Explaining embedding results for scoring alignments

Final Progress Report

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
2024

Glossary

ALBERT A Lite BERT 8

BERT Bidirectional Encoder Representations from Transformers 7, 8

BLAST Basic Local Alignment Search Tool 10

BLOSUM BLOcks SUbstitution Matrix 10

CDD Conserved Domain Database 17, 18

ENNA Evolutionary Neural Network Algorithm 10

LLM Large Language Models 13

LoRA Low-Rank Adaptation 13

MSA Multiple Sequence Alignments 12, 13, 17, 18

NLP Natural Language Processing 6, 7, 10, 16

PEFT Parameter-Efficient Fine Tuning 13

RoBERTa Robustly Optimized BERT 8

T5 Text-To-Text Transfer Transformer 7

Structured Abstract

Context and motivation

The *E*-score protein alignment scoring method (Ashrafzadeh et al., 2023) outperforms state-of-the-art methods, supported by comparing ProtT5 (Elnaggar et al., 2021) *E*-score results with BLOSUM45 (Henikoff & Henikoff, 1992).

This research aimed to understand *E*-score results, building upon the observation that mean cosine similarity results between two embeddings are not evenly distributed.

By understanding the underlying causes of the observed results, we can improve the *E*-score method. Insights can be used to fine-tune the transformer models (Elnaggar et al., 2021; Rives et al., 2019) and performance of embeddings.

Research questions

- What properties of embeddings produce better cosine similarity results?
- Why do cosine similarity results primarily fall within a positive range?
- How can models be fine-tuned to produce better embeddings?

Principal ideas

Positive cosine similarity results imply the produced embeddings are mostly similar. Comparing different embedding types provided insight into their distributions. Through these comparisons, conclusions about properties that improve *E*-score results were drawn.

Research methodology

This research is a data science investigation to obtain insight about the embeddings and cosine similarity results in the *E*-score method.

Anticipated results

This study primarily aimed to obtain insight and knowledge for the *E*-score method, specifically:

- Knowledge about the distributions of different embedding types
- Knowledge about the cosine similarity between embeddings
- Insight to fine-tune and improve models

Novelty

By building upon a novel method for scoring protein alignments using cosine similarity (Ashrafzadeh et al., 2023), novel conclusions about embeddings and cosine similarity were made, leading to further research that can improve embeddings and models.

Impact

Improvements in transformer models for the E-score alignment scoring method can be made through the insight this research found. Any anticipated improvements would also be applicable to Natural Language Processing Models, such as T5 (Raffel et al., 2020).

Progress and completed work

Insight into the properties behind embedding type distributions was obtained, and from this cosine similarity results were explained. These properties were explained through conducted research and simulation in combination with insight from biochemistry background research.

Limitations

No limitations are known to exist in this research.

Table of Contents

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 Ashrafzadeh et al., 2023. 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.

Background and Related Work

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 Khurana et al., 2023. NLP includes tasks such as summarization, sentiment analysis, and spam detection Khurana et al., 2023.

One significant advancement within NLP was the introduction of transformer models Vaswani et al., 2017. Before the introduction of Transformers within NLP, neural networks such as word2vec Mikolov et al., 2013 and GloVe Pennington et al., 2014 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 Vaswani et al., 2017.

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

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 Raffel et al., 2020. 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 Raffel et al., 2020.

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 Devlin et al., 2018. 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.

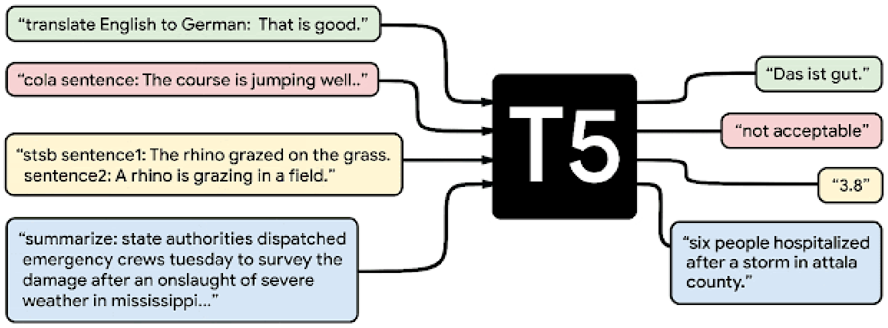

"translate English to German: That is good."
"cola sentence: The course is jumping well.."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather 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 Raffel et al., 2020.

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 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 Devlin et al., 2018.

A Lite BERT (ALBERT) addresses the limitations of training time and GPU/TPU memory by presenting parameter-reduction techniques for BERT Lan et al., 2020.

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 Liu et al., 2019.

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 Z. Yang et al., 2019. 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 Z. Yang et al., 2019.

2.2 Sequence alignment

Sequence similarity is essential in sequence analysis within bioinformatics Ofer et al., 2021. Peptide sequence alignment is the most complex case, with a language of 20 common amino

Table 2.1: GLUE benchmark scores Wang et al., 2019 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

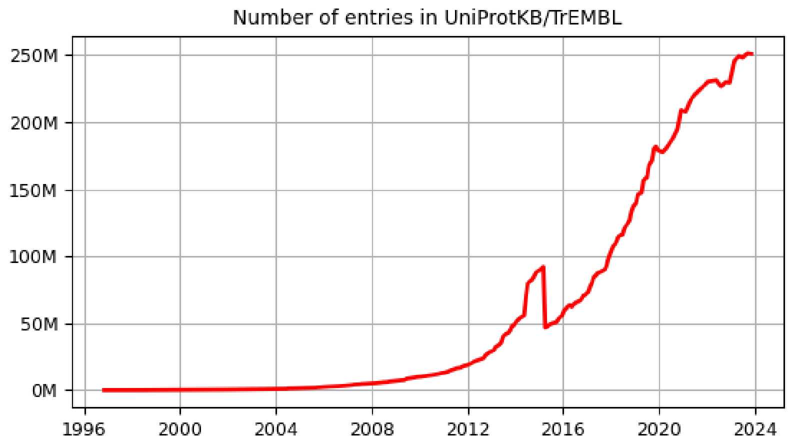


Figure 2.2: UniProtKB protein database release statistics as of May 2023 Consortium, 2022.

acid forming a theoretically countably infinite amount of unique peptide sequences shown in Equation 2.1 by taking the n-ary Cartesian product.

$$Theoretical\,Limit = \prod_{k=1}^{\infty} |A| = \prod_{k=1}^{\infty} 20 = 20 \times 20 \times \dots \tag{2.1}$$

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 Nevers et al., 2023.

Databases such as UniProt Consortium, 2022 and PeptideAtlas Desiere et al., 2006 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 Beals et al., 1999, 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

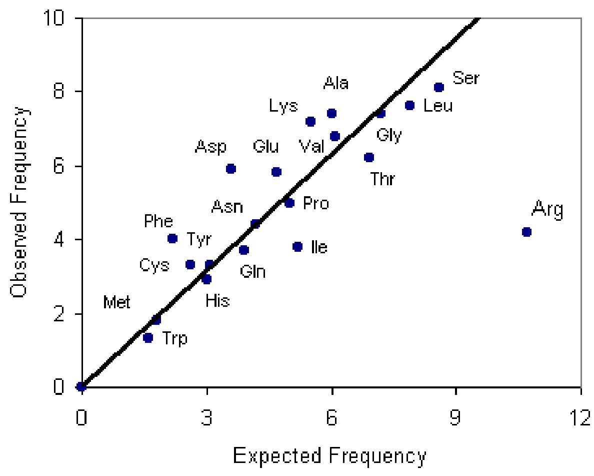


Figure 2.3: Observed frequency versus expected frequency of the 20 amino acids in vertebrates Beals et al., 1999

α helices and β pleated sheets Ma et al., 2018. 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% De Lucrezia et al., 2012.

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 Altschul et al., 1990. An essential part of BLAST is the scoring function; the most widely used functions are provided by the BLOcks SUbstitution Matrix (BLOSUM) Henikoff and Henikoff, 1992.

The *E*-score protein alignment scoring method Ashrafzadeh et al., 2023 is another one of these scoring functions, and outperforms state-of-the-art methods. The improved performance was supported by comparing ProtT5 Elnaggar et al., 2021 *E*-score results with BLOSUM45 Ashrafzadeh et al., 2023; Henikoff and Henikoff, 1992.

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 Devlin et al., 2018; Lan et al., 2020; Raffel et al., 2020; Rives et al., 2019; Z. Yang et al., 2019.

Contextual embeddings are embeddings produced by the self-attention mechanism in the transformer architecture Vaswani et al., 2017. Similar to word embeddings in NLP, they describe the position of a residue in a high-dimensional vector space. Contextual embeddings have many important applications in biology, including structure prediction Jumper et al., 2021; Senior et

[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 Elnaggar et al., 2021. ESM1b and ESM2 come from the Meta Fundamental AI Research Protein Team Rives et al., 2019.

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

al., 2020; J. Yang et al., 2019 and function prediction Gligorijević et al., 2021; Kulmanov and Hoehndorf, 2019; Lai and Xu, 2021.

The E-score alignment method is another application for these embeddings, outperforming the state-of-the-art methods Ashrafzadeh et al., 2023 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.

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.

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

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 Ashrafzadeh et al., 2023.

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

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 Ashrafzadeh et al., 2023.

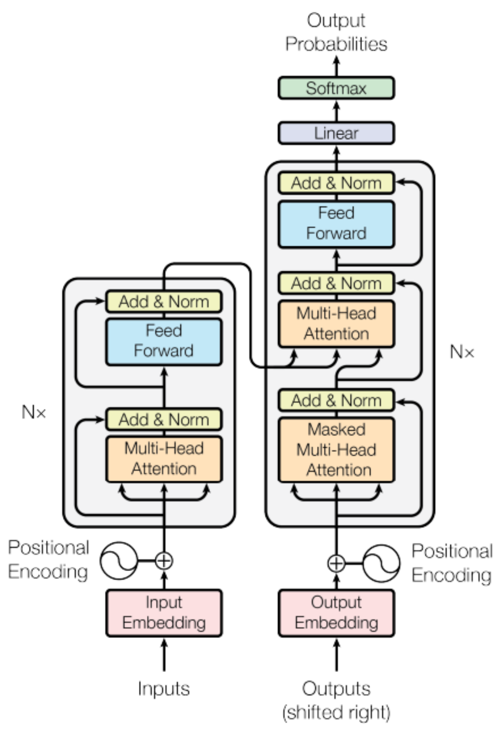


Figure 2.4: Transformer model architecture Vaswani et al., 2017.

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.

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 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).

Table 2.3: Pre-training configuration for protein language models Elnaggar et al., 2021; Rives et al., 2019. 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

The protein transformers models have significantly different pre-training configurations Elnaggar et al., 2021; Rives et al., 2019, 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) Hoffmann et al., 2022.

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 Mohri et al., 2012/2018. 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 Sutton and Barto, 2018.

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 Lialin et al., 2023. 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 Hu et al., 2021.

Research Objectives

- O1: Understand the reasoning behind the observed distributions of different embedding types. Explaining both individual and relative results for each model in the E-score method.
- O2: Understand what properties of embeddings help produce better cosine similarity and alignment results.
- O3: Understand why cosine similarity results primarily fall within a positive range.
- O4: Determine how models can be fine-tuned to improve E-score method results.

Methodology

4.1 Cosine similarity analysis

Proteins are not completely random in nature. By showcasing the frequencies of amino acids in our dataset of protein sequences, we showcase that there is not an equal distribution of amino acids present in nature. We also use these frequencies to perform a simulation on completely random proteins for a given length n of a polypeptide. By simulating every combination and calculating the cosine similarity for a given length of proteins using only the frequency of amino acids as a constraint, we are able to showcase one reason for cosine similarity results.

By applying the above analysis and further supporting it with more properties of proteins such as their secondary structures, we analyze and explain why cosine similarity results are mostly positive. Similar to how in NLP we would observe documents having similar sentences, the rules that proteins follow would result in similarities between sequences. An example from NLP would be that emails would commonly contain phrases such as ”I am emailing you in regards to...” or ”I am writing to you to follow up on...”, which would naturally be seen as more similar than dissimilar, resulting in positive cosine similarity results. In proteins, we would observe similarities such as alpha helices and beta sheets being correlated with primary sequences. AlphaFold Jumper et al., 2021 is a protein structure prediction method developed by Google DeepMind that uses protein transformers as the E-score method does. Because we are able to predict protein structure with transformers, it is evident that primary structures, secondary structures, structural motifs, and other properties of proteins are heavily correlated.

4.2 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 Elnaggar et al., 2021; Elnaggar et al., 2022. Details regarding ESM-1b are in Rao et al., 2020; Rives et al., 2019. Details regarding ESM2 are in Lin et al., 2022.

Table 4.1: 10 MSAs with the most proteins from CDD used in the E-score comparison procedure Ashrafzadeh et al., 2023.

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

4.3 Fine-tuning

4.3.1 Model

ProtT5 serves as the base model for fine-tuning for improve E-score performance, as it was the best-performing model Ashrafzadeh et al., 2023.

The ProtTrans project includes Jupyter Notebooks for fine-tuning ProtT5. The per-protein notebook will be used to fine-tune the model for sequence alignment scoring. Regression is preferred over classification because the results from the E-score method are not possible to classify into a few discrete classes; we fine-tune the model based on its distance from the reference alignment.

The ProtT5 fine-tuning notebook is heavily modified to support the specific fine-tuning case of taking as input a pair of sequences at a time, getting the embedding vectors for both sequences, computing the e-score alignment, and returning the distances between the pairwise alignment and the CDD reference alignment.

4.3.2 Dataset

Fine-tuning data is obtained through the Conserved Domain Database (CDD) Marchler-Bauer et al., 2015 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 4.1. These pairs of sequences serve as the training and validation datasets for the model. For each pair, the desired output is an average distance of 0 (d_{seq}, d_{pos}, d_{ssp}, d_c, d_d) between the reference alignment from CDD and the pairwise alignment generated from the model.

Obtaining each protein sequence from CDD:

1. Select a source from Table 4.1 for a conserved domain.
2. Search for the source on the CDD website.
3. Click ’Representatives’ under the ’Links’ section.
4. Click Send to: File. Select FASTA format.

Obtaining the reference alignments from CDD for each protein:

1. Select a source from Table 4.1 for a conserved domain.
2. Search for the source on the CDD website.
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 the FASTA file and obtaining every pair i, j of sequences, where $i \neq j$. The dataset is provided when fine-tuning the model as pairs of (sequence i, sequence j, 0), where 0 is the expected value from the reference alignment for each distance measure.

4.3.3 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 Ashrafzadeh et al., 2023. This includes comparing the fine-tuned model against the base ProtT5 model and BLOSUM45 Henikoff and Henikoff, 1992 for a significant amount of pairs from the 49 MSAs selected from CDD.

Results

The distribution of the observed amino acids in all of the protein sequences from the 10 MSAs in Table 4.1 is shown in Table 5.1.

Table 5.1: Distribution of amino acids found in the 10 selected MSAs. A few occurrences of 'B' (nondeterministically either N or D) and some occurrences of 'X' (undetermined or atypical amino acid) were left out for simplicity.

Amino Acid	Symbol	Frequency	Percent	Diff From Equal	P-value
Leucine	L	152859	9.099	4.099	0.0e+00
Serine	S	141844	8.443	3.443	0.0e+00
Alanine	A	127926	7.614	2.614	0.0e+00
Glutamic Acid	E	108476	6.457	1.457	0.0e+00
Valine	V	105408	6.274	1.274	0.0e+00
Arginine	R	99687	5.934	0.934	3.2e-293
Glycine	G	96906	5.768	0.768	3.6e-202
Threonine	T	96702	5.756	0.756	4.1e-196
Lysine	K	94251	5.610	0.610	3.6e-130
Aspartic Acid	D	88980	5.296	0.296	5.2e-33
Isoleucine	I	87579	5.213	0.213	5.9e-18
Proline	P	86463	5.146	0.146	2.5e-09
Glutamine	Q	74206	4.417	0.583	6.3e-134
Asparagine	N	73490	4.374	0.626	1.3e-154
Phenylalanine	F	64495	3.839	1.161	0.0e+00
Tyrosine	Y	46324	2.757	2.243	0.0e+00
Histidine	H	43163	2.569	2.431	0.0e+00
Cysteine	C	36749	2.187	2.813	0.0e+00
Methionine	M	35289	2.100	2.900	0.0e+00
Tryptophan	W	19243	1.145	3.855	0.0e+00

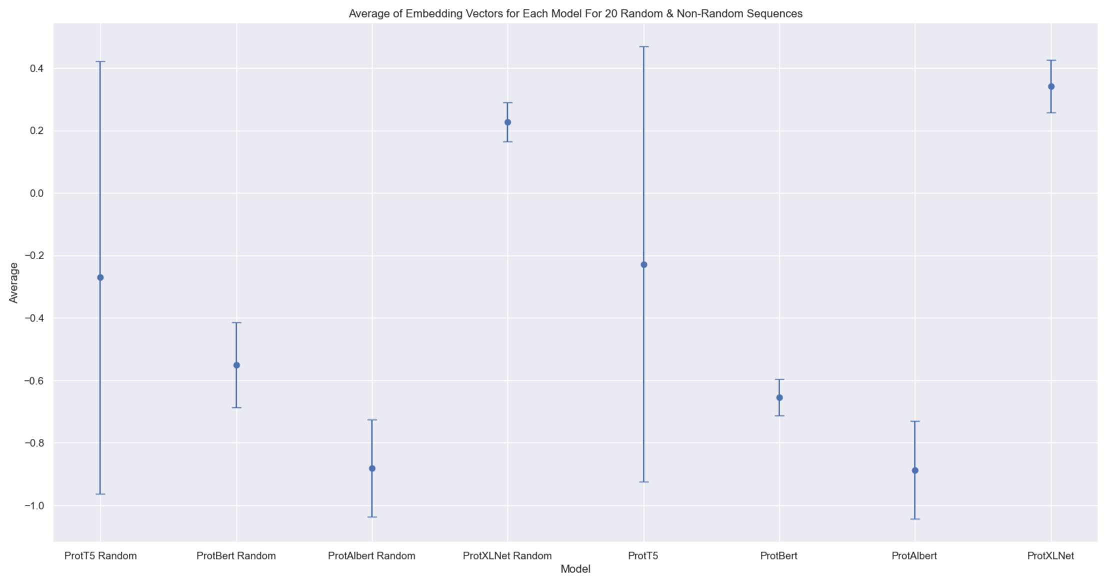


Figure 5.1: Average value of embeddings for random and non-random embeddings. Values all scaled to -1...1.

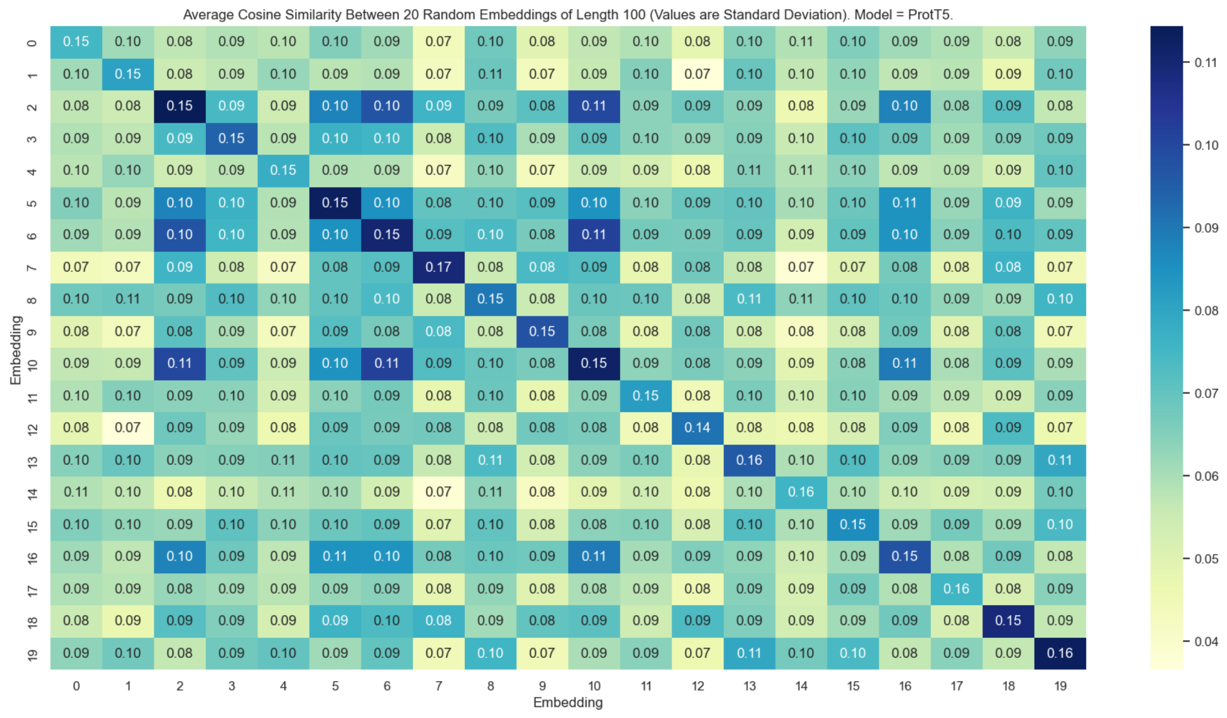
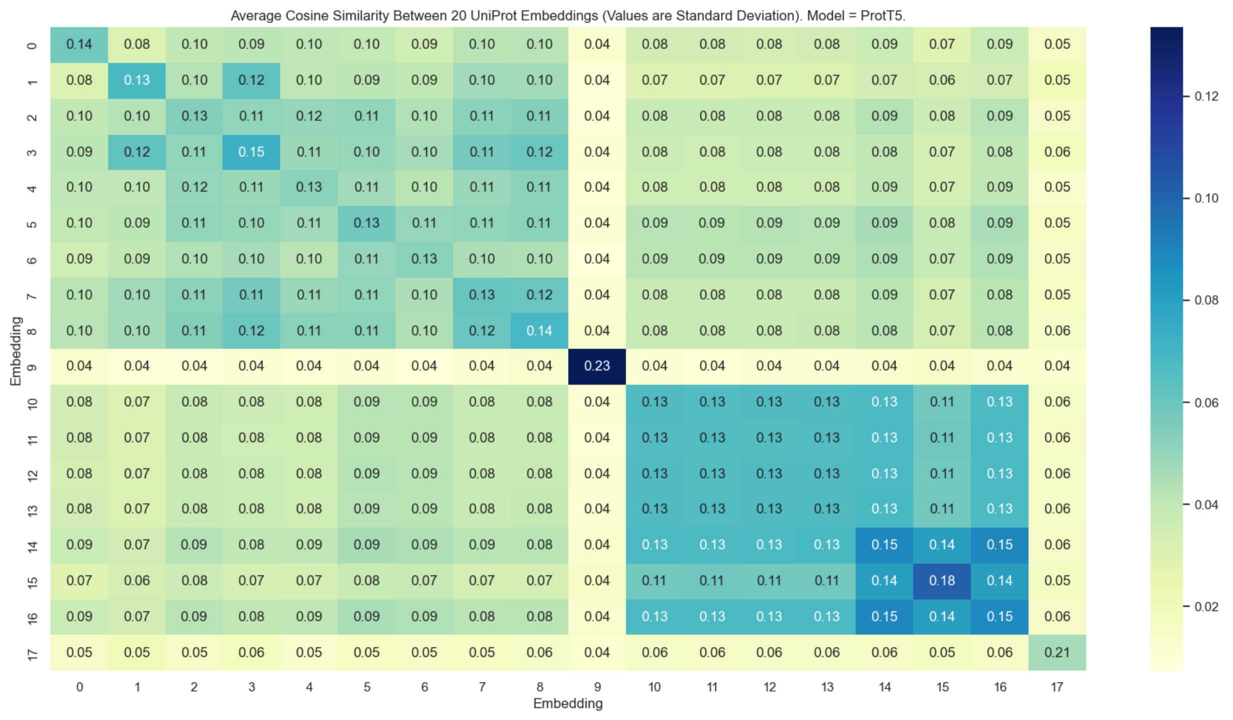


Figure 5.2: Average cosine similarity between 20 non-random embeddings for ProtT5. Values are the standard deviation.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0.14	0.08	0.10	0.09	0.10	0.10	0.09	0.10	0.10	0.04	0.08	0.08	0.08	0.08	0.09	0.07	0.09	0.05
1	0.08	0.13	0.10	0.12	0.10	0.09	0.09	0.10	0.10	0.04	0.07	0.07	0.07	0.07	0.07	0.06	0.07	0.05
2	0.10	0.10	0.13	0.11	0.12	0.11	0.10	0.11	0.11	0.04	0.08	0.08	0.08	0.08	0.09	0.08	0.09	0.05
3	0.09	0.12	0.11	0.15	0.11	0.10	0.10	0.11	0.12	0.04	0.08	0.08	0.08	0.08	0.08	0.07	0.08	0.06
4	0.10	0.10	0.12	0.11	0.13	0.11	0.10	0.11	0.11	0.04	0.08	0.08	0.08	0.08	0.09	0.07	0.09	0.05
5	0.10	0.09	0.11	0.10	0.11	0.13	0.11	0.11	0.11	0.04	0.09	0.09	0.09	0.09	0.09	0.08	0.09	0.05
6	0.09	0.09	0.10	0.10	0.10	0.11	0.13	0.10	0.10	0.04	0.09	0.09	0.09	0.09	0.09	0.07	0.09	0.05
7	0.10	0.10	0.11	0.11	0.11	0.11	0.10	0.13	0.12	0.04	0.08	0.08	0.08	0.08	0.09	0.07	0.08	0.05
8	0.10	0.10	0.11	0.12	0.11	0.11	0.10	0.12	0.14	0.04	0.08	0.08	0.08	0.08	0.08	0.07	0.08	0.06
9	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.23	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.04
10	0.08	0.07	0.08	0.08	0.08	0.09	0.09	0.08	0.08	0.04	0.13	0.13	0.13	0.13	0.13	0.11	0.13	0.06
11	0.08	0.07	0.08	0.08	0.08	0.09	0.09	0.08	0.08	0.04	0.13	0.13	0.13	0.13	0.13	0.11	0.13	0.06
12	0.08	0.07	0.08	0.08	0.08	0.09	0.09	0.08	0.08	0.04	0.13	0.13	0.13	0.13	0.13	0.11	0.13	0.06
13	0.08	0.07	0.08	0.08	0.08	0.09	0.09	0.08	0.08	0.04	0.13	0.13	0.13	0.13	0.13	0.11	0.13	0.06
14	0.09	0.07	0.09	0.08	0.09	0.09	0.09	0.09	0.08	0.04	0.13	0.13	0.13	0.13	0.15	0.14	0.15	0.06
15	0.07	0.06	0.08	0.07	0.07	0.08	0.07	0.07	0.07	0.04	0.11	0.11	0.11	0.11	0.14	0.18	0.14	0.05
16	0.09	0.07	0.09	0.08	0.09	0.09	0.09	0.08	0.08	0.04	0.13	0.13	0.13	0.13	0.15	0.14	0.15	0.06
17	0.05	0.05	0.05	0.06	0.05	0.05	0.05	0.05	0.06	0.04	0.06	0.06	0.06	0.06	0.06	0.05	0.06	0.21

Figure 5.3: Average cosine similarity between 20 randomly generated embeddings for ProtT5. Values are the standard deviation.

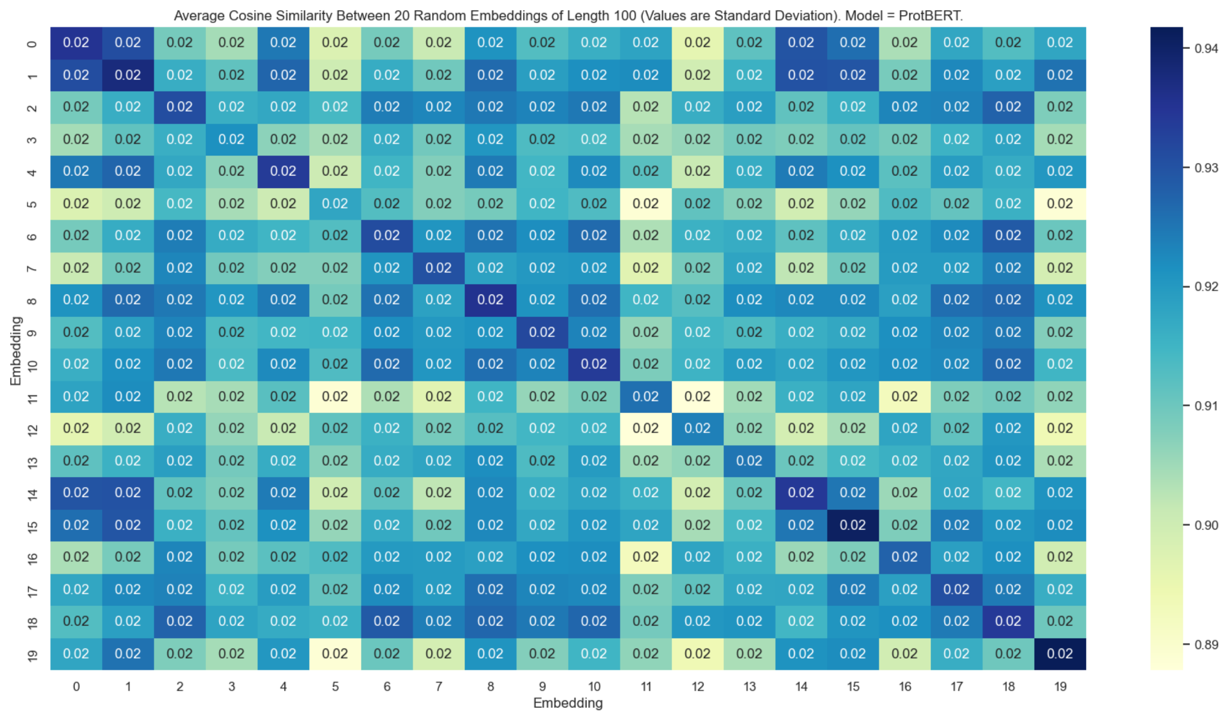
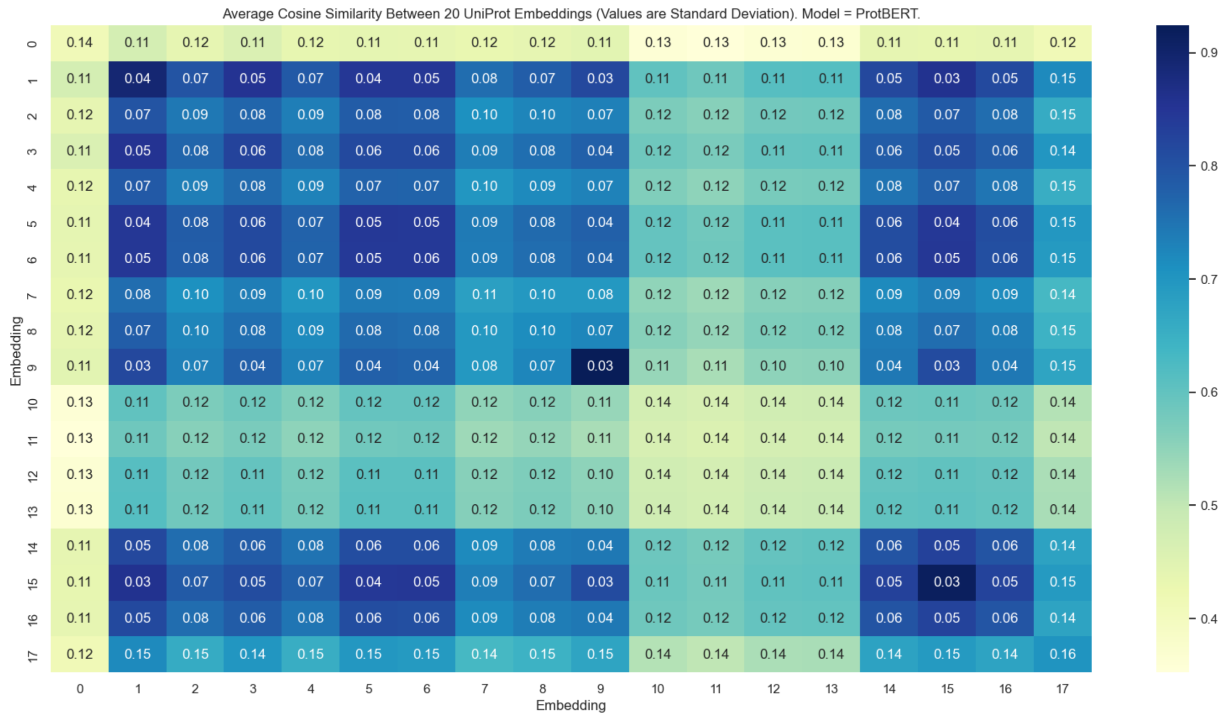


Figure 5.4: Average cosine similarity between 20 randomly generated embeddings for ProtBERT. Values are the standard deviation.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	0.14	0.11	0.12	0.11	0.12	0.11	0.11	0.12	0.12	0.11	0.13	0.13	0.13	0.13	0.11	0.11	0.11	0.12
1	0.11	0.04	0.07	0.05	0.07	0.04	0.05	0.08	0.07	0.03	0.11	0.11	0.11	0.11	0.05	0.03	0.05	0.15
2	0.12	0.07	0.09	0.08	0.09	0.08	0.08	0.10	0.10	0.07	0.12	0.12	0.12	0.12	0.08	0.07	0.08	0.15
3	0.11	0.05	0.08	0.06	0.08	0.06	0.06	0.09	0.08	0.04	0.12	0.12	0.11	0.11	0.06	0.05	0.06	0.14
4	0.12	0.07	0.09	0.08	0.09	0.07	0.07	0.10	0.09	0.07	0.12	0.12	0.12	0.12	0.08	0.07	0.08	0.15
5	0.11	0.04	0.08	0.06	0.07	0.05	0.05	0.09	0.08	0.04	0.12	0.12	0.11	0.11	0.06	0.04	0.06	0.15
6	0.11	0.05	0.08	0.06	0.07	0.05	0.06	0.09	0.08	0.04	0.12	0.12	0.11	0.11	0.06	0.05	0.06	0.15
7	0.12	0.08	0.10	0.09	0.10	0.09	0.09	0.11	0.10	0.08	0.12	0.12	0.12	0.12	0.09	0.09	0.09	0.14
8	0.12	0.07	0.10	0.08	0.09	0.08	0.08	0.10	0.10	0.07	0.12	0.12	0.12	0.12	0.08	0.07	0.08	0.15
9	0.11	0.03	0.07	0.04	0.07	0.04	0.04	0.08	0.07	0.03	0.11	0.11	0.10	0.10	0.04	0.03	0.04	0.15
10	0.13	0.11	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.11	0.14	0.14	0.14	0.14	0.12	0.11	0.12	0.14
11	0.13	0.11	0.12	0.12	0.12	0.12	0.12	0.12	0.12	0.11	0.14	0.14	0.14	0.14	0.12	0.11	0.12	0.14
12	0.13	0.11	0.12	0.11	0.12	0.11	0.11	0.12	0.12	0.10	0.14	0.14	0.14	0.14	0.12	0.11	0.12	0.14
13	0.13	0.11	0.12	0.11	0.12	0.11	0.11	0.12	0.12	0.10	0.14	0.14	0.14	0.14	0.12	0.11	0.12	0.14
14	0.11	0.05	0.08	0.06	0.08	0.06	0.06	0.09	0.08	0.04	0.12	0.12	0.12	0.12	0.06	0.05	0.06	0.14
15	0.11	0.03	0.07	0.05	0.07	0.04	0.05	0.09	0.07	0.03	0.11	0.11	0.11	0.11	0.05	0.03	0.05	0.15
16	0.11	0.05	0.08	0.06	0.08	0.06	0.06	0.09	0.08	0.04	0.12	0.12	0.12	0.12	0.06	0.05	0.06	0.14
17	0.12	0.15	0.15	0.14	0.15	0.15	0.15	0.14	0.15	0.15	0.14	0.14	0.14	0.14	0.14	0.15	0.14	0.16

Figure 5.5: Average cosine similarity between 20 non-random embeddings for ProtBERT. Values are the standard deviation.

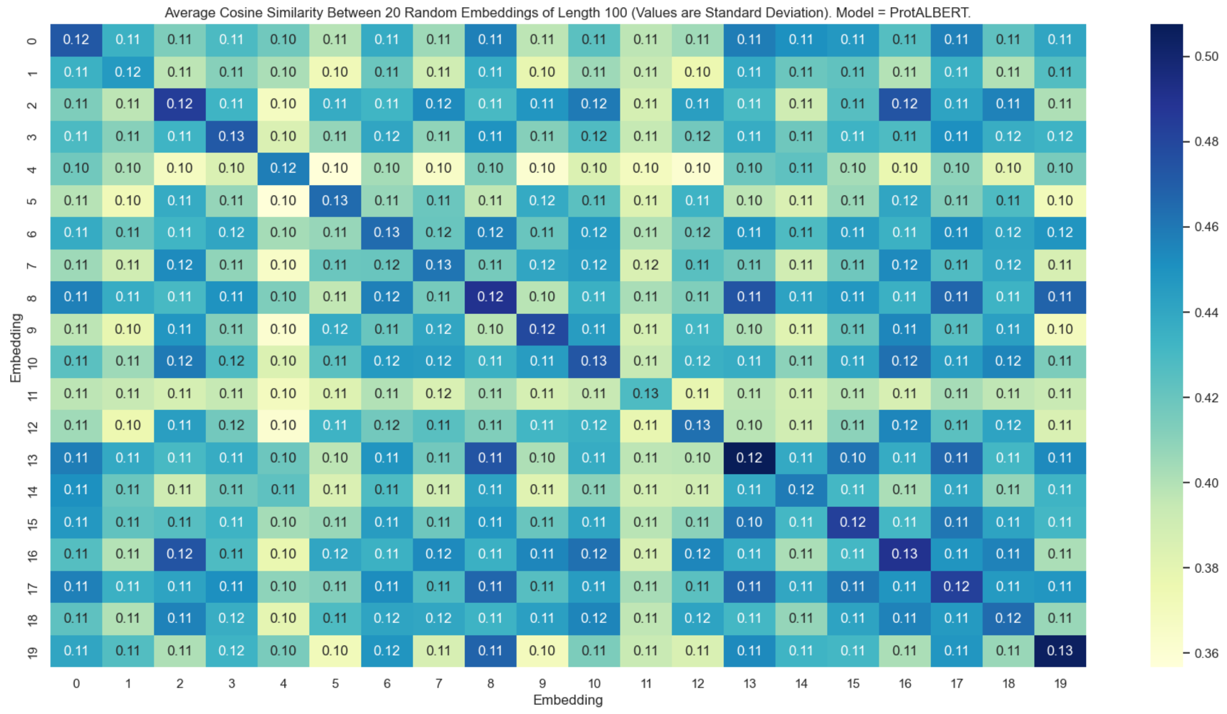


Figure 5.6: Average cosine similarity between 20 randomly generated embeddings for ProtALBERT. Values are the standard deviation.

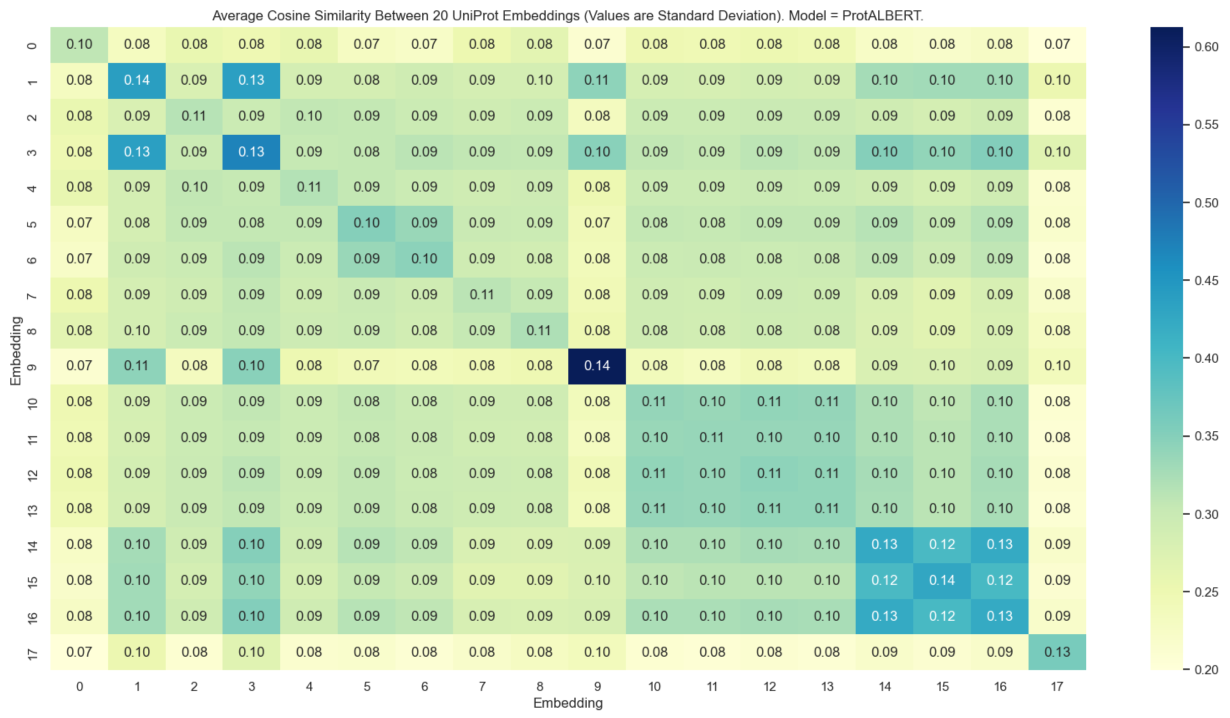


Figure 5.7: Average cosine similarity between 20 non-random embeddings for ProtALBERT. Values are the standard deviation.

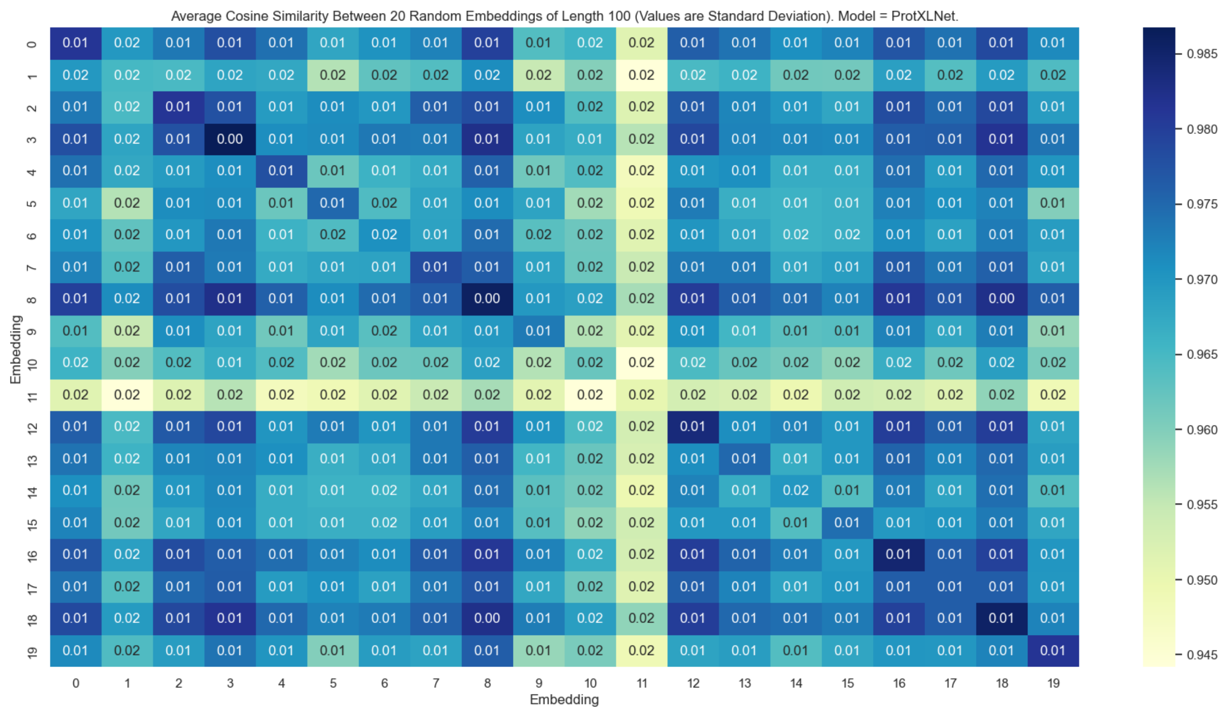


Figure 5.8: Average cosine similarity between 20 randomly generated embeddings for ProtXL-Net. Values are the standard deviation.

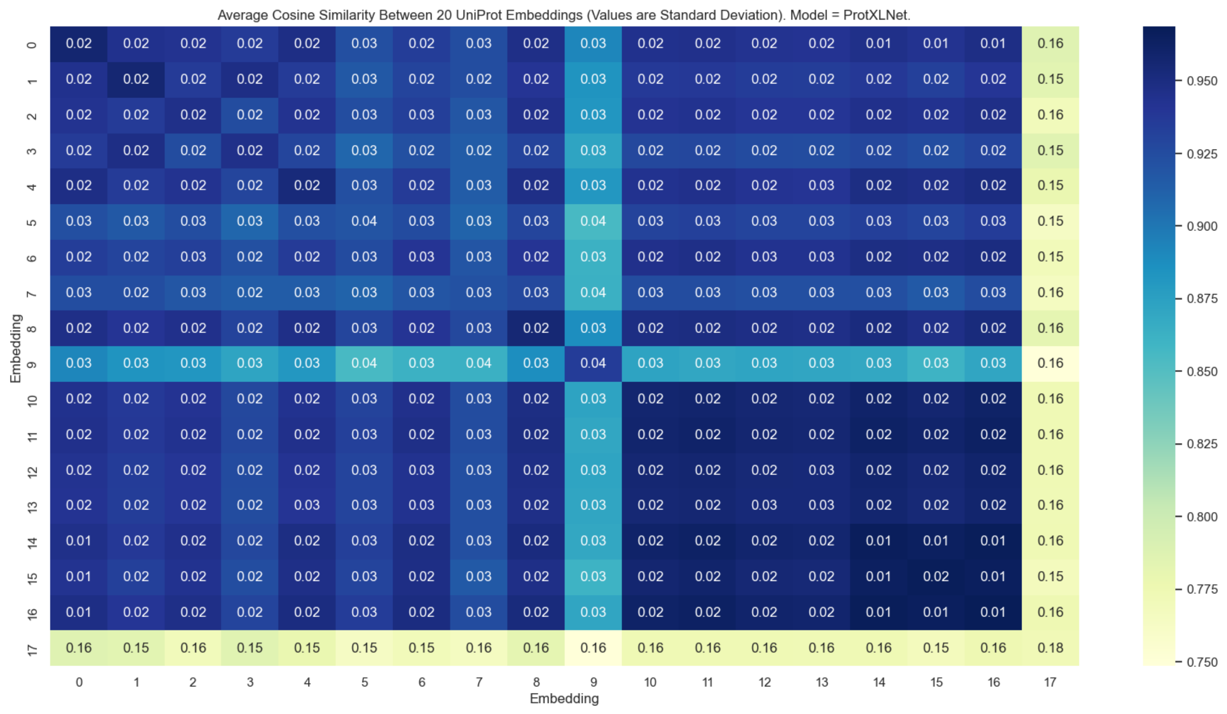


Figure 5.9: Average cosine similarity between 20 non-random embeddings for ProtXLNet. Values are the standard deviation.

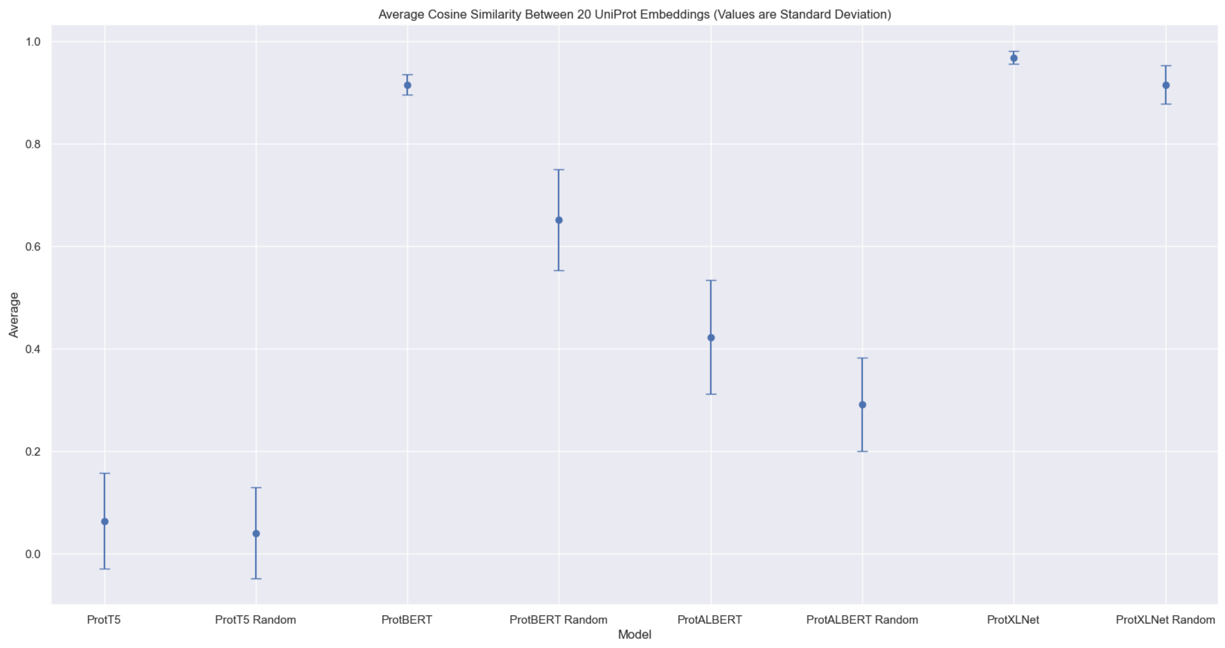


Figure 5.10: Average cosine similarity Between 20 Embeddings (Non-Random vs. Random) For Different Models.

Discussion

Conclusion

Future Work and Lessons Learned

Using the ProtTrans fine-tuning per-protein notebook as a basis to fine-tune the E-score method for the ProtT5 model may improve the performance of the method as a whole. This requires significant modifications to the fine-tuning process, as mentioned in the Discussion:

- Fine-tuning the model with the ProtT5 per-protein notebook as a basis, creating a LoRA adapter for the E-score method.
- Modifying the fine-tuning notebook to work on pairs of inputs as opposed to a singular input, with penalties being assigned based on how far the E-score alignment score for the pair of embeddings is from the labeled data.

Significant lessons learned from this research:

1. .

8.1 Acknowledgements

Dr. Lucian Ilie provided feedback on the thesis proposal and guidance throughout the process to conduct research following the original E-score paper Ashrafzadeh et al., 2023.

References

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*. https://doi.org/10.1016/S0022-2836(05)80360-2

Ashrafzadeh, S., Golding, G. B., Ilie, S., & Ilie, L. (2023). Scoring alignments by embedding vector similarity. https://doi.org/10.1101/2023.08.30.555602

Beals, M., Gross, L., & Harrell, S. (1999). Amino acid frequency. https://www.nimbios.org/~gross/bioed/webmodules/aminoacid.htm

Consortium, T. U. (2022). UniProt: the Universal Protein Knowledgebase in 2023. *Nucleic Acids Research*, *51*(D1), D523–D531. https://doi.org/10.1093/nar/gkac1052

De Lucrezia, D., Slanzi, D., Poli, I., Polticelli, F., & Minervini, G. (2012). Do natural proteins differ from random sequences polypeptides? natural vs. random proteins classification using an evolutionary neural network. *PLOS ONE*, *7*(5), 1–10. https://doi.org/10.1371/journal.pone.0036634

Desiere, F., Deutsch, E. W., King, N. L., Nesvizhskii, A. I., Mallick, P., Eng, J., Chen, S., Eddes, J., Loevenich, S. N., & Aebersold, R. (2006). The PeptideAtlas project. *Nucleic Acids Research*, *34*(suppl$_1$), D655–D658. https://doi.org/10.1093/nar/gkj040

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. https://doi.org/10.48550/arXiv.1810.04805

Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Yu, W., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., Bhowmik, D., & Rost, B. (2021). Prottrans: Towards cracking the language of lifes code through self-supervised deep learning and high performance computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. https://doi.org/10.1109/TPAMI.2021.3095381

Elnaggar, A., Heinzinger, M., Dallago, C., Rehawi, G., Wang, Y., Jones, L., Gibbs, T., Feher, T., Angerer, C., Steinegger, M., Bhowmik, D., & Rost, B. (2022). Prottrans: Toward understanding the language of life through self-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.*, *44*(10), 7112–7127. https://doi.org/10.1109/TPAMI.2021.3095381

Gligorijević, V., Renfrew, P. D., Kosciolek, T., Leman, J. K., Berenberg, D., Vatanen, T., Chandler, C., Taylor, B. C., Fisk, I. M., Vlamakis, H., Xavier, R. J., Knight, R., Cho, K., & Bonneau, R. (2021). Structure-based protein function prediction using graph convolutional networks. *Nature Communications*, *12*(1), 3168. https://doi.org/10.1038/s41467-021-23303-9

Henikoff, S., & Henikoff, J. G. (1992). Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*. https://doi.org/10.1073/pnas.89.22.10915

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., … Sifre, L. (2022). Training compute-optimal large language models.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2021). Lora: Low-rank adaptation of large language models.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., . . . Hassabis, D. (2021). Highly accurate protein structure prediction with alphafold. *Nature*, *596*(7873), 583–589. https://doi.org/10.1038/s41586-021-03819-2

Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural language processing: State of the art, current trends and challenges. *Multimedia Tools and Applications*, *82*(3), 3713–3744. https://doi.org/10.1007/s11042-022-13428-4

Kulmanov, M., & Hoehndorf, R. (2019). DeepGOPlus: improved protein function prediction from sequence. *Bioinformatics*, *36*(2), 422–429. https://doi.org/10.1093/bioinformatics/btz595

Lai, B., & Xu, J. (2021). Accurate protein function prediction via graph attention networks with predicted structure information. *bioRxiv*. https://doi.org/10.1101/2021.06.16.448727

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2020). Albert: A lite bert for self-supervised learning of language representations.

Lialin, V., Deshpande, V., & Rumshisky, A. (2023). Scaling down to scale up: A guide to parameter-efficient fine-tuning.

Lin, Z., Akin, H., Rao, R., Hie, B., Zhu, Z., Lu, W., Smetanin, N., dos Santos Costa, A., Fazel-Zarandi, M., Sercu, T., Candido, S., et al. (2022). Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. https://doi.org/10.48550/arXiv.1907.11692

Ma, Y., Liu, Y., & Cheng, J. (2018). Protein secondary structure prediction based on data partition and semi-random subspace method. *Scientific Reports*, *8*(1), 9856. https://doi.org/10.1038/s41598-018-28084-8

Marchler-Bauer, A., Derbyshire, M. K., Gonzales, N. R., Lu, S., Chitsaz, F., Geer, L. Y., Geer, R. ., He, J., Gwadz, M., Hurwitz, D. I., Lanczycki, C. J., Lu, F., Marchler, G. ., Song, H. S., Thanki, N., Wang, Z., Yamashita, R. A., Zhang, D., Zheng, C., & Bryant, S. H. (2015). Cdd: Ncbi's conserved domain database. https://doi.org/10.1093/nar/gku1221

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space.

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018). *Foundations of machine learning* (2nd ed.). MIT Press. (Original work published 2012)

Nevers, Y., Glover, N. M., Dessimoz, C., & Lecompte, O. (2023). Protein length distribution is remarkably uniform across the tree of life. *Genome Biology*, *24*(1), 135. https://doi.org/10.1186/s13059-023-02973-2

Ofer, D., Brandes, N., & Linial, M. (2021). The language of proteins: Nlp, machine learning & protein sequences. *Computational and Structural Biotechnology Journal*, *19*, 1750–1758. https://doi.org/https://doi.org/10.1016/j.csbj.2021.03.022

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. In A. Moschitti, B. Pang, & W. Daelemans (Eds.), *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (pp. 1532–1543). Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. https://doi.org/10.48550/arXiv.1910.10683

Rao, R., Meier, J., Sercu, T., Ovchinnikov, S., & Rives, A. (2020). Transformer protein language models are unsupervised structure learners. *bioRxiv*. https://doi.org/10.1101/2020.12.15.422761

Rives, A., Meier, J., Sercu, T., Goyal, S., Lin, Z., Liu, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., & Fergus, R. (2019). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *PNAS*. https://doi.org/10.1101/622803

Senior, A. W., Evans, R., Jumper, J., Kirkpatrick, J., Sifre, L., Green, T., Qin, C., Žídek, A., Nelson, A. W. R., Bridgland, A., Penedones, H., Petersen, S., Simonyan, K., Crossan, S., Kohli, P., Jones, D. T., Silver, D., Kavukcuoglu, K., & Hassabis, D. (2020). Improved protein structure prediction using potentials from deep learning. *Nature*, *577*(7792), 706–710. https://doi.org/10.1038/s41586-019-1923-7

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (Second). The MIT Press. https://inst.eecs.berkeley.edu/~cs188/sp20/assets/files/SuttonBartoIPRLBook2ndEd.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. https://doi.org/10.48550/arXiv.1706.03762

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019). Glue: A multi-task benchmark and analysis platform for natural language understanding.

Yang, J., Anishchenko, I., Park, H., Peng, Z., Ovchinnikov, S., & Baker, D. (2019). Improved protein structure prediction using predicted inter-residue orientations. *bioRxiv*. https://doi.org/10.1101/846279

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*. https://doi.org/10.48550/arXiv.1906.08237