Res2Vec: Amino acid vector embeddings from 3d-protein structure

Scott Longwell [1] **Tyler Shimko** [2]

Abstract

The 20 naturally-occurring amino acids are the building blocks of all proteins. These residues confer distinct physicochemical properties to proteins based on features like atomic composition, size, and charge. Here, inspired by recent work in NLP, we create vector embeddings of each amino acid based on their contexts of neighboring residues within folded proteins. We then test the utility of these embeddings in a data-scarce supervised task, classifying amino acid mutations as "neutral" or "destabilizing" to T4 lysozyme.

1. Introduction

One of the greatest open challenges in medicine is to accurately predict a protein's function (or dysfunction) from the primary sequence of amino acids that comprise it. The ubiquity of full genome sequencing has made this challenge all the more salient, as we now possess an abundance of genomic and, therefore, protein sequence data, but are unable to experimentally assess how the thousands of mutations that are routinely uncovered manifest as changes to a protein's function. We lack this ability because a) experimental characterization of a single protein is costly and time consuming and b) the space of protein sequences is unfathomably vast. Together, these two factors conspire to yield a sparsely sampled sequence-function landscape.

The high cost of experiments to determine protein function from structure makes predictive algorithms an attractive alternative. While supervised deep learning algorithms perform well over complex, high-dimensional landscapes, they also rely on ample labeled training examples. Until recently, the field of natural language processing (NLP) was stymied by the same quandary computational biology now faces - a preponderance of unlabeled sequences, with little way to generalize across small sets of labeled data. A breakthrough insight in NLP was that powerful models could be trained on smaller datasets providing their inputs - words - were featurized such that words with similar meanings had similar vector representations. To capture the meaning of a word, the idea of *distributional semantics* - that "you shall know a word by the company it keeps" - played a critical role. Leveraging the fact that words with similar meanings tend to be observed in similar contexts (e.g. sentences, documents), several models have been created to learn meaningful word embeddings from unlabeled text, most notably the watershed word2vec model (1). These word embeddings increasing performance on supervised tasks such as sentiment analysis and document classification because they effectively augment small datasets with information learned from an enormous text corpus.

Taking lessons of NLP into account, the obvious analogy for computational biology is to learn vector representations of amino acids that distill biochemical meaning from their context in proteins. Here, we adapt the word2vec algorithm to accept a 3D "context bubble" of neighboring amino acids surrounding a target amino acid as *input* and predict the target amino acid identity as *output*. The process of training this model, which we call *Res2Vec* or *r2v*, generates vector representations for each of the 20 naturally-occurring amino acid residues, the utility of which we validate on a protein mutation effect prediction task.

2. Related Work

Unsupervised methods to generate vector embeddings of words, such as word2vec (1) and GloVe (2) have underpinned rapid advancement in (NLP). At their core, these methods leverage the fact that words with similar *semantic meanings* tend to appear in similar contexts (e.g. sentences, documents). The analogous problem in the fields of biochemistry and structural biology is to encode the *biochemical meaning* of an amino acid based on the contexts (e.g. domains, proteins) in which it is frequently situated. Prior attempts have used this principle to create amino acid embeddings (3) based on abundant 1D primary protein sequences. While these approaches yield useful embeddings which improve performance on supervised tasks relative to one-hot amino acid encodings, they disregard information about the 3D structural context of each residue.

3. Datasets and Features

The ability to predict a protein's 3D structure - or fold - from its primary sequence of amino acids has been a longstanding challenge in the field of computational biology. To provide an unbiased benchmark for computational models

addressing this challenge, the *Critical Assessment of Protein Structure Prediction* (CASP) competition provides biennial releases of unpublished protein structures to guarantee that computational models can be evaluated against truly unseen data (4; 5). These releases have been further curated and extended by the AlQuraishi laboratory to create standardized, rationally chosen training/validation/test datasets for problems relating to protein structure prediction. In combination, these resources amount to an equivalent of the ImageNet dataset, appropriately named ProteinNet, for protein structural data (https://github.com/aqlaboratory/proteinnet). Table 3 displays the number of training example proteins within each of the ProteinNet data splits.

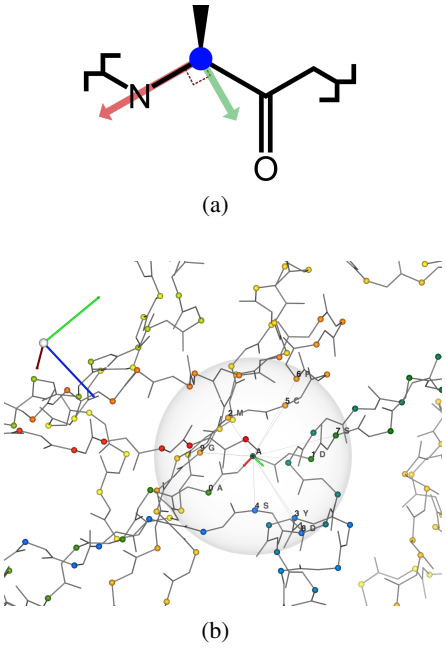
SEQ. SIM. THRESHOLD	PROTEINS
30%	22,344
50%	29,936
70%	36,005
90%	42,507
95%	43,544
100%	87,573
VAL	224
TEST	78

Table 1. ProteinNet dataset. The number of training example proteins is shown for each threshold of sequence similarity to the proteins withheld in the CASP11 test dataset. The number of examples in the validation and test sets are also shown. Note that the number of amino acids per protein is on the order of 100s.

To create our embeddings, we employed the ProteinNet datasets with the predefined training/validation/test splits as both sequence and 3D structural information are available for each protein in the database. The positions for each atom/residue in each PDB file (a type of protein structure file) are given in 3D space using a Cartesian coordinate system. However, there exists no universal standard for the orientation of protein structures within the coordinate system of these files. We therefore had to develop a reoriented coordinate system within which each context window could be examined while maintaining a consistent notion of orientation.

3.1. A target-centric coordinate system

The 3D spatial arrangement of a specific amino acid's neighbors defines the physicochemical environment in which the target amino acid resides. Therefore, not only distance, but also the orientation, of the neighboring residues is important to understand a target residue's context. To provide an embedding model with information about the 3D context, we devised a change of basis system such that the α carbon of the target amino acid is at the origin with the three basis vectors positioned with respect to the nitrogen and carbonyl carbon of the target residue (Fig. 1(a)).



(a)

(b)

Figure 1. Definition of a target residue's 3D context. (a) To give each target residue a consistent notion of direction, we defined orthogonal (X, Y, Z) axes, shown as (R, G, B) arrows, to elicit a change of coordinate basis. (b) Example target residue (alanine) with context of 10 nearest amino acids by αC-αC distance. Sidechains are hidden; the α carbons of each residue are shown as small spheres with a rainbow color-mapping from N to C terminus. Note the structure's original basis (top-left) and redefined target-centric unit-vectors (center).

For each target amino acid, we first subtracted the Cartesian coordinates of the target α carbon from every atom in the structure. We then defined the z-axis by taking the cross-product of the α carbon-nitrogen and α carbon-carbonyl carbon coordinate vectors to yield a vector orthogonal to the plane. We then normalized this vector to unit length and repeated this operation to yield the x and y basis vectors respectively. We then inverted this matrix to change the coordinate system of the entire protein structure such that the α carbon of the target residue was at the center and the three basis vectors of the coordinate system were orthogonal and consistently positioned. These adjusted coordinates for the α carbon of each context residue were then concatenated with a one-hot encoding vector indicating the identity of each context residue to yield the input to our embedding model.

4. Methods

Notation example: matrix $\mathbf{M}$; vector $\mathbf{m}$; scalar dimension M; index m.

Assume a set of S one-hot encoded symbols $\mathbf{S} \in \{0,1\}^{\mathrm{S}\times\mathrm{S}}$

(e.g. words; amino acids).

4.1. word2vec architecture choice

The word2vec algorithm has two flavors that generate dense vector representations of a symbol: skip-gram (SG) and continuous-bag-of-words (CBOW). Given a corpus of sequences (*e.g.* database of text documents or proteins sequences), either model will read through the sequences, at each step considering a context window of symbols $\mathbf{C} \in \{0,1\}^{\mathrm{C}\times\mathrm{S}}$ around a target symbol **t**. However, SG and CBOW have different (reciprocal) tasks: SG takes **t** as input and tries to predict **C** as output; CBOW takes **C** as input and tries to predict **t** as output. Canonically, SG performs better on infrequent symbols, while CBOW is faster to train (1). Both models can generate useful embeddings, but the CBOW training task - predicting amino acid probability from context - is likely to capture more contextual information pertinent to the task of mutation effect prediction, so we selected it for further development.

4.2. Generation of weighted context vector x

CBOW must first "summarize" the C one-hot vectors in **C** to create an input vector $\mathbf{x} \in \mathbb{R}^{\mathrm{S}}$. Traditionally, CBOW accomplishes this through simple averaging:

$$\mathbf{x} = \frac{1}{\mathrm{C}} \sum_{\mathrm{c}}^{\mathrm{C}} \mathbf{s}_c$$

This approach removes any information concerning the position (relative or absolute) of the context symbols. While this may be acceptable for a small window over a 1D sequence, we desired to provide our model with some notion of 3D position of the context residues. We first decided to calculate the inner product of the XYZ-coordinates of each context residue's alpha-carbon ($\mathbf{D} \in \mathbb{R}^{\mathrm{C}\times 3}$) and a 3D-vector of trainable parameters ($\mathbf{w}^{[0]} \in \mathbb{R}^3$), then pass the result through a sigmoid activation. We reasoned that the resulting vector $\mathbf{r} \in \mathbb{R}^{\mathrm{C}}$ should correspond to a learned pseudo-distance, which is then used to perform a weighted average of the context vectors to arrive at **x**:

$$\begin{aligned} \mathbf{r} &= \sigma(\mathbf{D}\mathbf{w}^{[0]}) \\ \mathbf{x} &= \mathbf{C}^T\mathbf{r} \end{aligned}$$

We compared this sigmoid weighting to an alternative weighting scheme via 2 fully connected neural network layers, parameterized by weights $\mathbf{W}^{[0]} \in \mathbb{R}^{3\times 3}$ and $\mathbf{W}^{[1]} \in \mathbb{R}^{3\times 1}$, with ReLU activation over the coordinate matrix **D** such that:

$$\begin{aligned} \mathbf{r} &= \sigma(\sigma(\mathbf{D}\mathbf{W}^{[0]})\mathbf{W}^{[1]}) \\ \mathbf{x} &= \mathbf{C}^T\mathbf{r} \end{aligned}$$

where σ is the ReLU function.

Finally, we also implemented an inverse distance weighting scheme where **x** was calculated as

$$\begin{aligned} \mathbf{r} &= (1+\mathbf{d}^*)^{-1} \\ \mathbf{x} &= \mathbf{C}^T\mathbf{r} \end{aligned}$$

where $\mathbf{d}^*$ is the Euclidean distance to the α carbon of the context residue. During training time we compared the performance of all four of the above weighting schemes and ultimately selected the fully-connected neural network weighting.

4.3. CBOW model

Given an input vector **x**, the CBOW model is similar to a softmax classifier, consisting of a linear hidden layer followed by a softmax layer.

It first multiplies **x** by weight matrix $\mathbf{W}^{[1]}$ to realize a hidden layer $\mathbf{z}^{[1]}$ (no activation is applied here). A second network layer is then applied with softmax, along with a negative log-likelihood loss (NLL):

$$\begin{aligned} \mathbf{z}^{[1]} &= \mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]} \\ \hat{\mathbf{y}} &= \mathbf{a}^{[2]} = \sigma^{[2]}(\mathbf{W}^{[2]}\mathbf{z}^{[1]} + \mathbf{b}^{[2]}) \\ J(\mathbf{y},\hat{\mathbf{y}}) &= -(\mathbf{y}^T \ln \hat{\mathbf{y}}) \end{aligned}$$

where:

$$\sigma^{[2]}(\mathbf{z}) = \frac{1}{\mathbf{1}^T e^{\mathbf{z}}} e^{\mathbf{z}}$$

5. Experiments

To selected a suitable model to generate the vector embeddings, we performed sweeps across the hidden layer size and learning rate hyperparameters of the standard word2vec-style model. We tested hidden layer sizes ranging from 10 up to 1,000 hidden units and learning rate values ranging from 1 to 0.0001. We found little difference in performance, as measured by validation set loss, when varying the hidden layer size. To make direct comparisons between our embedding vectors and standard one-hot encoding and BLOSUM (6) empirical substitution vectors, we ultimately decided on a hidden layer size of 20 (Fig. 2).

In contrast to hidden layer size, we found learning rate to have a large impact on model performance, especially relating to convergence. We found a learning rate of 1 to

be too aggressive, with models failing to converge, and learning rates at or below 0.01 to be too relaxed, leading to delayed convergence. We used a learning rate of 0.1 to train all future models.

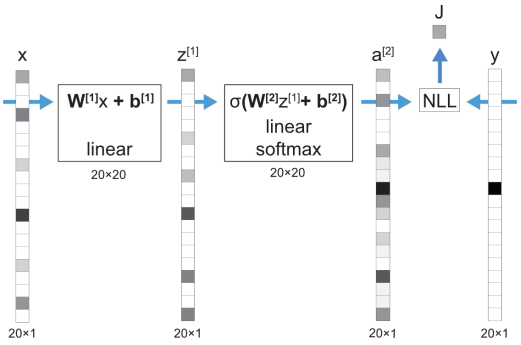
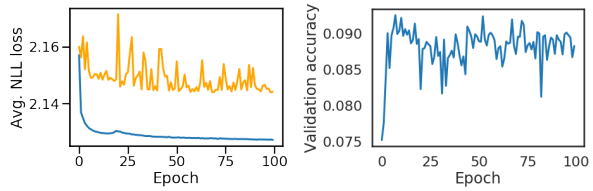


Figure 2. Core CBOW architecture employed by the r2v model.

To further boost model performance following selection of the hidden layer size and learning rate, we tested different methods of weighting the context residues surrounding the target residue. As noted in the Methods section, we first tried the weighted, sigmoid-activated coordinate system. We then also employed a pure average, the inverse distance metric, and a fully connected weighting system over the coordinates (all described in detail in the Methods section above). We found that the fully-connected weighting offered the best performance and selected this method for the final model.

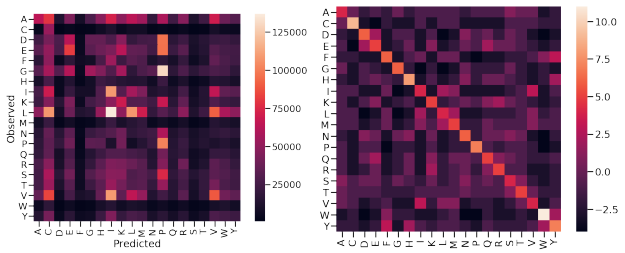


(a) (b)

Figure 3. Loss and accuracy metrics over training epochs. The average negative log likelihood loss value is shown for the training (blue) and validation (orange) datasets are shown in (a). The accuracy of the model on the validation dataset is shown in (b).

The final model was trained using the negative log likelihood loss function and all code employed the PyTorch library (7). Training and validation dataset losses for this model decreased to final values of roughly 2.12 and 2.14, respectively (Fig. 3(a)).

For the final model, prediction accuracy on the validation dataset increased, eventually leveling off around 8.8% (Fig. 3(b)) Despite the low accuracy of the model, closer inspection of the confusion matrix over the training dataset (Fig. 4(a)) indicates common and expected mistakes made by the model. For instance, leucine and isoleucine tend to be mistaken with high frequency. From a biochemical perspective, this is to be expected as these two residues are isomers of each other. Therefore, these two residues are similar in size and chemical properties, giving rise to a high level of confusion for the model. Comparison to the BLOSUM matrix (Fig. 4(a)) (6) indicates that this substitution is generally well-tolerated and, therefore, expected.



(a) r2v confusion matrix (b) BLOSUM score matrix

Figure 4. The confusion matrix for the model on the training dataset is shown in (a). For comparison, the scores present the BLOSUM62 empirical substitution matrix are shown in (b). The BLOSUM baseline indicates that some level of confusion is expected based on empirical tolerated substitution frequencies.

6. Results/Discussion

The final set of weights before the softmax layer of the model provide the final embedding vectors for each amino acids and can be interpreted as carrying the biochemical "meaning" of each individual residue. To gauge whether these vectors were encoding biochemically meaningful information, we performed a principal components analysis of the matrix of embeddings. We note that the first two principal components seem to cluster the residues by important biochemical properties (Fig. 5), such as charge and size, confirming the potential utility of these vectors.

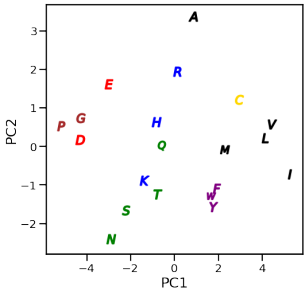


Figure 5. The first 2 principal components of the embedding vectors for each residue are shown (44.5% and 9.4% variance explained, respectively). Residues are identified by single letter code and colored by chemical property (blue: polar positive, red: polar negative, green: polar neutral, black: non-polar aliphatic, purple: non-polar aromatic, yellow: cysteine, brown: other).

The purpose of generating vector representations from abundant data is to transfer generalizable features to label-scarce supervised tasks, enabling simple models to still make effective predictions. To validate the utility of our vector representations, we considered a small supervised task established by Torng and Altman (8): given 1 of 40 amino acid mutations to T4 lysozyme characterized in literature, classify the functional effect as "neutral" or "destabilizing" (8). For comparison, we considered 5 different approaches to featurize each example's wild type and mutant amino acid with vectors (which are concatenated to form the input):

- one-hot: zero-vector except at the index corresponding to the amino acid, which is given a value of 1
- BLOSUM62: corresponding row of the score matrix
- r2v-v_freq: row of symmetric pseudo-score matrix generated from r2v confusion matrix (calculated by S_{freq} as described in (8))
- r2v-v_dot: similar to above, but calculated with S_{dot}
- r2v-W2: corresponding row of the second layer weights in Fig. 2 (*i.e.* the r2v embedded vectors)

All vectorizations are of length 20. We had hoped to make a comparision to the 3D-CNN featurization presented in Torng and Altman, but their substitution matrices were not directly accessible from the manuscript or supplement.

Following the example of Torng and Altman, we performed 4-fold cross validation on the dataset and evaluated the performance of support vector classifiers (SVC) with radial basis functions (RBF), implemented by scikit-learn. For each featurization, we performed a grid search over two arguments: C, for which smaller values regularize the classifier by trading training accuracy for margin of separation, and γ, for which smaller values increase the influence radius of the support vectors.

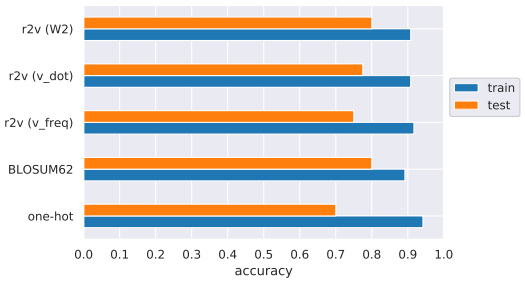


Figure 6. Comparison of mean accuracies of different input featurizations on T4 lyzozyme mutant classifcation task.

For a given 4-fold split, we refer to the held out fold as the "test" set. The mean training and test accuracies of the SVC-RBF model which yielded the highest mean test accuracy for each featurization are summarized in Fig. 6. The confusion-matrix derived vectors (r2v- v_freq and v_dot) did better than one-hot encoding, demonstrating that the confusion of r2v helps a SVC understand which mutations are tolerable. However, it is important to note that the confusion matrix of a perfect embedding model will be a diagonal matrix, effectively recapitulating a weighted-one hot encoding. Thus, the generation of useful confusion-matrix representations is at odds with embedding model training tasks focused on maximizing accuracy. On the other hand, embedding model weights must extract useful features if the embedding model is to have high accuracy. Indeed, the r2v-W2 featurization gave a 10% increase in mean test accuracy over one-hot encoding, identical to the performance of the BLOSUM vectors.

7. Conclusion and Future Work

The current performance of the model, at roughly 9% accuracy is lower than might be expected. To address this issue of model accuracy, we would change the way that context information is encoded and fed to the model. Primarily, we would reconsider the way in which the 10 closest residues are calculated. Currently, the distance between residues is calculated as the distance between the α carbons of each residue. However, given the great range of sizes for the side chains of the amino acids, we would suggest changing the distance calculation to reflect the distance between the two closest atoms of the residue side chains. This strategy would likely improve implicit encoding of spatial information and may have dramatic effects on contexts crowded with large amino acids residues. A second change that may improve context representation is encoding of the directionality of the α carbon-β carbon bond. Including this information would provide the model with the general directionality of the amino acid side chain (*e.g.* whether the chain is pointed toward or away from the target residue) that may be helpful in predicting the target residue.

Despite the relatively low accuracy of the model, the embedding vectors proved significantly better than residue identities alone at predicting mutational effects. The performance of the embedding vectors matched that of the BLOSUM62 (6) matrix, indicating that they may be helpful and informative for more intense mutation effect prediction tasks. Specifically, given more time, we would like to compare the performance of the one-hot, BLOSUM, and r2v encoding on a quantitative (continuous) effect prediction task. Such a task may be more sensistive to the implicit encoding of the r2v vectors, and it might be possible to distinguish performance of the r2v and BLOSUM vectors.

8. Contributions

We note that the authors contributed equally to this work. Both authors conceived of and designed the initial implementation of this model. Both authors contributed substantially to the codebase necessary to download, process, clean, and store the data as well as to define, train, and run the models and apply them. Finally, both authors contributed to the drafting and revising of this report and the associated project poster.

References

[1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv.org*, January 2013.

[2] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Stroudsburg, PA, USA, 2014. Association for Computational Linguistics.

[3] Ehsaneddin Asgari and Mohammad R K Mofrad. ProtVec: A Continuous Distributed Representation of Biological Sequences. *arXiv.org*, page e0141287, March 2015.

[4] Lisa N Kinch, Wenlin Li, R Dustin Schaeffer, Roland L Dunbrack, Bohdan Monastyrskyy, Andriy Kryshtafovych, and Nick V Grishin. CASP 11 target classification. *Proteins*, 84 Suppl 1:20–33, September 2016.

[5] Helen M Berman, John Westbrook, Zukang Feng, Gary Gilliland, T N Bhat, Helge Weissig, Ilya N Shindyalov, and Philip E Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, January 2000.

[6] S Henikoff and J G Henikoff. Amino acid substitution matrices from protein blocks. *Proceedings of the National Academy of Sciences*, 89(22):10915–10919, November 1992.

[7] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[8] Wen Torng and Russ B Altman. 3D deep convolutional neural networks for amino acid environment similarity analysis. *BMC bioinformatics*, 18(1):302, December 2017.