k. To obtain the final protein representation, we averaged all the amino acid representations from the last layer of the protein encoder. Our implementation of the model follows `transformers.EncoderDecoderModel` from Huggingface (Wolf et al., 2019). The details of our model architectures can be found in Section A.

Pretrained Models. We studied pretrained protein models from two model families: ProtT5 (Elnaggar et al., 2021b) and ESM (Rives et al., 2021a) (Lin et al., 2023a). ProtT5 adopts the architecture of Transformer-XL (Dai et al., 2019) and is trained on UniRef100 (Suzek et al., 2015). We only used the encoder component of ProtT5 with 1.5 billion parameters. We picked two model variants of ESM: ESM1b (Rives et al., 2021a) and ESM2-650M (Lin et al., 2023a). They use the same architectures as BERT (Devlin et al., 2018) and are trained with the masked language modelling loss on UniParc (Consortium, 2020). We initialized the decoder with SciBERT (Beltagy et al., 2019b), which was trained on 1.14 million biomedical and engineering papers from Semantic Scholar.

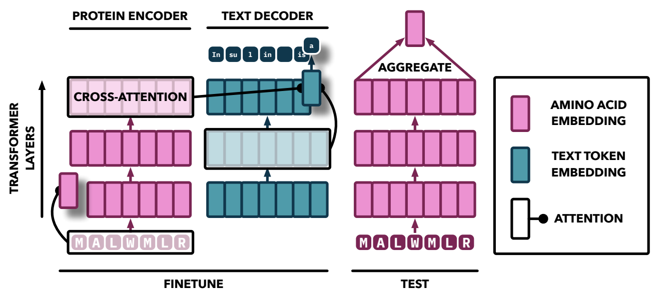


Figure 2. **PAIR uses an attention-based sequence-to-sequence architecture.** The encoder and the decoder are initialized by the pretrained models. A randomly-initialized cross-attention layer is then added between the encoder and the decoder. During training, the encoder, decoder and the cross-attention are updated jointly to generate the annotations of the input amino acid sequences. To obtain the final protein representation, we averaged all amino acid representations from the last layer of the protein encoder.

5. Evaluation

To comprehensively assess the representation quality, we curated nine important function annotation tasks from Swiss-Prot. For each task, we performed k-nearest neighbors (k-NN) on the embedding space. This process is analogous to the sequence searching method commonly employed in bioinformatics. Specifically, for a query protein in the validation set, we assigned it the labels of the closest k protein(s) in the training set based on Euclidean distance. k-NN is a simple discriminator that does not require additional training, which makes it a convenient and fair method for representation evaluations. For all tasks, we used $k = 1$.

Below, we present a brief description of each task, how we parsed the label(s), and what we used as evaluation metrics.

Gene Ontology annotations (GO). GO is a major bioinformatics initiative to unify the representation of genes and gene product attributes (including proteins) across all species. It provides a controlled, hierarchical vocabulary to describe three aspects of protein annotations: molecular function (GO-MF), biological process (GO-BP), and cellular component (GO-CC). We parsed GO labels from the `<feature type="GO">` tag in the downloaded Swiss-Prot XML. Similar to Kulmanov & Hoehndorf (2022), we only kept the annotations with tags denoting that the labels were validated from real-world experiments. Our evaluation follows Kulmanov & Hoehndorf (2022).

Enzyme Commission number (EC). The EC numbering system is a classification mechanism for the enzymes based on the chemical reactions they catalyze. It consists of 4 digits, each of which represents a progressively more refined categorization. We obtained the EC numbers from the tags of `<feature type="EC">`. Then we removed any numbers with fewer than 4 digits. We also removed numbers if they contained the character “n”, which indicates that the annotation has not yet been officially assigned. We used F_1 as the evaluation metric for this task.

Recommended protein name (Name). Gane et al. (2023) showed that if the protein name can be predicted, it can give information about its function. Inspired by this, we extracted the recommended name of the protein from the `<protein>/<recommendedName>/<fullName>` tag. For evaluation, we computed the exact match accuracy.

UniProt protein family (Family). A protein family is a set of proteins with common evolutionary origin. UniProt assigns a family name based on information from external protein family databases, sequence similarity and analysis tools, and literature. This is documented in the ”Sequence similarities” section on UniProt, and typically appears as a sentence *”Belongs to the XX family”*. We parsed the family name from the `<comment type="similarity">/<text>` tags and computed the exact match accuracy during evaluation.

Pfam domains (Domain). A protein domain is a distinct structural and functional unit within a protein structure. We extracted the Pfam domain annotations from `<dbReference type="Pfam">/<property type="entry name">` tags. We used F_1 as the evaluatiion metrics.

Active sites. Active sites are related to the enzymatic behaviour and describe which protein components are involved in catalytic reactions. On UniProt, this information exists as a table where one column is a residue number and another describes the residue role (e.g. “proton acceptor”, “eletrophile”). We extracted the residue roles from the `<feature type="active site">` tags in the XML file. We used F_1 as the evaluation metric.

Binding sites. This task studies the interaction between proteins and small ligands, such as metals and cofactors. We extracted the names of the ligands within `<feature type="binding site"><ligand>` tags and used F_1 as the evaluation metric.

6. Results

6.1. PAIR’s representation quality depends on the annotation type

The category of the function annotations plays a vital role in determining the effectiveness of the learned representations. Prior research has not rigorously examined which specific types of annotations are most beneficial for learning high-quality protein representations. To bridge this gap, our initial study systematically evaluates the impact of different annotation data types on the learned representation quality. For a fair comparison, we fine-tuned ESM2-150M on each annotation type separately for one epoch with the same hyperparameters on the training set, and evaluated the performance of the nine function prediction tasks (introduced in Section 5) on the validation set. These results are shown in Table 1.

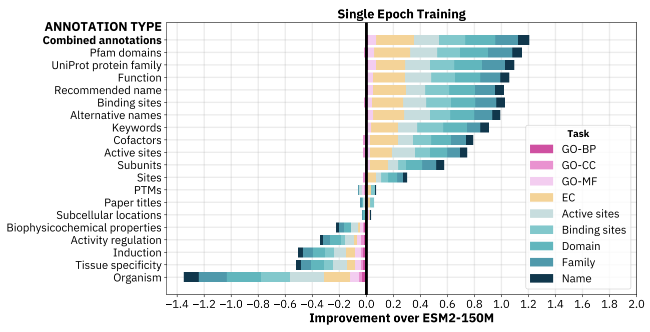


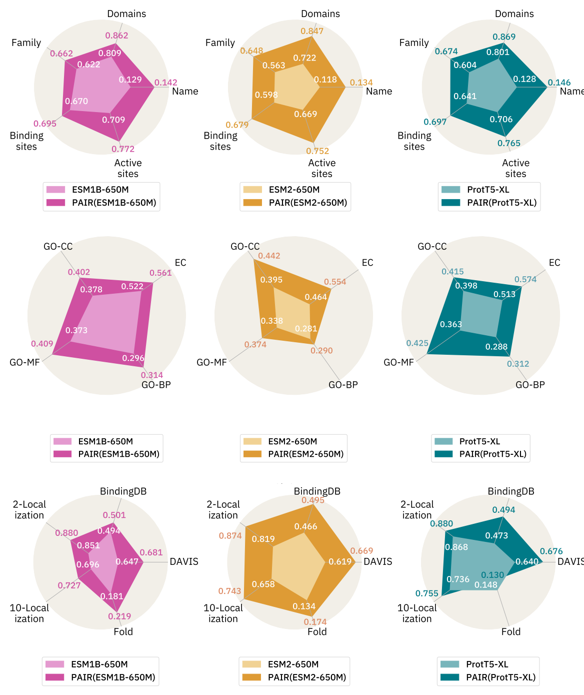
Figure 3. **Annotation types play an important role in the quality of learned representations.** Each row corresponds with the performance change on the 10% validation set after fine-tuning ESM2-150M on the corresponding category. Bars on the right indicate improved performance over the base ESM2-150M for that specific task, while bars extending to the left indicate a performance degradation. We find that different annotation types had varying effects on the quality of the learned representations, with some annotation types even degrading the model’s performance.

Figure 3 summarizes the change of performance after training on each annotation category, compared with the original ESM2-150M. Out of the 19 annotation types we collected, training on 14 of them led to improved performance. Among them, *Pfam domain*, *UniProt protein family* and *Recommended name* resulted in the largest performance gains. Moreover, it is remarkable that these performance gains are observed in tasks that the model was not specifically fine-tuned on. This shows PAIR’s ability to learn useful protein representations that can generalize beyond the domains it was trained on.

On the other hand, the performance on the evaluation decreased after being fine-tuned on *Organism*, *Tissue specificity*, *Biophysicochemical properties*, *Induction* and *Activity regulation* annotations. We speculate that this performance degradation can be attributed to two reasons: a) organism and tissue specificity represent higher levels of biological organization. Training on them provides little to no information about a protein’s molecular functions. b) as shown in Figure 1, less than 5% of the proteins have the annotations *Biophysicochemical properties*, *Induction* and *Activity regulation*, which means they do not provide sufficient information for the model to learn generalizable representations. Additionally, the *Biophysicochemical properties* of a protein may be too vague for predicting its function (i.e. if we the know the optimal pH of a protein is 9.0, that would not be informative about what the protein does).

Based on the findings above, we removed these 5 annotation categories from the training set. We then investigated whether it is better to train on the remaining 14 types by combining them all together compared to training on them individually. We fine-tuned ESM2-150M on those annotations for one epoch and show the performance in the first row of Figure 3. The result indicates that training on all of them yields superior performance compared to training on any single annotation, demonstrating that there is complementary information across different annotation categories.

6.2. PAIR improves function annotations for uncharacterized proteins



(c)

Figure 4. **PAIR improves function predictions.** PAIR consistently improves upon base models when predicting (a) five function annotation categories in the training set, (b) four function annotation categories *not* in the training set, and (c) five tasks in the external benchmarks. Each spider plot shows the performance of PAIR compared to the base model it was finetuned on. We plot exact performance values for the baseline models in white, PAIR(ESM1b) in pink, PAIR(ESM2-650M) in orange, and PAIR(ProtT5) in blue. To better visualize the performance differences across tasks of different scales, we use Z-score normalization to scale the plots, as in Karamcheti et al. (2024).

Training details. We then fine-tuned three larger models (ESM2-650M , ESM1b and ProtT5) with PAIR on the combined 14 annotations. We denote each fine-tuned model as PAIR(ESM2-650M), PAIR(ESM1b) and PAIR(ProtT5). To save memory, we trained our model in the half-precision format of bfloat16 (Kalamkar et al., 2019). Our training is performed on an HPC cluster of AMD M100 GPU nodes. Each node consist of eight 32GB M100 GPUs. We trained 2 ESM models on 4 nodes, and ProtT5 on 8 nodes. For protein sequences longer than 1024 amino acids, we randomly sampled window of 1024 amino acids. For the text annotation, we truncated the token length to 128. We used the AdamW optimizer ((Loshchilov & Hutter, 2018)) with a learning rate of $1e-4$ and a weight decay of $1e-4$. We clipped the gradient norm to 1. Our total batch size across all GPUs was 160 for ESM and 128 for ProtT5 . We trained ESM2-650M and ESM1b for $50,000$ steps, and ProtT5 for $30,000$ steps.

We tuned hyperparameters for each model based on the average performance of the validation set on the nine evaluation tasks. Afterwards, we re-trained with the same hyperparameters on the whole February-2023 Swiss-Prot. In order to simulate the real-world scientific discovery process, we generated a temporal test set for model evaluations. Our test set includes all 1631 proteins that were added to Swiss-Prot after the training set was published (those that were introduced between February 2023 and January 2024). The results can be found in Table 2.

Figure 4a plots the performance difference between each base model and the PAIR-enhanced model on predicting the in-distribution function annotations on the temporal test set. The result demonstrates that PAIR consistently improves upon all those tasks for the three models. Averaged over all models, PAIR outperforms the base model by $10.89\%, 12.53\%, 10.55\%, 8.49\%, 9.84\%$ in predicting family, name, domain, binding sites and active sites, respectively. ESM2-650M has the largest improvement with $15.10\%, 13.56\%, 17.31\%, 13.55\%, 12.41\%$ respectively.

Figure 4b evaluates on the tasks not appearing in PAIR's training set, namely EC and GO. In terms of EC predictions, PAIR improves ESM2-650M, ESM1b and ProtT5 by $19.39\%, 7.47\%$ and 11.89% respectively. PAIR also improves all three subontologies of GO consistently across all models, with an average improvement of $7.08\%, 12.12\%, 5.84\%$ for GO-CC, GO-MF, GO-BP.

6.3. PAIR improves remote homology detections and drug-target interactions

We then evaluated PAIR on five external benchmarks: binary localization (Almagro Armenteros et al., 2017), subcellular location (Almagro Armenteros et al., 2017), fold (Hou et al., 2018), DAVIS (Davis et al., 2011) and BindingDB (Liu et al., 2007). The first there are classification tasks for protein functions. A detailed description of these benchmarks can be found in Appendix Section B. We split the training and test set according to Xu et al. (2022), which deliberately lowered the similarity between training and test to evaluate

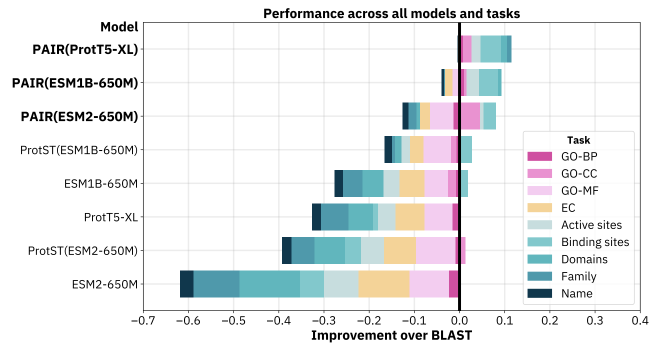


Figure 5. **PAIR is on-par with or superior to BLAST in annotating uncharacterized proteins.** Each row represents the performance compared to BLAST of an individual model on the test set. Bars extending to the right imply an improvement over BLAST for the corresponding task, whereas bars extending to the left indicate inferior performance. PAIR(ProtT5) emerges as the sole model that either matches or surpasses BLAST's performance across all the evaluated tasks.

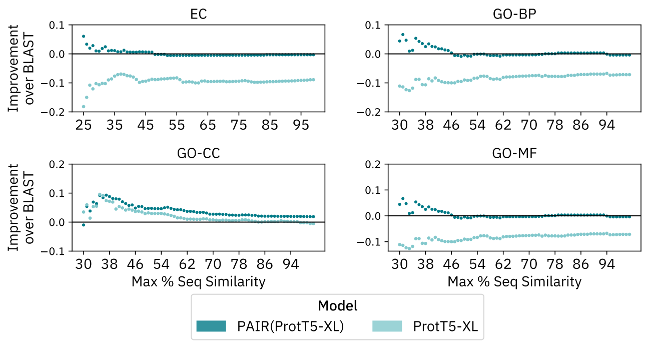


Figure 6. **PAIR demonstrates larger advantages over BLAST when predicting the proteins with low similarity.** The x-axis represents the maximum sequence similarity between a test sample and the training set. The y-axis depicts the performance difference between the model and BLAST, where positive values indicate superior performance compared with BLAST. The lighter blue line corresponds to the original ProtT5 model, while the darker line represents PAIR(ProtT5). The results show that our approach exhibits superior performance when predicting the test samples with low sequence similarity to the training data.

the model's generalization for remote homologous proteins. For the localization tasks, the training and test set share no higher than 30% sequence similarity. For fold, entire superfamilies are held out in the test set from the training set. Figure 4c shows that PAIR-enhanced model outperforms the original model in all but one set-ups.

DAVIS and BindingDB are drug-target interaction (DTI) tasks where the goal is to predict the binding affinity between a molecule and the target protein. We obtained the representations for a molecule-protein pair by concatenating the protein representation with the molecule representation, which we obtained from Text+ChemT5 (Christofidellis et al., 2023). We split the dataset so that there is no overlap of protein sequences across training, validation and test. We repeated this random split 50 times and report the average. Figure 4c shows that PAIR improves the DTI predictions for all models over the two datasets.

6.4. PAIR outperforms BLAST

Furthermore, we compared PAIR with the *de facto* sequence searching method in bioinformatics, BLAST (basic local alignment search tool (Altschul et al., 1990)). First, we find that the base models (ESM and ProtT5) perform significantly worse than BLAST, as shown in Figure 5. Even when compared with the best-performing base model ESM1b , BLAST still outperforms it on eight of the nine tasks. In the case of EC predictions, BLAST surpassed ESM1b by a substantial margin of 10.54%. However, after fine-tuning with our approach, PAIR(ProtT5) emerges as the sole model that either equals or surpasses BLAST's performance across the evaluated tasks. As illustrated in Figure 5, PAIR(ProtT5) outperformed BLAST on six tasks and achieved comparable results on the remaining three. In addition, we compared PAIR with ProtST (Liu et al., 2023). ProtST also provides a framework to fine-tune the protein encoders on text annotations. Figure 6 shows that PAIR consistently outperforms ProtST across tasks given the same pretrained model.

We also examined the performance of PAIR and BLAST across varying degrees of sequence similarity in the test set, where sequence similarity is defined as the percent identity to the top-1 BLAST hit. Our results in Figure 6 indicate that PAIR exhibits superior performance with the low-similarity sample. This finding is particularly promising because low similarity samples pose a greater challenge for sequence similarity alignments, and this fact suggests that PAIR learns complementary information to BLAST.

Furthermore, we evaluate our approach against BLAST on the five external benchmarks. It is not feasible to apply BLAST to the drug-target interaction task since it involves predicting relationships between two entities, whereas BLAST is designed for single-instance prediction tasks. For the other benchmark tasks, our method demonstrates substantially superior performance compared to BLAST (binary localization: 73.76, subcellular location: 53.73, fold: 2.92). We attribute the poor performance of BLAST to the relatively small size of the other datasets and minimal overlap between their training and test sets, which is a scenario where BLAST faces challenges. Our approach seems to generalize better to previously unseen data when training examples are limited.

Moreover, PAIR can be more computationally efficient than BLAST. Running the BLAST algorithm is computationally

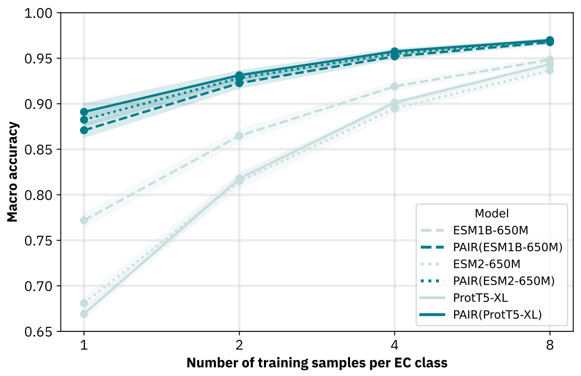


Figure 7. **PAIR improves few-shot EC classifications.** We train a linear classifier on the frozen embeddings with 1, 2, 4, or 8 training samples per EC class. The light blue line represents the base protein models and the dark line represents the PAIR-enhanced models. PAIR-enhanced models outperform the baseline models across all shots. The gap between the PAIR and baseline models becomes larger as the number of training examples decreases.

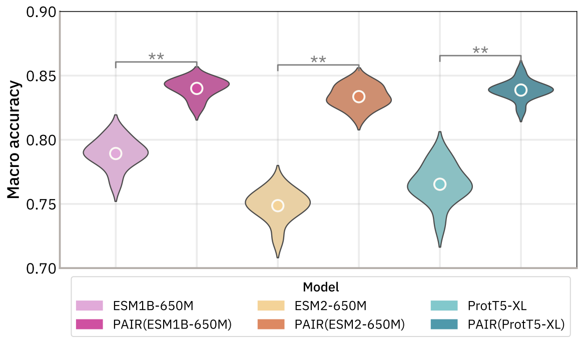


Figure 8. **PAIR improves one-shot EC classifications for low-resource enzymes.** We train a linear classifier on 1 training sample per EC class for low-resource EC numbers (those that have 3-9 samples in the entire dataset). We repeat each experiment for 10 random seeds. We find that the PAIR outperforms each baseline protein language model significantly (** indicates $p < 0.02$ according to a Wilcoxon signed-rank test).

expensive, whereas in our approach, we can pre-compute and store the representations of the training set, after which performing k-nearest neighbor predictions is a highly efficient operation. This allows our method to make predictions much more rapidly compared to having to run the full BLAST algorithm for every new example.

6.5. PAIR improves low-resource enzyme function predictions

We trained a linear classifier on top of PAIR and the base model's embeddings to predict EC numbers in a few-shot setting. We selected enzymes from our Swiss-Prot 2023 dataset that were annotated with a single EC number and belonged to EC classes with at least 10 members. We randomly split them into train (80%), validation (10%) and test (10%) subsets. For each EC class in the training set, we sampled k training samples, where $k \in \{1, 2, 4, 8\}$ and trained a linear classifier to predict the enzyme class. We tuned the learning rate from the values of $\{1e-2, 1e-3, 1e-4, 1e-5\}$ on the validation set and plotted the performance on the test set in Figure 7. We repeated this 10 times for each shot and plotted the standard deviation as error bars. We find that across all k training shots and all base models, there is a clear improvement using PAIR compared with the base models. Even with 1-shot, PAIR embeddings achieve at least 85% accuracy, compared to 67-77% for the baseline models. As the number of training shots increases, performance for all models goes up, but PAIR embeddings consistently maintain superior performance.

In the real-world, many EC classes are unstudied and have very few annotations. We further investigated the performance of one-shot learning with PAIR embeddings for low-resource EC numbers. We considered the subset of Swiss-Prot 2023 containing EC classes with 3-9 members. We then split them into training (1 sample per EC class), validation (1 sample per EC class) and test (remaining samples) sets and fine-tuned a linear classifier to predict the EC number. We show performance on the test set in Figure 8, comparing the performance of PAIR embeddings to embeddings from the corresponding baseline. We find that even in low-resource settings, PAIR embeddings obtain close to 85% accuracy on average, which is significantly higher than baseline models ($p < 0.02$). This demonstrates that functional annotations can significantly enhance the performance of protein embeddings, especially in scenarios where training data is limited.

7. Conclusion

We introduce PAIR, a flexible fine-tuning framework designed to improve the representational quality of protein language models. We then conducted an extensive study examining the impact of training on diverse annotation types derived from the Swiss-Prot database. Our findings demonstrate that fine-tuning PAIR on the collective set of annotation types yields substantial improvements in the protein function prediction capabilities on a temporal split, even in functional tasks that the model was not trained on, as well as on external datasets. This motivates our future directions in that diverse types of protein information and modalities, even if seemingly unrelated to target tasks, should be investigated as important resources in the language modelling paradigm.

References

AI4Science, M. R. and Quantum, M. A. The impact of large language models on scientific discovery: a preliminary study using gpt-4. *arXiv preprint arXiv:2311.07361*, 2023.

Alley, E. C., Khimulya, G., Biswas, S., AlQuraishi, M., and Church, G. M. Unified rational protein engineering with sequence-based deep representation learning. *Nature methods*, 16(12):1315–1322, 2019.

Almagro Armenteros, J. J., Sønderby, C. K., Sønderby, S. K., Nielsen, H., and Winther, O. Deeploc: prediction of protein subcellular localization using deep learning. *Bioinformatics*, 33(21):3387–3395, 2017.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

Beltagy, I., Lo, K., and Cohan, A. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019a.

Beltagy, I., Lo, K., and Cohan, A. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019b.

Bran, A. M., Cox, S., Schilter, O., Baldassari, C., White, A. D., and Schwaller, P. Chemcrow: Augmenting large-language models with chemistry tools. *arXiv preprint arXiv:2304.05376*, 2023.

Chou, K.-C. Using amphiphilic pseudo amino acid composition to predict enzyme subfamily classes. *Bioinformatics*, 21(1):10–19, 2005.

Christofidellis, D., Giannone, G., Born, J., Winther, O., Laino, T., and Manica, M. Unifying molecular and textual representations via multi-task language modelling. 202:6140–6157, 2023. URL https://proceedings.mlr.press/v202/christofidellis23a.html.

Consortium, T. U. UniProt: the universal protein knowledgebase in 2021. *Nucleic Acids Research*, 49(D1): D480–D489, 11 2020. ISSN 0305-1048. doi: 10.1093/nar/gkaa1100. URL https://doi.org/10.1093/nar/gkaa1100.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*, 2019.

Davis, M. I., Hunt, J. P., Herrgard, S., Ciceri, P., Wodicka, L. M., Pallares, G., Hocker, M., Treiber, D. K., and Zarrinkar, P. P. Comprehensive analysis of kinase inhibitor selectivity. *Nature biotechnology*, 29(11):1046–1051, 2011.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., et al. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(10):7112–7127, 2021a.

Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., et al. Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(10):7112–7127, 2021b.

Gane, A., Bileschi, M. L., Dohan, D., Speretta, E., Héliou, A., Meng-Papaxanthos, L., Zellner, H., Brevdo, E., Parikh, A., Martin, M. J., Orchard, S., Collaborators, U., and Colwellm, L. Protnlm: Model-based natural language protein annotation, 2023. URL https://www.uniprot.org/help/ProtNLM.

Hie, B. L., Shanker, V. R., Xu, D., Bruun, T. U., Weidenbacher, P. A., Tang, S., Wu, W., Pak, J. E., and Kim, P. S. Efficient evolution of human antibodies from general protein language models. *Nature Biotechnology*, 42 (2):275–283, 2024.

Hou, J., Adhikari, B., and Cheng, J. Deepsf: deep convolutional neural network for mapping protein sequences to folds. *Bioinformatics*, 34(8):1295–1303, 2018.

Huang, K., Fu, T., Gao, W., Zhao, Y., Roohani, Y., Leskovec, J., Coley, C. W., Xiao, C., Sun, J., and Zitnik, M. Therapeutics data commons: Machine learning datasets and tasks for drug discovery and development. *Proceedings of Neural Information Processing Systems, NeurIPS Datasets and Benchmarks*, 2021.

Kalamkar, D., Mudigere, D., Mellempudi, N., Das, D., Banerjee, K., Avancha, S., Vooturi, D. T., Jammalamadaka, N., Huang, J., Yuen, H., et al. A study of bfloat16 for deep learning training. *arXiv preprint arXiv:1905.12322*, 2019.

Karamcheti, S., Nair, S., Balakrishna, A., Liang, P., Kollar, T., and Sadigh, D. Prismatic vlms: Investigating the design space of visually-conditioned language models. In *International Conference on Machine Learning (ICML)*, 2024.

Kulmanov, M. and Hoehndorf, R. Deepgozero: improving protein function prediction from sequence and zero-shot learning based on ontology axioms. *Bioinformatics*, 38 (Supplement_1):i238–i245, 2022.

Kulmanov, M., Khan, M. A., and Hoehndorf, R. Deepgo: predicting protein functions from sequence and interactions using a deep ontology-aware classifier. *Bioinformatics*, 34(4):660–668, 2018.

Li, Y., Wang, S., Umarov, R., Xie, B., Fan, M., Li, L., and Gao, X. Deepre: sequence-based enzyme ec number prediction by deep learning. *Bioinformatics*, 34(5):760–769, 2018.

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637): 1123–1130, 2023a.

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., Verkuil, R., Kabeli, O., Shmueli, Y., et al. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637): 1123–1130, 2023b.

Liu, S., Zhu, Y., Lu, J., Xu, Z., Nie, W., Gitter, A., Xiao, C., Tang, J., Guo, H., and Anandkumar, A. A text-guided protein design framework. *arXiv preprint arXiv:2302.04611*, 2023.

Liu, T., Lin, Y., Wen, X., Jorissen, R. N., and Gilson, M. K. Bindingdb: a web-accessible database of experimentally determined protein–ligand binding affinities. *Nucleic acids research*, 35(suppl_1):D198–D201, 2007.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018.

Mirza, A., Alampara, N., Kunchapu, S., Emoekabu, B., Krishnan, A., Wilhelmi, M., Okereke, M., Eberhardt, J., Elahi, A. M., Greiner, M., et al. Are large language models superhuman chemists? *arXiv preprint arXiv:2404.01475*, 2024.

Radivojac, P., Clark, W. T., Oron, T. R., Schnoes, A. M., Wittkop, T., Sokolov, A., Graim, K., Funk, C., Verspoor, K., Ben-Hur, A., et al. A large-scale evaluation of computational protein function prediction. *Nature methods*, 10(3):221–227, 2013.

Rao, R., Bhattacharya, N., Thomas, N., Duan, Y., Chen, P., Canny, J., Abbeel, P., and Song, Y. Evaluating protein transfer learning with tape. *Advances in neural information processing systems*, 32, 2019.

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021a.

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15):e2016239118, 2021b.

Rothe, S., Narayan, S., and Severyn, A. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280, 2020.

Ryu, J. Y., Kim, H. U., and Lee, S. Y. Deep learning enables high-quality and high-throughput prediction of enzyme commission numbers. *Proceedings of the National Academy of Sciences*, 116(28):13996–14001, 2019.

Söding, J. Protein homology detection by hmm–hmm comparison. *Bioinformatics*, 21(7):951–960, 2005.

Steinegger, M., Meier, M., Mirdita, M., Vöhringer, H., Haunsberger, S. J., and Söding, J. Hh-suite3 for fast remote homology detection and deep protein annotation. *BMC bioinformatics*, 20(1):1–15, 2019.

Suzek, B. E., Wang, Y., Huang, H., McGarvey, P. B., Wu, C. H., and Consortium, U. Uniref clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

Taylor, R., Kardas, M., Cucurull, G., Scialom, T., Hartshorn, A., Saravia, E., Poulton, A., Kerkez, V., and Stojnic, R. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

Unsal, S., Atas, H., Albayrak, M., Turhan, K., Acar, A. C., and Doğan, T. Learning functional properties of proteins with language models. *Nature Machine Intelligence*, 4 (3):227–245, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Vu, M. H., Akbar, R., Robert, P. A., Swiatczak, B., Sandve, G. K., Greiff, V., and Haug, D. T. T. Linguistically inspired roadmap for building biologically reliable protein language models. *Nature Machine Intelligence*, 5(5): 485–496, 2023.

Wang, J., Yang, B., Revote, J., Leier, A., Marquez-Lago, T. T., Webb, G., Song, J., Chou, K.-C., and Lithgow, T. Possum: a bioinformatics toolkit for generating numerical sequence feature descriptors based on pssm profiles. *Bioinformatics*, 33(17):2756–2758, 2017.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Xu, M., Zhang, Z., Lu, J., Zhu, Z., Zhang, Y., Chang, M., Liu, R., and Tang, J. Peer: a comprehensive and multitask benchmark for protein sequence understanding. *Advances in Neural Information Processing Systems*, 35: 35156–35173, 2022.

Xu, M., Yuan, X., Miret, S., and Tang, J. Protst: Multimodality learning of protein sequences and biomedical texts. *arXiv preprint arXiv:2301.12040*, 2023.

You, R., Huang, X., and Zhu, S. Deeptext2go: improving large-scale protein function prediction with deep semantic text representation. *Methods*, 145:82–90, 2018.

Yu, T., Cui, H., Li, J. C., Luo, Y., Jiang, G., and Zhao, H. Enzyme function prediction using contrastive learning. *Science*, 379(6639):1358–1363, 2023.

Zhang, N., Bi, Z., Liang, X., Cheng, S., Hong, H., Deng, S., Lian, J., Zhang, Q., and Chen, H. Ontoprotein: Protein pretraining with gene ontology embedding. *arXiv preprint arXiv:2201.11147*, 2022.

A. Model

We used an attention-based sequence-to-sequence architecture proposed in (Vaswani et al., 2017). The encoder and the decoder are initialized by the pretrained models. A randomly-initialized cross-attention is then added between the encoder and the decoder, which enables the decoder to gather relevant information from each position of the input sequence. Our implementation follows `transformers.EncoderDecoderModel` from Huggingface (Wolf et al., 2019).

The core building block of the sequence-to-sequence model is the attention mechanism. The attention function maps a query vector Q and set of key-value vector pairs (K, V) to an output vector. The output is computed as a weighted sum of the value vectors V. Each weight is calculated by a similarity function between Q and K. Vaswani et al. (2017) used the scaled dot product as the similarity function:

$$\texttt{Attention(Q, K, V) = softmax(}\frac{QK^T}{\sqrt{d_k}}\texttt{)V}$$

It has proven advantageous to perform multiple attention operations in parallel, called multi-head attention mechanism. Specifically, the queries Q, keys K, and values K are linearly projected h times using distinct learned linear transformations. The outputs of these h parallel attention computations are then combined, typically through concatenation and linear projection, to yield the final attended representation.

$$\texttt{Multihead(Q, K, V) = Concat(head\_1, ..., head\_h)}\ W$$
$$\texttt{where head\_i = Attention(}QW_i^Q, KW_i^K, VW_i^V\texttt{)}$$

Let $x = (x_1, \cdots, x_n)$ denote the input protein sequence and $y = (y_1, \cdots, y_m)$ denote the output text annotation. Each x_i is an amino acid and each y_i is a text token. The encoder first maps $(x_1, \cdots, x_n)$ to a sequence of continuous representations $V_N \in \mathbf{R}^{n \times d}$ through N self-attention blocks, where n is the sequence length and d is the representation dimension. Self-attention is a special attention mechansim where the query, key and value all come from the previous layer of the encoder V_{j-1}. It allows the model to learn the interaction between different amino acids. The self-attention computation at layer j is defined as:

$$\texttt{Self-attention\_j }(V_{j-1})\texttt{ = MultiHead(}V_{j-1}\texttt{, }V_{j-1}\texttt{, }V_{j-1}\texttt{)}$$

After the encoder processes the protein sequence, the representation of the last layer V_N is then passed to the decoder to generate the text annotation y in an auto-regressive fashion. The training objective is to maximize the probability of predicting the next token correctly for each position:

$$\prod_{k=0}^{m} P(y_k|, y_1, \cdots, y_{k-1})$$

Different from the encoder, the attention block of the decoder consists of a self-attention layer followed by a cross-attention layer. The self-attention layer is similar to the encoder except that it operates on the prefix $(y_1, \cdots, y_{k-1})$. It means the decoder cannot attend to the tokens after the current position k. After the self-attention layer, the representations U_k are then passed to the cross-attention to incorporate the information from the encoder. In cross-attention, the queries of the attention come from the previous self-attention layer U_k, and the keys and values come from the output of the encoder V_N:

$$\texttt{Cross-attention\_k }(V_N\texttt{, }U_k\texttt{) = MultiHead(}U_k\texttt{, }V_N\texttt{, }V_N\texttt{)}$$

Note that in the case where the encoder and the decoder output representation of different lengths, we add a linear projection head on the encoder side to map its representation to the same dimension as the decoder.

After going through a stack of attention blocks, the decoder then outputs a continuous representation r_k for the position k. We then projected r_k using a linear head to compute the logits over the vocabulary of the decoder. At the end, we minimized the cross-entropy loss between this logit and one-hot encoding of the true token y_k to maximize $P(y_k|, y_1, \cdots, y_{k-1})$.

B. External benchmarks

Binary localization (Almagro Armenteros et al., 2017). This is a binary classification task that predicts whether a protein is ”membrane-bound” or ”soluble”. We followed the training (5161), test (1746), validation (1727) split by Xu et al. (2022).

The dataset is split to test the model's generalization behaviour across homologous proteins.The protein sequences undergo a clustering process based on a 30% sequence identity threshold, after which they are partitioned into five distinct subsets. Four of these subsets are utilized for training and validation, while the remaining fold is reserved for test. The evaluation metrics is accuracy.

Subcellular Localization (Almagro Armenteros et al., 2017). This is a more fine-grained version of Binary localization. The task asks to classify the locations of a protein in a cell into ten categories: *cell membrane*, *cytoplasm*, *endoplasmic reticulum*, *golgi apparatus*, *lysosome*, *mitochondrion*, *nucleus*, *peroxisome*, *plastid*, *extracellular*. We used the same the split of Xu et al. (2022) (training: 8945, val: 2248, test: 2768). The split was conducted in a similar way as binary localization to test the model's ability to correctly predict the subcellular localization of homologous proteins. The evaluation metrics is accuracy.

Fold (Hou et al., 2018). The task aims to aims to categorize the global structural topology of a protein into one of the 1194 distinct fold designations. We used the original split in Hou et al. (2018) (training: 12312, validation: 736, test: 718). To construct the test set, entire superfamilies are held out from the training and validation. This approach enables an assessment of the model's capability to identify proteins exhibiting similar structural topology despite dissimilar sequences, i.e. remote homology detection. The evaluation metrics is accuracy.

Drug Target Interaction (Liu et al., 2007) (Davis et al., 2011). The effect of a small-molecule drug is primarily determined by its binding affinity with the target protein. The drug-target interaction prediction task seeks to computationally predict the interaction activity score between a protein sequence and a small molecule. We used two datasets for this task: DAVIS (Davis et al., 2011) and BindingDB (Liu et al., 2007). Both datasets are regression tasks that predict the equilibrium dissociation constant (Kd) of a protein-molecule pair. The smaller the KD value, the greater the binding affinity of that pair is. We downloaded both datasets from Therapeutics Data Commons (Huang et al., 2021). We used the pearson correlarion coefficient following Huang et al. (2021). Pearson Correlarion Coefficient measures how linearly correlated between the predictions x and the targets y:

$$r(x, y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{1}$$

We applied cold-start split (Huang et al., 2021) on the protein sequences to split the datasets into training/validation/test with a ratio of $6 : 2 : 2$. The cold split is a split method for multi-instance prediction problems that involve two types of entities such as protein and molecules. We first partitioned data based on only the protein sequences into training, validation, and test sets. Then, all pairs associated with the entities in each set are correspondingly assigned to the respective training, validation, and test splits. This split makes sure that there are no overlapped protein sequences across different partitions, so we can evaluate the model's generalization for unseen proteins.

C. Baselines

BLAST. Basic Local Alignment Search Tool (BLAST) is a widely-used bioinformatics tool that allows users to compare an input or query sequence against a predefined database of sequences to find the most similar one (Altschul et al., 1990). BLAST uses heuristics on local protein regions to approximate the Smith-Waterman algorithm in order to increase the speed of alignment. For each query, BLAST returns similar sequences in the database above a specified significance threshold (E-value), which represents the number of hits with a similar alignment score that could be found by chance in a random database of a similar size. We used BLASTp, which is specialized for comparing protein sequences. We downloaded the BLASTp executable [2]. For each downstream task, we created a BLAST database from sequences in Swiss-Prot 2023-02 that had a label for the given task: `makeblastdb -in swissprot202302_{task}.fasta -out {task}_database -dbtype prot`. For each query sequence in the test set, we ran the BLAST algorithm to obtain a list of sequence hits from the database: `blastp -query swissprot202401_{task}.fasta -db {task}_database -evalue 1000`. Note that we set a high E-value to ensure that each query sequence had a hit. We then took the hit with the lowest E-value (most significant) as the closest training sequence and used it for evaluation.

[2] https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/

ProtST. ProtST is a multi-modal protein language model that was trained to align protein sequences with four types of natural language descriptions from Swiss-Prot (Function, Subcellular Location, Family, Name) (Xu et al., 2023). This model was the first to show that pre-training on both sequences and text descriptions can boost performance on downstream protein tasks. We build on this work by incorporating several more annotation categories and reducing the loss function to a simple next token prediction loss, and show that even with a simpler framework, the incorporating more annotations results in better performance. We obtained ProtST embeddings from their model checkpoints [3].

ProtBert. ProtBert is a BERT-based protein language model trained on 217 million unlabelled amino acid sequences from UniRef100 using a masked language modelling loss (Elnaggar et al., 2021a). We obtained the model from HuggingFace [4] and followed the same sequence pre-processing. We set the maximum length to 1026 (1024 for amino acids and 2 for special tokens).

Galactica. Galactica is a decoder-only Transformer model trained on 48 million scientific papers, textbooks, websites, and other sources of scientific information (Taylor et al., 2022), outperforming several large language models on a range of science and math tasks. We obtained embeddings from Galactica-6.7B [5] by using the same prompt format as in Taylor et al. (2022). We set the maximum length to 1026 (1024 for amino acids and 2 for special tokens) and averaged the embeddings from the last hidden state.

GPT. GPT models have previously demonstrated domain knowledge in biology and chemistry (AI4Science & Quantum, 2023; Bran et al., 2023; Mirza et al., 2024). To create the most similar settings as ours, we followed AI4Science & Quantum (2023) and used API calls to obtain embeddings from GPT-3 (`text-embedding-3-large`, the most capable GPT model providing access to embeddings at the time of writing). We structured the prompt using the FASTA format.

D. Results

	EC	Name	GO-CC	GO-MF	GO-BP	Domain	Family	Binding	Active
ESM2-150M	0.292	0.213	0.387	0.317	0.232	0.515	0.436	0.573	0.630
Recommended Name	0.535	0.279	0.385	0.355	0.243	0.704	0.581	0.754	0.774
Domain	0.560	0.282	0.372	0.366	0.244	0.726	**0.615**	0.738	0.827
Family	0.511	0.281	0.381	0.371	0.248	0.721	0.606	0.754	0.811
Alternative Name	0.518	0.271	0.387	0.357	0.246	0.694	0.568	0.724	0.822
Function	0.527	0.278	0.385	0.360	0.239	0.703	0.584	0.746	0.828
Keywords	0.489	0.273	0.386	0.350	0.237	0.689	0.539	0.764	0.774
Binding sites	0.527	0.274	0.372	0.352	0.238	0.697	0.592	0.756	0.798
Active sites	0.455	0.265	0.366	0.343	0.233	0.648	0.529	0.683	0.800
Sites	0.359	0.245	0.369	0.322	0.227	0.581	0.478	0.624	0.670
Cofactor	0.501	0.266	0.365	0.341	0.233	0.661	0.546	0.709	0.742
Paper title	0.311	0.211	0.394	0.323	0.232	0.502	0.425	0.599	0.608
subunit	0.425	0.271	0.384	0.340	0.236	0.637	0.538	0.629	0.708
PTM	0.326	0.223	0.382	0.303	0.225	0.509	0.435	0.604	0.610
Subcellular location	0.297	0.221	**0.403**	0.323	0.231	0.499	0.431	0.571	0.621
Induction text	0.229	0.182	0.370	0.268	0.214	0.423	0.362	0.508	0.542
Biophysical	0.284	0.198	0.375	0.294	0.216	0.464	0.398	0.573	0.572
Activity regulation	0.267	0.193	0.368	0.286	0.211	0.439	0.382	0.542	0.567
Tissue	0.231	0.181	0.368	0.276	0.208	0.419	0.357	0.507	0.529
Organism	0.102	0.105	0.361	0.252	0.202	0.262	0.228	0.357	0.376
Combined annotations	**0.568**	**0.296**	0.383	**0.375**	**0.249**	**0.737**	0.606	**0.766**	**0.818**

Table 1. Validation-set performance of ESM2-150M fine-tuned with different types of annotations.

[3] https://github.com/DeepGraphLearning/ProtST
[4] https://huggingface.co/Rostlab/prot_bert
[5] https://huggingface.co/facebook/galactica-6.7b

Model	EC	Name	GO-CC	GO-MF	GO-BP	Domain	Family	Binding	Active
Naive	0.003	0.000	0.354	0.185	0.189	0.001	0.000	0.115	0.247
BLAST	**0.577**	**0.147**	0.396	**0.425**	0.304	0.855	0.665	0.652	0.745
ESM2-650M	0.464	0.118	0.395	0.338	0.281	0.722	0.563	0.598	0.669
ESM1b	0.522	0.129	0.378	0.373	0.296	0.809	0.622	0.670	0.709
ProtT5	0.513	0.128	0.398	0.363	0.288	0.801	0.604	0.641	0.706
Prot-BERT	0.392	0.088	0.350	0.251	0.229	0.565	0.427	0.530	0.586
Galactica	0.134	0.047	0.254	0.169	0.162	0.203	0.174	0.235	0.406
GPT	0.068	0.030	0.222	0.126	0.110	0.099	0.099	0.162	0.307
ProtST(ESM2-650M)	0.506	0.126	0.409	0.337	0.295	0.787	0.615	0.616	0.695
ProtST(ESM1b)	0.547	0.131	0.384	0.364	0.297	0.841	0.658	0.679	0.726
PAIR(ESM2-650M)	0.554	0.134	**0.442**	0.374	0.290	0.847	0.648	0.679	0.752
PAIR(ESM1b)	0.561	0.142	0.402	0.409	**0.314**	0.862	0.662	0.695	**0.772**
PAIR(ProtT5)	0.574	0.146	0.415	**0.425**	0.312	**0.869**	**0.674**	**0.697**	0.765

Table 2. Temporal test-set performance of different models.

E. Parsing Datasets

In the following subsections, we describe how we parsed various annotations of UniProt. We downloaded a snapshot of the Swiss-Prot XML from 2023-03 (train/validation). We parsed the raw XML tree once before training. During training, we generated a data buffer by sampling a set of random proteins and a random annotation category, and then reformatted the annotations in that category to a text prompt. Each data point was added to the buffer as a dictionary containing the following information:

Training sample template

```
protein_sequence: Canonical amino acid sequence,
text_prompt: Text prompt generated from a single annotation,
pid: UniProt identifier
```

For a given batch, dictionaries from the buffer were randomly sampled. From each dictionary, we passed the value of the ”protein_sequence” key to the protein encoder and use the value of the ”text_prompt” key as the target output during training. In each paragraph below, we describe how we parsed and loaded each annotation category.

Function. The function section on UniProt, which describes any information related to the general function of the protein described in natural text. To parse the function section, we extracted the contents of all `<comment type="function"><text>` tags in the UniProt XML and added them to a list. During training, we loaded the list of function descriptions. For each description, we first removed any PubMed references. We then remove any sentences that contain the prefix ”(By similarity)”, as those are inferred from sequence similarity. Finally, we anonymized the description by replacing any instances of the protein name with ”this protein” to prevent the model from falsely achieving high performance simply by memorizing the protein name. We found that for each function description, the first sentence is often a summary of the entire paragraph. Because descriptions could be very long (making it more difficult for the model to learn), we extracted only the first sentence of each function paragraph and added each to the data buffer separately. An example of a training sample containing function information is:

Function

```
protein_sequence: MNNINKIFITFLCIELIIGGGGRLLEPLGIFPLRYLLFVFSFILLIFNLVTFN...
text_prompt: Function: may link the o-antigen tetrasaccharide units into long chains, giving rise to
typical smooth lps.
pid: P37784
```

Active sites. This fact type relates to enzymatic behaviour and describes protein sections that are involved in catalytic reactions. On UniProt, this fact type exists as a table where one column is a residue number and another describes the residue role (e.g. proton acceptor, eletrophile). We extracted the name description of the active sites and their locations from the `<feature type="active site">` tags in the XML file and created a dictionary of {residue_role: [residue_1, residue_2]}. During training, we appended all the active sites to the data buffer:

Active Sites

```
protein_sequence: MHKVTKFAIRHLSDKASRFVPKAGVYPKGYAVGGIHCGVKKDGKSLDLAILQNTFGKN...
text_prompt: Active site: nucleophile.
pid: Q5AH38
```

Binding sites. This fact type describes the interaction between protein residues and small ligands, such as metals, cofactors, and regulators. On UniProt, this fact type exists as a table, with information about the binding site type and location. During parsing, we extracted the names and locations contained within `<feature type="binding site"><ligand>` tags in the XML file and generated a dictionary where the key is the ligand and the values are a list of amino acid locations where that ligand binds. During training, we sampled a random protein and looped over all of its binding sites, adding each of them independently to the data buffer using the following format:

Binding Sites

```
protein_sequence: MRRLFTSESVTEGHPDKMCDQISDAILDAILTKDPNARVACETCTTTGLVMVMGEIST...
text_prompt: Binding sites: l-methionine
pid: B2TK10
```

Sites. The sites section on UniProt refers to any notable single amino acid sites on the protein that are not active sites or binding sites, such as cleavage sites or inhibitory sites. Similar to active sites, we extracted the name description and location information of all `<feature type="site">` tags and stored is as as a dictionary, where the keys are the site names and values are locations. During training, we loaded these annotations using the structure of the following example:

Sites

```
protein_sequence: MTWRVAVLLSLVLGAGAVPVGVDDPEDGGKHWVVIVAGSNGWYNYRHQADACHA...
text_prompt: Sites: reversibly protonated during proton transport.
pid: P69448
```

Activity regulation. The activity regulation section on UniProt describes mechanisms for regulating the activity of enzymes, transporters, and transcription factors, through both activation or inhibition. This section exists as natural language sentences. For a given protein, we extracted the contents of all `<comment type="activity regulation"><text>` tags on UniProt. If there were multiple descriptions for a given protein identifier, we combined them into a single description. During training, we first cleaned the description to remove any PubMed identifiers and added it to the buffer:

Activity Regulation

```
protein_sequence: MAEAEAGGDEARCVRLSAERAKLLLAEVDTLLFDCDGVLWRGETAVPGAPETLR...
text_prompt: Inhibited by orthovanadate, beryllium trifluoride, Ca(2+) and EDTA.
pid: Q8CHP8
```

Biophysicochemical properties. This section describes chemical and physical properties of proteins, including: reaction kinetics, light absorption, redox potentials, and dependence on temperature and pH. We extracted all tags that are children of `<comment type="biophysicochemical properties">` from the XML (for example, `<comment type="biophysicochemical properties">/<phDependence>`, `<comment type="biophysicochemical properties">/<temperatureDependence>`). We then removed any PubMed identifiers and added them to a list. Most biophysicochemical properties were natural text, but we noticed that absorption and redox potential tended to only be the numerical value and unit, and so we added the prefixes ”Absorption: ” and ”Redox potential: ” for those properties to provide more context. During training, we added each biophysicochemical property value to the data buffer separately:

Biophysicochemical property

```
protein_sequence:  MENIMTLPKIKHVRAWFIGGATAEKGAGGGDYHDQGGNHWIDDHIATPMSKYR...
text_prompt:  Biophysicochemical property:  KM is 0.25 mM for L-rhamnonate
pid:  Q8ZNF9
```

Cofactors. This section describes any non-protein entities that are required by the protein to engage in catalytic activity, such as metal atoms or vitamins. Cofactors were parsed by extracting the contents of the `<comment type="cofactor">/<cofactor>/<name>` tags and adding them all to a list. During traing, each cofactor name was added independently to the data buffer using the following format:

Cofactors

```
protein_sequence:  MKHHSLIFLTGFSGSGKSTIGPLLANSLGYDFIDLDQAIEAITGKSVSRIFA...
text_prompt:  Cofactor:  Mg(2+)
pid:  A1BER1
```

Domains. In UniProt, the Domains section describes domain(s) present in a protein. The subsection can provide both natural descriptions about the general roles of domains in the protein, as well as a list of domains from various external databases, such as Pfam [6]. We extracted the contents of all `<comment type="domain">/<text>` tags to obtain any natural language descriptions, as well as all values in `<dbReference type="Pfam">/<property type="entry name">` tags for a given protein and stored them in a list. Because the Pfam domains are abstracted as codes (e.g. "APP_E2"), we added a step when loading the data to map the Pfam code to its natural language description, obtained from UniProt documentation [7]. For example, the natural language description of "APP_E2" is "E2 domain of amyloid precursor protein". When loading the data, we looped over each domain in the list (both Pfam and natural description) and added it separately to the data buffer:

Domains

```
protein_sequence:  MAEGGSPDGRAGPGSAGRNLKEWLREQFCDHPLEHCEDTRLHDAAYVGDLQTLRSLLQEESYRSR...
text_prompt:  Domain:  Ankyrin repeats (3 copies)
pid:  Q9Y576
```

UniProt protein family.

This subsection identifies the protein family (or families) that the protein belongs to. UniProt assigns a family name based on information from external protein family databases (such as InterPro [8]), sequence similarity and analysis tools, and literature. This is documented in the "Sequence similarities" section on UniProt, and typically appears as a sentence *"Belongs to the XX family"*. We parsed the family name from the `<comment type="similarity">/<text>` tags from the UniProt XML. In the event of multiple instances, we joined them into a single string. During training, we removed any PubMed IDs from the string and loaded it into the buffer.

UniProt protein family

```
protein_sequence:  MELTITGSGKGISVSDAAFAKDFNEALVHQVVTAYMAAGRQGTKAQKTRSEVS...
text_prompt:  Belongs to the universal ribosomal protein uL4 family.
pid:A1TYJ8
```

Keywords. UniProt identifies the most relevant keywords that summarize each protein. These keywords are from a controlled vocabulary, which are structured into ten higher-level categories: *Biological process, cellular component, coding sequence diversity, developmental stage, disease, domain, ligand, molecular function, post-translational modification, technical term*. During parsing, we extracted keywords for each protein from the `<keyword>` tags, along with the keyword ID. We obtained a mapping of each keyword ID to its higher-level category [9]. When loading the data during training,

[6] http://pfam.xfam.org

[7] https://ftp.uniprot.org/pub/databases/uniprot/previous_major_releases/release-2023_02/knowledgebase/, Pfam-A.clans.tsv

[8] https://www.ebi.ac.uk/interpro/

[9] https://ftp.uniprot.org/pub/databases/uniprot/previous_major_releases/release-2023_02/knowledgebase/, keywlist.txt

we only included keywords if they did not belong to the following categories: 'Technical term', 'Developmental stage', 'Biological process', 'Cellular component', 'Molecular function'. We excluded 'Technical term' and 'Developmental stage' since they are not related to protein function and added noise to the labels. We removed 'Biological process', 'Cellular component', 'Molecular function' since these are related to GO annotations, which are downstream tasks we evaluated on.

We added each keyword to the data buffer individually:

Keywords

protein_sequence: `MSLLKMEYNLYAELKKMTCGQPISLFNEDGDFVEVEPGSSFKFLIPKG...`
text_prompt: `BKeyword: ATP-binding`
pid:`Q8QMT5`

Tissue specificity. Tissue specificity describes where mRNA and proteins are expressed in cells or tissue (if the organism is multicellular) in natural language. We extracted the contents of all `<comment type="tissue specificity">/<text>` tags and joined all descriptions into a single string. When loading this annotation during training, we removed PubMed references from the description and anonymized it from the protein name. Below is an example:

Tissue specificity

protein_sequence: `MLKRKPSNVSEKEKHQKPKRSSSFGNFDRFRNNSLSKPDDSTEAHEG...`
text_prompt: `Tissue specifity: detected in peripheral blood b-cells (at protein level). detected in spleen, liver and peripheral blood.`
pid:`Q8QMT5`

Organism. Organism is a subsection of the protein taxonomy that identifies the name of the organism the protein was sourced from. The organism name can consists of the Latin scientific name, followed by the common English name, or only the common name in the case of viruses. To parse the organism name, we extracted the values of the `<organism>/<name type="common">` and `<organism>/<name type="scientific">` tags. When loading the data, we only kept common organism names. If the name contained information about a bacterial or viral strain (e.g. Influenza A virus (strain A/Aichi/2/1968 H3N2)), we removed information related to the strain since we did not expect our model to be able to learn it well.

Organism

protein_sequence: `MPRTLSLHEITDLLETDDSIEASAIVIQPPENATAPVSDEESGDEEGGTINNLP...`
text_prompt: `Organism: Homo sapiens`
pid:`Q8N328`

Paper titles. The motivation of using paper titles as an annotation category is that they can provide a summary of important findings related to a protein. The UniProt XML contains paper titles in `<reference>/<citation>/<title>` tags, as well as an attribute related to the scope of the paper (e.g. FUNCTION, EXTRACELLULAR COPPER-BINDING DOMAIN) in `<reference key="1">/<scope>` tags. We extracted all paper titles and scopes for a protein and added them to a list. During training, we added each paper title separately to the buffer if its scope contained the word "FUNCTION":

Paper Titles

protein_sequence: `MATMQLQRTASLSALVFPNKISTEHQSLMFVKRLLAVSVSCITYLRGIFPERAYG...`
text_prompt: `Paper title: Hormad1 mutation disrupts synaptonemal complex formation, recombination, and chromosome segregation in mammalian meiosis.`
pid:`Q9D5T7`

Recommended names. Gane et al. (2023) showed that if the protein name can be predicted, it can give information about its function. Inspired by this, we extracted the Recommended name of the protein, which is the official protein name agreed upon by the UniProt consortium. We extracted the name from the `<protein>/<recommendedName>/<fullName>` tag. During training, we added it to the data buffer as shown below:

Recommended names

```
protein_sequence:  MPRTLSLHEITDLLETDDSIEASAIVIQPPENATAPVSDEESGDEEGGTINNLPGSLLHTAAYLIQDGSD...
text_prompt:  Recommended name:  PiggyBac transposable element-derived protein 3.
pid:Q8N328
```

Alternative names. We also included alternative names of the protein, which are synonyms of the recommended name. We extracted all synonyms from the `<alternativeName>/<fullName>` tags. During training, we loaded the list of alternative names for a given protein and joined them into a single prompt:

Alternative names

```
protein_sequence:  VYYSGKVDKNGKRGFTATVIPNRGAWLEYETDAKDVVYVRIDRTRKLPVTVLLRA...
text_prompt:  Alternative names:  RNA polymerase subunit beta, Transcriptase subunit beta.
pid:Q63H98
```

Subunits. Subunits describe quaternary structures of a protein, as well as how they interact with other proteins (protein-protein interactions) in natural language. We extracted the contents of each `<comment type="subunit">/<text>` tag in the UniProt XML and added each description to a list. When loading the data, we removed PubMed references and the ”(By similarity)” suffix from each description, and add the first sentence of each description as separate entries into the data buffer. Below is an example:

Subunits

```
protein_sequence:  MASANTRRVGDGAGGAFQPYLDSLRQELQQRDPTLLSVA...
text_prompt:  Subunit:  Heterodimer with SRPRA.
pid:Q4FZX7
```

Post-translational modifications. Post-translational modifications (PTMs) describe covalent modifications to amino acids, which can modulate the function of a protein. We extracted the PTMs that are described using natural language on UniProt by parsing the contents of the `<comment type="PTM">/<text>` tags. During loading, we extracted the first sentence of the description, anonymized it, removed PubMed substrings, removed any instances of the phrase ”(By similarity)”, and added it to the data buffer.

PTM

```
protein_sequence:  MKTSKLNFLTLVASTGLALAFLSGCTSNTGTTQSAKLYSEEELGLRKATIYNE...
text_prompt:  PTM: binds 2 heme c groups per subunit.
pid:Q7M963
```

Subcellular location. This section on UniProt describes the location of the protein in the cell. We extracted the contents of all `<comment type="subcellular location">/<subcellularLocation>/<location>` and `<comment type="subcellular location">/<subcellularLocation>/<topology>` tags, as well as any natural language descriptions in `<comment type="subcellular location">/<text>` and added them to separate lists (one for each tag). For each protein, we generated a dictionary with keys ”locations”, ”topologies”, and ”text”, and set the correspondings lists as values. During data loading, we passed all locations, topologies, and text descriptions separately into the data buffer.

Subcellular location

```
protein_sequence:  MPTSVPRGAPFLLLPPLLMLSAVLAVPVDRAAPPQEDSQATETPDT...
text_prompt:  Subcellular locations:  Cis-Golgi network membrane.
pid:Q02819
```

Induction. The Induction section is a natural language description of how a protein can be up-regulated or down-regulated in the presence of inducers or repressors like chemical compounds. We extracted all natural language descriptions from `<comment type="induction">/<text>` tags into a single prompt. During data loading, we removed PubMed substrings from the prompt and anonoymized it before adding it to the buffer.

Induction

```
protein_sequence:  MYVYKRDGRKEPVQFDKITARISRLCYGLDPKHIDAVKVTQRIISGVYEGVTTIELDNLA...
text_prompt:  cell cycle-regulated with highest activity in s phase.  moderately induced by dna-damage.
pid:P21524
```