Explaining embedding results for scoring alignments Thesis proposal (CS 4490Z)

Riley Gavigan
Thesis Supervisor: Lucian Ilie, Course Instructor: Nazim Madhavji
Department of Computer Science, University of Western Ontario, London, N6A 5B7, Ontario, Canada

November 27, 2023

Abstract

The proposed E-score alignment scoring method generates results that further research will explore, leading to insight for performance improvement. Cosine similarity results generated by the models were mostly positive instead of being randomly distributed. Exploration of this distribution will lead to a better understanding of the different models and the embedding vectors being produced. Different embedding models also differ greatly in performance with varying mean ranges. From this research, insight will be obtained to improve embedding performance and transformer model architecture.

1 Structured Abstract

Context and motivation

The *E*-score protein alignment scoring method [1] outperforms state-of-the-art methods, supported by comparing ProtT5 [2] *E*-score results with BLOSUM45 [3].

This research aims 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 [2, 4] and performance of embeddings.

Research questions

- What properties of embeddings produce better cosine similarity results?
- Why does ProtT5 produce the best embeddings?
- Why do cosine similarity results primarily fall within a positive range?
- How can the underlying 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 will provide insight into their distributions. Through these comparisons, conclusions about properties that improve *E*-score results can be 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.
Research resources:

- *E*-score analysis notebook
- ProtT5 fine-tuning notebook
- SageMaker Studio and Bioinformatics Lab compute power

Anticipated type of results

This study primarily aims 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
- Findings that can be reproduced for Natural Language Processing models and embeddings

Anticipated novelty

By building upon a novel method for scoring protein alignments using cosine similarity [1], anticipated results from this research will be novel conclusions that can further be explored and built upon.

Anticipated impact of results

Improvements in transformer models for the *E*-score alignment scoring method can be made through the

Table 1: *Transformer models available in the E-score method.*

Model	Embedding Dim	Pre-Trained On
ProtT5	1024	UniRef50
ProtBert	1024	UniRef100
ProtAlbert	4096	UniRef100
ProtXLNet	1024	UniRef100
ESM1b	1280	UniRef50
ESM2	1280	UniRef50

insight this research finds. Any anticipated improvements would also be applicable to Natural Language Processing Models, such as T5 [5].

2 Background and Related Work

Proteins and Alignments

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 BLAST program[1] is one of the most widely used tools in science [6]. An essential part of BLAST is the scoring function; the most widely used functions are provided by the BLOSUM matrices [3].

Protein transformers

The *E*-score alignment method depends on several protein transformer models (shown in Table 1) to generate embeddings for a protein sequence prior to calculating cosine similarity. These transformers include ProtT5, ProtBert, ProtAlbert, and ProtXLNet[2] from the ProtTrans project [2], as well as ESM1b and ESM2[3] from the Meta Fundamental AI Research Protein (FAIR) Team [4].

The ProtTrans models are based off their Natural Language Processing (NLP) equivalents, modified to understand the language of proteins. These models were pre-trained on protein databases, such as the UniProt Reference Clusters [7].

All Transformer models are derivations of the original Transformer model architecture [8]. For example, T5 is an encoder-decoder model (commonly referred to as a sequence-to-sequence model) that uses both the encoder and decoder of the Transformer architecture [5]. Contrasting this, the BERT models [9, 10] are auto-encoding models, using only the encoder from the original architecture.

Protein embeddings are similar to word embeddings in NLP and describe the position of a residue in a high-dimensional vector space. These embedding vectors are contextual because of the self-attention mechanism [8], meaning the embedding vector for a given residue differs depending on the context surrounding it.

ProtT5's average *E*-score performance was the most similar to the BLOSUM45 matrix, shown in Figure 2. BLOSUM45 is the best-performing old method for scoring alignments [1], making it a valuable comparison for the *E*-score models. ProtT5 serves as a strong benchmark to compare other models to in understanding embedding *E*-score performance.

Embedding vectors

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 dimension outlined in Table 1 determines the columns of the embedding vector. The dimensionality of these vectors represents the number of features encoded in the embedding, and is a fixed value for a given model.

E-score calculations

The embeddings produced by a model for a protein P, calculated in Equation 1, are used as the input to calculate the cosine similarity.

$$E(P) = GetEmbeddings(Model = ProtT5) \quad (1)$$

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.

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

E-score is calculated by taking the cosine similarity between the embedding vector for two residues (i, j), shown in Equation 3 where P_1 and P_2 are proteins [1].

$$E\text{-}score(i,j) = CosSim(E(P_1)_i, E(P_2)_j) \quad (3)$$

[1] Exceeds 108,000 citations, according to Google Scholar.
[2] https://github.com/agemagician/ProtTrans
[3] https://github.com/facebookresearch/esm

3 Research Goal and Objectives

Goal: Obtain significant insight about the *E*-score results and the differences between different models, leading to improvement of both the embedding performance and the embeddings being generated by the models.

Insight about performance differences between models has a significant impact outside of the *E*-score method. These models are used extensively for other purposes and insight that can lead to performance improvements is important to the research community.

Embedding vector distributions

By understanding the distributions of the embedding vector values, there will be a further understanding as to why the cosine similarity results are mostly positive (0..1) when calculating *E*-score, instead of being randomly distributed from (-1..1).

Because cosine similarity averages are mostly positive, there are factors within the embedding vectors contributing to the angles between the embedding vectors being mostly less than $\frac{\pi}{2}$.

Cosine similarity results

After having a stronger understanding of embedding distributions, the cosine similarity results will be explored by visualizing *E*-score dispersion. Testing different combinations of alignments and modifying parameters will aid in exploring cosine similarity.

By understanding cosine similarity distributions, conclusions can be drawn about contributing factors to the observed results.

Model performance differences

The results between different embedding models vary greatly in their average range of values and in comparison to BLOSUM [3] matrix results. Understanding why different embedding models generate different ranges in *E*-score values will serve as a foundation for drawing significant research implications.

With knowledge about the factors contributing to embedding performance, improvements can be made to the different models for scoring alignment.

```
>gi|464921|sp|P34981|TRFR_HUMAN
TILLVLIICGLGIVGNIMVVLVVMRTKHMRTPTNCYLVSLAVADLMVLV...
>gi|20455271|sp|Q9NSD7|R3R1_HUMAN
ISVVYWVVCALGLAGNLLVLYLMKSMQGWRKSSINLFVTNLALTDFQFV...
```

Figure 1: *Sample FASTA input file containing two sequences from the 7tm_GPCRs MSA.*

4 Research Approach and Methodology

Technical issues

Batch processing and analyzing a large dataset of FASTA files is a time-consuming, computationally-expensive process. Attention to ensure the mathematical basis and reasoning for analysis prior to running extensive notebook jobs is critical, and smaller sample calculations will be performed in advance to validate the anticipated results.

Data

Sequences will be obtained from sample multiple sequence alignments from the Conserved Domain Database [11] by NCBI. The reference alignments that will be used will be taken from the *E*-score paper's 49 curated multiple sequence alignments [1]. A sample FASTA file that will be used as input is shown in Figure 1.

Additionally, UniProt [12] peptide searches[4] will be performed to obtain FASTA data for embedding tests. Utilizing a significant number of sequences in FASTA format will strengthen findings between different models.

Embedding vectors and cosine similarity

With FASTA input files containing two protein sequences, the embedding vectors and cosine similarity will both be extracted independently for analysis and visualization. By using the *E*-score notebook as a starting point, modifications and new methods will be created to extract this information for a provided FASTA file and model.

Defined in the *E*-score analysis[5] repository are methods to retrieve this data, such as *get_embedding*().

[4] https://www.uniprot.org/peptide-search
[5] https://github.com/rgavigan/e-score

Data analysis

Analysis of embedding vectors and cosine similarity to generate useful visualizations and perform statistical analysis on results will be done in the *E*-score analysis notebook. The following tools will be used:

- Seaborn [13] and Matplotlib [14] for data visualization
- SciPy [15] and NumPy [16] for statistical significance testing
- Scikit-learn [17] for normalization

Embeddings

Normalizing the embeddings generated between models to have a consistent scale and distribution is required prior to analysis. Common techniques for normalization to consider include min-max scaling, Z-Score standardization, and L2 normalization.

The distribution of normalized values can be visualized in multiple ways including histograms, kernel density plots, or box plots. Summary statistics (mean, median, deviation, range) may provide additional insight.

Correlations and differences between the embeddings can be visualized on a per-residue basis with heatmaps.

Statistical tests are required to validate and assess the significance of the visualized results. ANOVA, t-tests, or other means can be used to assess significance.

E-score

To gain a deeper understanding of the *E*-score results generated for the different embeddings as shown in Figure 2, dispersion will also be visualized. By visualizing the standard deviation and range of the results for each embedding type, there will be more insight to draw conclusions from. This will help understand why cosine similarity mean results are mostly positive (0..1), and why different models generate broader or narrower average ranges.

Modifying parameters

By modifying different parameters and generating *E*-scores for different combinations from insight gained by comparing different transformer models, further understanding of the primary factors contributing to performance will provide insight to improve embeddings. Examples of parameters that can be modified are alignment type, gap penalty, and gap extension penalty.

Interpretation

Through the visualizations and statistical analysis performed on embedding vectors and cosine similarity data, comparing properties that are present in stronger models (i.e. ProtT5) with weaker models will be a focus for interpreting the relevance of the analysis.

By benchmarking the models against one another and extracting features present in the stronger performing models, insight about why performance differences are present will be obtained and used to explain performance improvement approaches.

Type of results

Visualizations, statistical comparisons/tables, and inference using empirical findings to answer the research questions will be the anticipated results from this study.

Challenges or threats

Interpreting the meaning of embeddings is a challenging process because embeddings lack direct human interpretability due to their large dimensionality. Conducting appropriate statistical tests to validate findings and visualizing results to be human-readable is crucial.

Representative data for input sequences is necessary to minimize the chance of biased results.

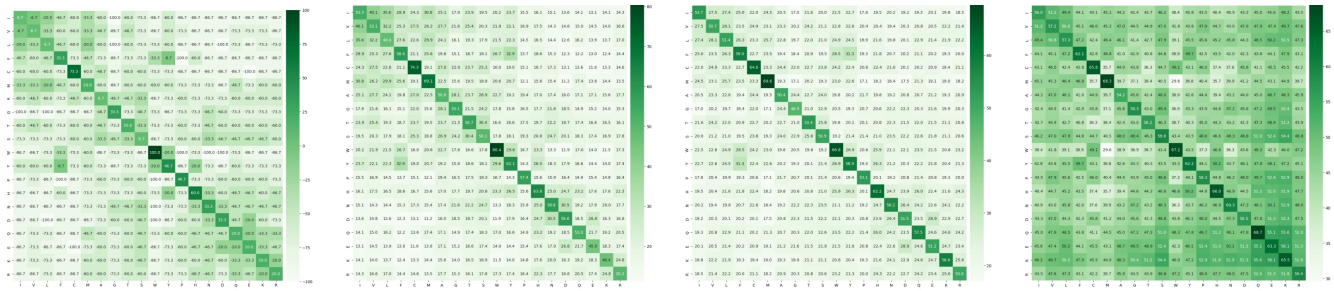
5 Novelty and Impact

The anticipated result from this research is thorough insight into the *E*-score method's results. These data-supported findings are expected to contribute significantly to the understanding and improvement of embeddings and the models that generate them.

There are three primary advancements that can be explored from the implications drawn from the results. These would serve as beneficial follow-up research to improve embedding performance, transformer models, and to apply the same advancements to Natural Language Processing.

Improving embedding performance

From the novel insight gained about the properties that contribute to the performance of different protein embeddings, the performance of the embeddings can be improved for sequence alignment. This would be an application of the data-supported explanation for why different embeddings have different cosine similarity averages and ranges.



(a) (b) (c) (d)

Figure 2: *Heatmaps of (a) BLOSUM45 matrix (scaled to -1..1) and three aligned matrices of average E-scores for the NBD_sugar-kinase_HSP70_actin MSA: (b) ProtT5-score, (c) ProtAlbert-score, and (d) ProtXLNet-score [1].*

Transformer model improvement

The above insights can be used as a starting point to modify the architecture of the different transformer [8] models such as the ProtTrans models ProtT5, ProtBert, ProtXLNet, and ProtAlbert [2]. This research can be extended to improve these models through different processes such as hyperparameter tuning and optimization.

The ProtT5 model, the best performing model from the ProtTrans project, would serve as a valuable model to fine-tune with the research implications drawn from this research. The ProtTrans GitHub repository[6] provides Jupyter Notebooks for fine-tuning this model for per-protein prediction, per-residue classification, and per-residue regression.

Natural Language Processing

The ProtTrans models available for the *E*-score method are all derivations of Natural Language Processing models modified to work with the language of proteins, the 20 amino acids.

All of the above research (improving embedding performance and transformer models) can be repeated for Natural Language Processing contextual embeddings such as ELMo [18], BERT [9], RoBERTa [10], XLNet [19], and T5 [5].

References

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

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

[7] B. E. Suzek et al. "UniRef: comprehensive and non-redundant UniProt reference clusters". In: *Bioinformatics* 23.10 (Mar. 2007), pp. 1282–1288. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btm098. URL: https://doi.org/10.1093/bioinformatics/btm098.

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

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

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

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

[11] A. Marchler-Bauer et al. "CDD: NCBI's conserved domain database". In: (2015). DOI: 10.1093/nar/gku1221.

[12] 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. URL: https://doi.org/10.1093/nar/gkac1052.

[13] M. L. Waskom. "seaborn: statistical data visualization". In: *Journal of Open Source Software* 6.60 (2021), p. 3021. DOI: 10.21105/joss.03021. URL: https://doi.org/10.21105/joss.03021.

[14] 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.

[15] 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.

[16] 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. URL: https://doi.org/10.1038/s41586-020-2649-2.

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

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

[19] 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.