Explaining embedding results for scoring alignments

by

Riley Gavigan

CS 4490Z
Thesis Supervisor: Lucian Ilie
Course Instructor: Nazim Madhavi

Department of Computer Science
University of Western Ontario, London, N6A 5B7, Ontario, Canada
2023

© Riley Gavigan 2023

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ALBERT	A Lite BERT 4
BERT	Bidirectional Encoder Representations from Transformers 3, 4
BLAST	Basic Local Alignment Search Tool 6
CDD	Conserved Domain Database 12, 13
ENNA	Evolutionary Neural Network Algorithm 5
LLM	Large Language Models 9, 10
LoRA	Low-Rank Adaptation 10
MSA	Multiple Sequence Alignments 8, 10, 12
NLP	Natural Language Processing 1–3, 6
PEFT	Parameter-Efficient Fine Tuning 10
RoBERTa	Robustly Optimized BERT 4
T5	Text-To-Text Transfer Transformer 3

Chapter 1

Introduction

Proteins are one of the four molecules of life. Finding similarities among protein sequences is essential in identifying protein structure and function. This is done by computing alignments between sequences.

The E-score method is a method to compute alignments between sequences using contextual embeddings produced by transformer models [2]. This method uses several different transformer models based off of models in Natural Language Processing (NLP).

This research addresses the results observed for the E-score method. Namely, I explain the observed cosine similarity results and explain significant differences and similarities between the models used (Table 2.2), both qualitative and quantitative. Combining the comparison of models with visualization and analysis of embedding vector and cosine similarity distributions, I propose the contributing factors to better E-score performance.

Using inference about the proposed factors contributing to E-score performance, I describe the procedure and techniques for fine-tuning ProtT5 and other models to produce better embeddings for sequence alignment.

1.1 Thesis outline

Chapter 2 provides a reader with background on important concepts and details discussed later in the thesis. Chapter 3 outlines the materials and methods used in the research conducted on the E-score method. Chapter 4 provides the results from analysis performed in the data science investigation. Chapter 5 concludes the study by addressing the research questions outlined in the thesis proposal, and discusses impact and novelty of the results.

Chapter 2

Background

2.1 Natural Language Processing

Natural Language Processing is the branch of artificial intelligence that deals with providing computers with the ability to understand text and spoken words, similar to how human being do [17]. NLP includes tasks such as summarization, sentiment analysis, and spam detection [17].

One significant advancement within NLP was the introduction of transformer models [39]. Before the introduction of Transformers within NLP, neural networks such as word2vec [26] and GloVe [31] generated contextual independent embedding vectors for words. Transformer models such as T5, BERT, ALBERT, RoBERTa, and XLNet outperformed these models with the introduction of contextual embeddings generated through self-attention [39].

The transformer models used in NLP vary significantly. Examples of differences between models includes with architecture, training procedure, and size. These differences contribute to different use cases and performance between models. The models used in the E-score method for protein sequence alignment are based on the models introduced below for NLP. Comparison between the GLUE benchmark scores for these models is shown in Table 2.1

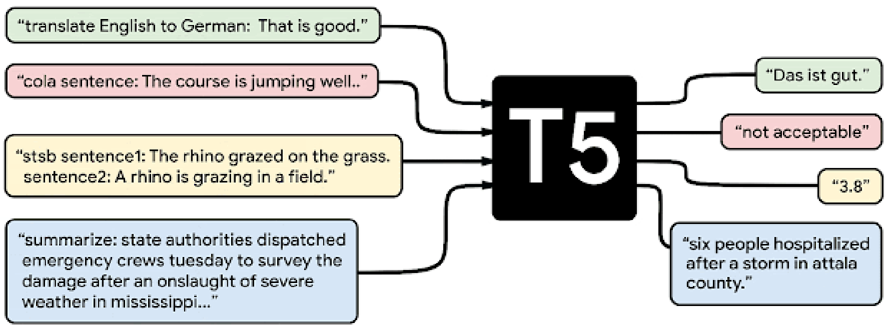
"translate English to German: That is good."
"cola sentence: The course is jumping well.."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi..."
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Figure 2.1: Diagram of T5's text-to-text framework. Each task uses text as the model input, which is trained to generate some target text [33].

2.1.1 T5

Text-To-Text Transfer Transformer (T5) uses a text-to-text approach using the transformer architecture. That is, T5's input and output are always text strings [33]. It uses both the encoder and decoder from the transformer architecture, and relies on transfer learning to fine-tune the model on downstream tasks. An example of T5's input and output is shown in Figure 2.1.

By training T5 with fill-in-the-blank-style denoising objectives, where T5 was trained to recover missing words in the input, and using transfer learning on smaller labeled datasets, T5 was able to achieve state-of-the-art NLP performance [33].

2.1.2 BERT, ALBERT, and RoBERTa

Bidirectional Encoder Representations from Transformers (BERT) achieved state-of-the-art NLP performance at the time of it being published [7]. ALBERT and RoBERTa are both derivations of BERT, and T5 also drew significant inspiration from BERT, such as the denoising objective being inspired by BERT's masked language modeling objective.

BERT is an encoder-only model that applies bidirectional training using the "masked language modeling" objective that was created for the model. This contrasts the previous framework where models read the text input sequentially, allowing for a deeper sense of context. Masked Language Modeling works by replacing 15% of the words in an input

Table 2.1: GLUE benchmark scores [41] for the Natural Language Processing models that serve as foundation for the E-score models.

Model	Avg	CoLA	SST-2	MRPC	STS-B	QQP	MNLI	RTE	WNLI
T5	88.7	71.6	97.5	92.8	93.1	75.1	92.1	92.8	94.5
XLNet	87.5	70.2	97.1	92.9	93.0	74.7	90.9	88.5	92.5
ALBERT	87.3	69.1	97.1	93.4	92.5	74.2	91.1	89.2	91.8
RoBERTa	86.4	67.8	96.7	92.3	92.2	74.3	90.5	88.2	89.0

sequence being replaced with a ”MASK” token, which the model attempts to predict the original value of based on the context from the other words [7].

A Lite BERT (ALBERT) addresses the limitations of training time and GPU/TPU memory by presenting parameter-reduction techniques for BERT [20].

Robustly Optimized BERT (RoBERTa) addresses the observation that BERT was significantly under-trained. RoBERTa improved upon BERT by training longer, removing the next-sentence pretraining objective from BERT, and training with larger mini-batches and learning rates [23].

2.1.3 XLNet

XLNet is a model that overcomes the pretrain-finetune discrepency that BERT suffers from because it relies on masking the input during training [44]. XLNet is a decoder-only model (also known as autoregressive) that overcomes BERT's limitations because of it's autoregressive formulation.

XLNet outperforms BERT significantly on 20 tasks, such as question answering and sentiment analysis [44].

2.2 Sequence alignment

Sequence similarity is essential in sequence analysis within bioinformatics [29]. Peptide sequence alignment is the most complex case, with a language of 20 common amino acid forming a theoretically countably infinite amount of unique peptide sequences shown in Equation 2.1 by taking the n-ary Cartesian product.

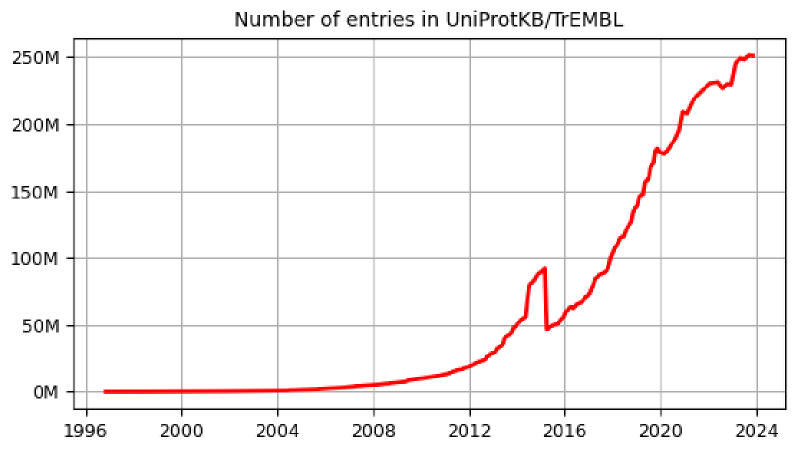

Figure 2.2: UniProtKB protein database release statistics as of May 2023 [4].

$$Theoretical\ Limit = \prod_{k=1}^{\infty} |A| = \prod_{k=1}^{\infty} 20 = 20 \times 20 \times \ldots \qquad (2.1)$$

While there is theoretically a countably infinite number of peptide sequences, the observed sequences in living organisms are constrained by biological, genetic, and functional factors. For example, the average eukaryotic protein size is 353 ± 62.5 residues [28].

Databases such as UniProt [4] and PeptideAtlas [6] are repositories filled with peptide sequences. UniProt contains over 250 million unique peptide sequences and counting, showcased in Figure 2.2.

Peptide sequences are not completely random because of the constraints imposed on them. Similar to letters or words in a given language within natural language, the frequency of each amino acid observed in nature is not equally distributed [3], which can be observed in Figure 2.3.

Proteins are also not completely random and form different secondary structures as part of the tertiary and quaternary structure of a protein. The most common of these secondary structures are α helices and β pleated sheets [24]. Because of the nature of proteins, algorithms such as an Evolutionary Neural Network Algorithm (ENNA) are able to distinguish natural proteins from randomly generated proteins with an accuracy of over 94% [5].

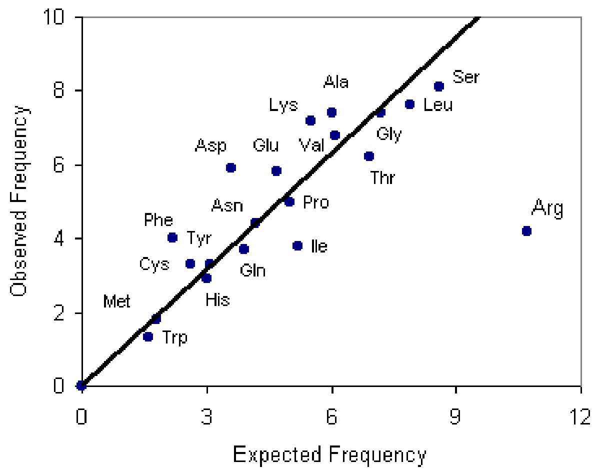

Figure 2.3: Observed frequency versus expected frequency of the 20 amino acids in vertebrates [3]

Finding similarities among protein sequences is essential in identifying protein structure and function. This is done by computing alignments between sequences. The Basic Local Alignment Search Tool (BLAST) program[1] is one of the most widely used tools in science [1]. An essential part of BLAST is the scoring function; the most widely used functions are provided by the BLOSUM matrices [12].

The E-score protein alignment scoring method [2] is another one of these scoring functions, and outperforms state-of-the-art methods. The improved performance was supported by comparing ProtT5 [8] E-score results with BLOSUM45 [12, 2].

2.3 *E*-score

E-score uses transformer models to produce contextual embeddings for the residues in peptide sequences. Model information is available in Table 2.2. These models are based off of their NLP equivalents [33, 7, 20, 44, 35].

Contextual embeddings are embeddings produced by the self-attention mechanism in the transformer architecture [39]. Similar to word embeddings in NLP, they describe the position of a residue in a high-dimensional vector space. Contextual embeddings have

[1]Exceeds 108,000 citations, according to Google Scholar.

Table 2.2: Transformer models available in the E-score method; n = number of residues. ProtT5, ProtBert, ProtAlbert, and ProtXLNet come from ProtTrans [8]. ESM1b and ESM2 come from the Meta Fundamental AI Research Protein Team [35].

Model	Architecture	Embedding Dim	Pre-Trained Dataset
ProtT5	Encoder-Decoder	n * 1024	UniRef50
ESM1b	Encoder	n * 1280	UniRef50
ESM2	Encoder	n * 1280	UniRef50
ProtBert	Encoder	n * 1024	UniRef100
ProtAlbert	Encoder	n * 4096	UniRef100
ProtXLNet	Decoder	n * 1024	UniRef100

many important applications in biology, including structure prediction [36, 43, 16] and function prediction [18, 10, 19].

The E-score alignment method is another application for these embeddings, outperforming the state-of-the-art methods [2] by completely changing the way alignments are computed.

The embedding vector produced for each protein residue varies based on the model that was used. For example, the embedding for a protein sequence of 310 residues using ProtT5 will have the dimensions [310, 1024]. The embedding dimensions are outlined in Table 2.2. The dimensionality of the embedding vectors represents the number of features encoded in the embedding, and is a fixed value for a given model.

2.3.1 Calculations

The embeddings produced by a model for a protein P, calculated in Equation 2.2, are used as the input to calculate the cosine similarity.

$$E(P) = GetEmbeddings(Model = ProtT5) \tag{2.2}$$

Calculating the cosine similarity between two vectors $A = (A_i)_{i=1..n}$ and $B = (B_i)_{i=1..n}$ is shown in Equation 2.3.

$$CosSim(A,B) = cos(\theta) \equiv \frac{A \cdot B}{\|A\|\|B\|} \equiv \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2}\sqrt{\sum\limits_{i=1}^{n} B_i^2}} \tag{2.3}$$

E-score is calculated by taking the cosine similarity between the embedding vector for two residues (i, j), shown in Equation 2.4 where P_1 and P_2 are proteins [2].

$$E\text{-}score(i,j) = CosSim(E(P_1)_i, E(P_2)_j) \tag{2.4}$$

In calculating sequence alignment using the E-score method, the cosine similarity results were mostly mostly less than $\frac{\pi}{2}$. It was also determined that ProtT5 performed better than the other models [2].

Below are two examples that demonstrate obtaining the E-score between two protein sequences.

1. Two protein sequences, P_1 and Q_1, are highly similar and have diverged slightly through evolution. The embedding vectors produced by any of the Transformer models within E-score for these sequences should be highly similar. Calculating the cosine similarity between the embedding vectors produced for P_1 and Q_2 should produce a result that is close to 1. The alignment score produced by the E-score method through dynamic programming should be high.

2. Two protein sequences, P_2 and Q_2, are very different and have diverged extensively through evolution. The embedding vectors produced for these sequence should be highly dissimilar. Calculating the cosine similarity between the embedding vectors produced for P_2 and Q_2 should produce a result that is close to -1. The alignment score produced by the E-score method through dynamic programming should be low.

2.3.2 Transformers

The transformer models used in the E-score method described in Table 2.2 vary in performance. ProtT5 outperformed the 5 other models available when computing end-gap-free alignments for six different conserved domain Multiple Sequence Alignments (MSA). ESM2 and ProtBert outperformed ProtT5 in one conserved domain each. ProtT5 and ESM2 were

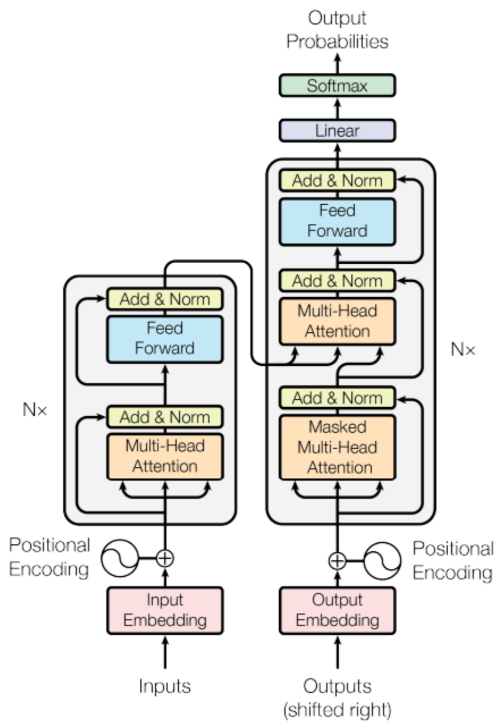

Figure 2.4: Transformer model architecture [39].

compared and it was evident that ProtT5 outperformed ESM2 with statistically significant results.

Through the results from the comparison between models, it was evident that the encoder-decoder model ProtT5 outperformed both the encoder-only models (ESM1b, ESM2, ProtBert, ProtAlbert) and the decoder-only model (XLNet).

The protein transformers models have significantly different pre-training configurations [8, 35], some of which are highlighted in Table 2.3. These configurations have a heavy impact on the performance of a model for scoring alignment and for model performance as a whole. For example, ProtT5 has 3 billion parameters compared to ProtAlbert having 224 million; as models grow their performance generally increases, which is supported by the Chinchilla paper's findings for training compute-optimal Large Language Models (LLM) [13].

Table 2.3: Pre-training configuration for protein language models [8, 35]. UR = UniRef.

Hyperparam	ProtT5	ProtBert	ProtXLNet	ProtAlbert	ESM1b	ESM2
Dataset	UR50	UR100	UR100	UR100	UR50	UR50
# of Layers	24	30	30	12	33	33
Embedding Dim	1024	1024	1024	4096	1280	1280
# of Params	3B	420M	409M	224M	650M	650M
Learning Rate	0.01	0.002	0.00001	0.002	0.0004	0.0004

2.4 Fine-tuning

Fine-tuning LLMs is a powerful technique to leverage pre-trained models and adapt them to perform better at a specific task or tasks. The purpose of fine-tuning is to avoid the need to pre-train a model from scratch for a task; instead relying on powerful pre-trained models and modifying them to better suit the task.

There are two common approaches to fine-tuning: supervised learning and reinforcement learning. Supervised learning involves providing the model with a labeled dataset, and the model will learn to map the input to the output by minimizing its loss function [27]. Reinforcement learning involves providing a reward signal to the model when it generates a desired output, and the model learns to generate the desired output for a task by maximizing the reward signal [37].

In the case of E-score, fine-tuning refers to taking a pre-existing model from Table 2.2 and training it on a dataset specific to E-score. For supervised learning, this dataset would be comprised peptide sequences and their representative MSAs. Fine-tuning would involve the model minimizing it's loss function for this dataset, and the end goal would be a fine-tuned model that outperforms the base model at scoring alignment.

There are multiple optimizations that can be made throughout the fine-tuning process to reduce the memory load and improve the efficiency of the process. Parameter-Efficient Fine Tuning (PEFT) is a memory optimization that either selects a subset of the LLMs initial parameters to fine-tune; or freezes the layers of the LLM and introduces a small number of new, trainable layers that the fine-tuned model will use [21]. Another PEFT technique is Low-Rank Adaptation (LoRA), which freezes the pre-trained weights and injects a trainable rank decomposition matrix into each layer of the architecture with much smaller dimensions than the base model [14].

Chapter 3

Materials & Methods

3.1 Comparing model properties

To support findings from embedding vector and cosine similarity analysis, background knowledge about the properties of different models was used to explain the performance differences. Table 2.3 highlights some key properties about the models available in the E-score method.

Results from the papers proposing each model are used to support findings in Chapter 4. Details regarding the ProtTrans models are in [8, 9]. Details regarding ESM-1b are in [35, 34]. Details regarding ESM2 are in [22].

3.2 Fine-tuning

ProtT5 serves as the base model for fine-tuning for improve E-score performance, as it was the best-performing model [2].

The ProtTrans project includes Jupyter Notebooks for fine-tuning ProtT5[1]. These notebooks are used to fine-tune the model for our purposes.

[1]https://github.com/agemagician/ProtTrans/tree/master/Fine-Tuning

Table 3.1: 10 MSAs with the most proteins from CDD used in the E-score comparison procedure [2]

MSAs			
Conserved Domain	Source	Proteins	Length
CS_CSD	cd00024	522	98
$7tm\_classA\_rhodopsinlike$	cd00637	405	808
$FYVE\_like\_SF$	cd00065	392	266
$Mblike$	cd01040	384	239
$SH2$	cd00173	352	214
$C1$	cd00029	281	99
$KAZAL\_FS$	cd00104	273	74
$Globin\_sensor$	cd01068	193	223
$Bbox2$	cd19756	127	65
$NBD\_sugarkinase\_HSP70\_actin$	cd00012	125	1154

3.2.1 Data

Fine-tuning data is obtained through the Conserved Domain Database (CDD) [25] for different MSAs. We use as many pairs of sequences as desired for a given MSA of the 49 MSAs selected in the E-score paper, those of which have the most proteins are shown in Table 3.1. These pairs of sequences serve as the training and validation datasets for the model. For each pair, the desired output is a distance of 0 $(d_{cc}, d_d, d_{pos}, d_{ssp}, d_{seq}, d_{pos})$ between the reference alignment from CDD and the pairwise alignment generated from the model.

Obtaining the data works as follows:

1. Select a source from Table 3.1 for a conserved domain.
2. Search for the source on the CDD website (https://www.ncbi.nlm.nih.gov/cdd).
3. Click 'Download alignment' to download a FASTA file of each sequence in the MSA.

With the data selected from the CDD, pairs can be easily enumerated by iterating through each pair within the FASTA file and obtaining every pair of i, j sequences where $i \neq j$.

3.2.2 Validation

The same procedure used in the E-score paper is used to validate the performance of the fine-tuned ProtT5 model against the base ProtT5 model for alignment [2]. This includes comparing the fine-tuned model against the base ProtT5 model and BLOSUM45 [12] for a significant amount of pairs from the 49 MSAs selected from CDD.

References

[1] S. F. Altschul et al. “Basic local alignment search tool”. In: *Journal of molecular biology* (1990). DOI: 10.1016/S0022-2836(05)80360-2.

[2] S. Ashrafzadeh et al. “Scoring alignments by embedding vector similarity”. In: (2023). DOI: 10.1101/2023.08.30.555602.

[3] M. Beals, L. Gross, and S. Harrell. “Amino Acid Frequency”. In: (1999). URL: https://www.nimbios.org/~gross/bioed/webmodules/aminoacid.htm.

[4] T. U. Consortium. “UniProt: the Universal Protein Knowledgebase in 2023”. In: *Nucleic Acids Research* 51.D1 (Nov. 2022), pp. D523–D531. ISSN: 0305-1048. DOI: 10.1093/nar/gkac1052.

[5] D. De Lucrezia et al. “Do Natural Proteins Differ from Random Sequences Polypeptides? Natural vs. Random Proteins Classification Using an Evolutionary Neural Network”. In: *PLOS ONE* 7.5 (May 2012), pp. 1–10. DOI: 10.1371/journal.pone.0036634.

[6] F. Desiere et al. “The PeptideAtlas project”. In: *Nucleic Acids Research* 34.suppl$_1$ (Jan. 2006), pp. D655–D658. ISSN: 0305-1048. DOI: 10.1093/nar/gkj040.

[7] J. Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: (2018). DOI: 10.48550/arXiv.1810.04805.

[8] A. Elnaggar et al. “ProtTrans: Towards Cracking the Language of Lifes Code Through Self-Supervised Deep Learning and High Performance Computing”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021). DOI: 10.1109/TPAMI.2021.3095381.

[9] A. Elnaggar et al. “ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning”. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 44.10 (2022), pp. 7112–7127. DOI: 10.1109/TPAMI.2021.3095381.

[10] V. Gligorijević et al. "Structure-based protein function prediction using graph convolutional networks". In: *Nature Communications* 12.1 (2021), p. 3168. ISSN: 2041-1723. DOI: 10.1038/s41467-021-23303-9.

[11] C. R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[12] S. Henikoff and J. G. Henikoff. "Amino acid substitution matrices from protein blocks". In: *Proceedings of the National Academy of Sciences* (1992). DOI: 10.1073/pnas.89.22.10915.

[13] J. Hoffmann et al. *Training Compute-Optimal Large Language Models*. 2022. arXiv: 2203.15556 [cs.CL].

[14] E. J. Hu et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. arXiv: 2106.09685 [cs.CL].

[15] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[16] J. Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *Nature* 596.7873 (2021), pp. 583–589. ISSN: 1476-4687. DOI: 10.1038/s41586-021-03819-2.

[17] D. Khurana et al. "Natural language processing: state of the art, current trends and challenges". In: *Multimedia Tools and Applications* 82.3 (2023), pp. 3713–3744. ISSN: 1573-7721. DOI: 10.1007/s11042-022-13428-4.

[18] M. Kulmanov and R. Hoehndorf. "DeepGOPlus: improved protein function prediction from sequence". In: *Bioinformatics* 36.2 (2019), pp. 422–429. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btz595.

[19] B. Lai and J. Xu. "Accurate Protein Function Prediction via Graph Attention Networks with Predicted Structure Information". In: *bioRxiv* (2021). DOI: 10.1101/2021.06.16.448727.

[20] Z. Lan et al. *ALBERT: A Lite BERT for Self-supervised Learning of Language Representations*. 2020. arXiv: 1909.11942 [cs.CL].

[21] V. Lialin, V. Deshpande, and A. Rumshisky. *Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning*. 2023. arXiv: 2303.15647 [cs.CL].

[22] Z. Lin et al. "Language models of protein sequences at the scale of evolution enable accurate structure prediction". In: *bioRxiv* (2022).

[23] Y. Liu et al. "Roberta: A robustly optimized bert pretraining approach". In: (2019). DOI: 10.48550/arXiv.1907.11692.

[24] Y. Ma, Y. Liu, and J. Cheng. “Protein Secondary Structure Prediction Based on Data Partition and Semi-Random Subspace Method”. In: *Scientific Reports* 8.1 (2018), p. 9856. ISSN: 2045-2322. DOI: 10.1038/s41598-018-28084-8.

[25] A. Marchler-Bauer et al. “CDD: NCBI’s conserved domain database”. In: (2015). DOI: 10.1093/nar/gku1221.

[26] T. Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[27] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. 2nd ed. Adaptive Computation and Machine Learning. Cambridge, MA: MIT Press, 2018. 504 pp. ISBN: 978-0-262-03940-6.

[28] Y. Nevers et al. “Protein length distribution is remarkably uniform across the tree of life”. In: *Genome Biology* 24.1 (2023), p. 135. ISSN: 1474-760X. DOI: 10.1186/s13059-023-02973-2.

[29] D. Ofer, N. Brandes, and M. Linial. “The language of proteins: NLP, machine learning & protein sequences”. In: *Computational and Structural Biotechnology Journal* 19 (2021), pp. 1750–1758. ISSN: 2001-0370. DOI: https://doi.org/10.1016/j.csbj.2021.03.022.

[30] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[31] J. Pennington, R. Socher, and C. Manning. “GloVe: Global Vectors for Word Representation”. In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Ed. by A. Moschitti, B. Pang, and W. Daelemans. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162.

[32] M. E. Peters et al. “Deep contextualized word representations”. In: (2018). DOI: 10.48550/arXiv.1802.05365.

[33] C. Raffel et al. “Exploring the limits of transfer learning with a unified text-to-text transformer”. In: (2020). DOI: 10.48550/arXiv.1910.10683.

[34] R. Rao et al. “Transformer protein language models are unsupervised structure learners”. In: *bioRxiv* (2020). DOI: 10.1101/2020.12.15.422761.

[35] A. Rives et al. “Biological Structure and Function Emerge from Scaling Unsupervised Learning to 250 Million Protein Sequences”. In: *PNAS* (2019). DOI: 10.1101/622803.

[36] A. W. Senior et al. “Improved protein structure prediction using potentials from deep learning”. In: *Nature* 577.7792 (2020), pp. 706–710. DOI: 10.1038/s41586-019-1923-7.

[37] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018. URL: https://inst.eecs.berkeley.edu/~cs188/sp20/assets/files/SuttonBartoIPRLBook2ndEd.pdf.

[38] B. E. Suzek et al. “UniRef: comprehensive and non-redundant UniProt reference clusters”. In: *Bioinformatics* 23.10 (2007), pp. 1282–1288. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btm098.

[39] A. Vaswani et al. “Attention is all you need”. In: (2017). DOI: 10.48550/arXiv.1706.03762.

[40] P. Virtanen et al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[41] A. Wang et al. *GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding*. 2019. arXiv: 1804.07461 [cs.CL].

[42] M. L. Waskom. “seaborn: statistical data visualization”. In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021.

[43] J. Yang et al. “Improved protein structure prediction using predicted inter-residue orientations”. In: *bioRxiv* (2019). DOI: 10.1101/846279.

[44] Z. Yang et al. “XLNet: Generalized autoregressive pretraining for language understanding”. In: *Advances in neural information processing systems* (2019). DOI: 10.48550/arXiv.1906.08237.

Glossary

amino acid molecules that combine to form proteins, containing both an amino and a carboxyl group 4

denoising the process of the auto-encoder in a Transformer learning to capture the most important features of the data distribution, ignoring the noise 3

peptide a compound consisting of two or more amino acids linked in a chain, the carboxyl group of each acid being joined to the amino group of the next plural 4–6

residue a single unit that makes up a polymer, such as an amino acid in a polypeptide chain plural 5–8

transfer learning a technique in machine learning in which knowledge learned from a task is re-used to boost performance on a related task. a model already developed for 1 task is re-used in another task. 3

transformer a type of neural network architecture used to solve the problem of transduction or transformation of input sequences into output sequences in deep learning applications using self-attention 1–3, 6, 8