Lost in Translation: How the Math Behind Language Models is Revolutionizing Biology

Sydney Edwards

1.1. Introduction

The cells inside the body are decision makers —everything they do must be perfectly calibrated, perfectly timed, and in perfect coordination with the cells around them. To carry out their tasks, cells use proteins, building blocks of the cell that can also serve as signals to coordinate with other cells. This kind of coordination, as could be expected for any smooth operation, requires language. Language that says: make this protein at this time to this amount, oh and don't forget to add a little extra leucine[1] this time. We call the language which determines the interactions inside a cell DNA. Much like the languages we're used to, this DNA is prone to typos, grammatical errors, and complete nonsense[2], but more on why this analogy makes sense later [10].

For some added context, it's helpful to know how a biology lab operates. A typical biology lab usually explores answers to the natural world by a familiar process: proof by contradiction. We wish to prove a gene is important for a particular process. So, assume no gene —delete it from the DNA. Then, show that the process has suddenly gone haywire. But we assumed that this gene was not important for the process, a contradiction. Thus, our gene is central to the process we wanted to initially investigate. In this whole schema, DNA is merely a means to an end —play with it just enough so that the desired assumption can be proven. But DNA is so much more than a means to an end —it is the language underpinning biology that makes life happen. Recent work in the field of theoretical biology has begun to use concepts from linguistics, computer science, and mathematics to interrogate DNA's properties as a language. Treating DNA as a language provides insight into how cells operate and coordinate with one another and how to produce novel proteins that haven't yet popped up in the evolutionary timescale. In this paper, we'll examine the history of the linguistics of DNA and the mathematics behind recent advances in the field. Ultimately we wish to demonstrate why examining DNA as a language is useful, how this is done, and what the next steps are for this field.

[1] Leucine is a common amino acid that is used to build proteins

[2] For further reading on complete nonsense found within the English language, see https://twitter.com/

1.2. Everything You Forgot From Bio 1

Information inside a cell always follows the same flow: DNA becomes amino acids which become proteins. We call this the central dogma of biology. The important piece is that the final proteins are 3D. The amino acid sequence is known as the primary structure. It is just a string of characters that biologists call a peptide. Different amino acids in this peptide interact differently with each other. This results in local patterns and folds within the protein. We call this the secondary structure. A secondary structure is just a description of a local topology that arises because certain amino acids play nice and like to fold with other amino acids. Changes in this secondary structure result in global changes to the whole protein which we conveniently call the tertiary structure. Once the secondary structure is known, the local topology fixes the tertiary structure and it becomes easier to predict the final protein [12]. For a visual of this process, see Figure 1.1.

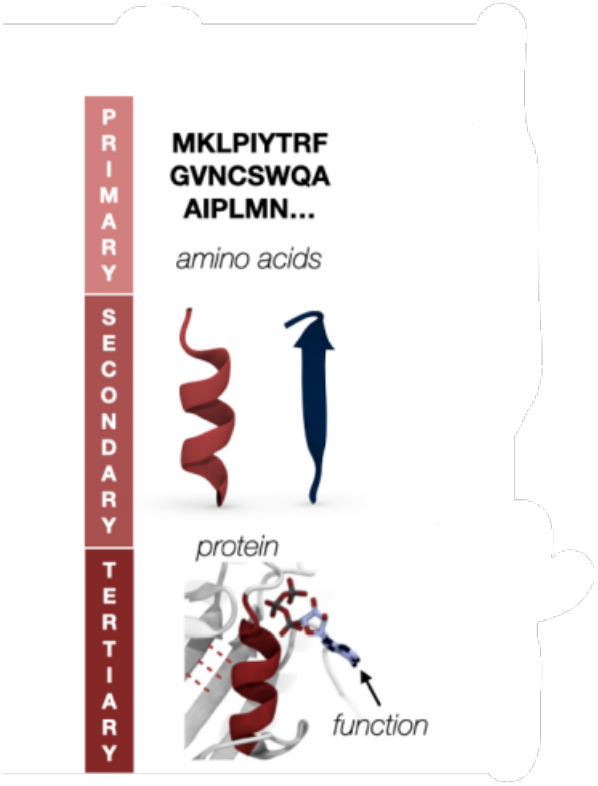


Figure 1.1. The flow of information from primary to secondary to tertiary structure of peptide sequences. Figure taken from [10].

To sum up: peptide sequences are strings of amino acids (primary structure) that fold (secondary structure) with themselves to create proteins (tertiary structure). Why does this concept matter? The amino acid sequence provides clues to the final protein structure and, due to advances in sequencing, is relatively inexpensive to find. Thus, there are rapidly expanding databases containing these sequences but not the final protein

structure. Enter computational biologists who like working with big data. Protein structure prediction provides a unique challenge in the amount of unlabeled data in the training set —we simply do not know what many proteins do or what many proteins look like [7]. Thus, it becomes difficult to always know whether our models are working, especially for the proteins that are not yet characterized. For a pithy explanation of why the problem of protein structure prediction is challenging, see Figure 1.2.

Figure 1.2. Topologically determining protein structure is difficult. By formulating protein folding first as a language problem and then attempting to solve the topology of a protein, we eliminate some of the headache involved in fully characterizing all topological properties of proteins. This concept is known as transfer learning. Comic taken from xkcd [20].

One overarching principle within biology is that form follows function. If the structure or form of a molecule, cell, or organ is known, its function can be readily inferred. For instance, certain proteins are hydrophobic or water-hating while others are hydrophilic or water-loving, which can inform where in the cell they are found. Thus, theoretically, understanding the structure of a protein provides a better understanding of how that protein works. In a practical sense, this means that how a protein works to cause a disease or how a medication might target a protein in the body could be predicted. Additionally, understanding protein structure helps to solve one of the major problems of modern medicine —side effects. Even after medications are approved, new, and possibly dangerous, side effects are still reported. Almost one-third of drugs approved between 2001 and 2010 caused unexpected side effects [1]. With knowledge of a protein's structure, we can determine whether a medication will interact with proteins of interest, including proteins in the body that we don't want to target which could lead to nasty side effects [18]. Protein structure prediction ultimately helps us to better understand what goes on in the body to inform future research and possible new therapies.

1.3. Proteins, Language, and Math, Oh My

The alphabet of proteins consists of the characters within a peptide chain —all 20 amino acids. For an n-length peptide, there are thus 20^n possible peptides that can be made. Similarly, in the English language, for a k-length word, there are 26^k possible words that can be made. As described by Furruz et al.,

> Protein sequences can be described as a concatenation of letters from a chemically defined alphabet, the natural amino acids, and like human languages, these letters arrange to form secondary structural elements ("words"), which assemble to form domains ("sentences") that undertake a function ("meaning"). [11]

Let's extend the analogy of peptides to language further. Words used today derive from certain root words but widely diverge from the root word's original meaning. An example of this is the Semitic root qnw meaning cane or reed. This root word gave us words like canon (in music), canister, and canyon. Each word diverges from the other in its modern usage —it is unlikely that one would use a musical canon and canyon in similar contexts. Similarly, the effect of primordial peptides —which we know act like words —can be seen in modern-day proteins. Like one root word that spurs the creation of words with different meanings, primordial peptides are found in modern-day proteins with vastly different functions [4].

Furthermore, the language of peptides, as was touched on earlier, contains typos, grammar errors, and complete nonsense. Consider the following sentence:

> Jane lies to her friend.

If we add a letter, we can get the sentence: Janes flies to her friend. Suddenly, Jane has sprouted wings. Changing a letter in the sentence still results in correct words, but the meaning of the sentence has changed. Similarly in biology, mutations can add or change one of the amino acids in a peptide, changing the function of the final protein —one letter is changed which alters the meaning of the sentence. We call the mutations that alter one amino acid in a sequence missense mutations. For example, a missense mutation in the BRCA1 gene affects the final protein's ability to bind to the tumor suppressor protein p53. A failure to bind to a tumor suppressor promotes cancer growth. The missense mutations in BRCA1 are ultimately responsible for 80% of hereditary breast and ovarian cancer [23]. In biology, being one letter off can have devastating consequences.

While any metaphor can't be perfect, the comparison between language and peptides is pretty good. As we've seen in the example of "root words" in protein evolution,

the use of linguistic techniques in biology can yield important biological insight. Furthermore, by understanding how and why linguistics applies to biology, we have a better intuition for which techniques from natural language processing *should* be applied in biology. A great illustration of this is BLAST, an algorithm routinely employed by biologists to determine sequence similarity between two peptides.

BLAST is an algorithm chiefly concerned with sequence similarity. Similarity between sequences can mean there is homology between the sequences, but not necessarily. To conflate the two was cheekily described by Pertsemlidis & Fondon III as an act of BLASTphemy [21]. Sequence similarity means that there are patterns of letters in common between two sequences while homology means that the two proteins share a common ancestor. To call back to an earlier analogy, homology means that two peptide sequences share the same root word.

BLAST works by crawling along a peptide sequence three letters at a time. For instance, ABCDE would be split into ABC, BCD, and CDE and each smaller sequence would be analyzed as an input for the next step in the algorithm. Because there are 20 amino acids, there are 20^3 possible sequences for a three-letter sequence —and no less because the order here matters. BLAST then compares the three-letter sequence to all possible sequences using a scoring matrix[3]. Low-scoring comparison sequences are then discarded if they fall below a certain threshold [19]. BLAST doesn't care about the linguistic differences between sequences —i.e. are the sentences and meaning different? Instead, BLAST is like comparing words to each other, where bike and kite are called "similar" even though bikes and kites are quite different. But comparing words can be useful —canyon and canon have "can" and "on" in common and share the same root word. Sometimes similarity search can likewise be useful to determine which proteins might be related, but certainly more information is needed.

So we've seen that there are strong similarities between peptides and language, suggesting that linguistic techniques could be used to uncover biological insight. We've also looked at the example of BLAST, an algorithm that is useful but requires more power behind it to determine how proteins are related. To truly interrogate the linguistic structure of peptides, we need language models.

[3]For those who are interested, the scoring matrix that is used is called BLOSUM62

1.4. Autoregressive Models: You Won't Believe What Comes _____ !

Consider the following sentence:

I'm just a soul whose intentions are good
Oh Lord, please don't let me be [blank].

Clearly the last word of the sentence should be misunderstood. But for those who have not had the pleasure to listen to The Animal's *Don't Let Me Be Misunderstood*, the final word is not obvious. What to do? One approach is to begin by representing this sentence as a time series $y_1, y_2, \ldots, y_t$ [26]. At a given time n, we could have $y_n =$ "intentions", $y_{n+1} =$ "are", $y_{n+2} =$ "good" and so on. We could then take the final word to be a product of previous words. For instance, the word "good" is likely to come after the phrase "whose intentions are". Each word in the phrase could be assigned a weight depending on its significance in predicting the final word "good." In this example, "intentions" might have a higher weight than "are" since intentions tend to be good or bad, while "are" is a fairly common word. Formally, to predict y_t we use the following equation:

$$y_t = c + \sum_{i=1}^{p} w_i y_{t-i} + e_t,$$

where w_i is a learned weight and c and e_t are parameters. Models which use this type of objective function are termed autoregressive models. You've probably already encountered the most popular autoregressive model —GPT3 [14].

Ultimately, autoregressive models seek to predict a word in terms of the words that precede it. To do this, we use a function whose weights we must determine —we need to approximate the function. In the past, function approximation was done using Fourier series or Taylor series, but this is the 21^{st} century —we have access to a universal function approximator. We have neural networks [15]. With the use of neural networks, we can solve our objective function and develop good predictions based on preceding words. A popular autoregressive model in biology, ProtGPT2, uses a transformer architecture, a type of neural network, to generate novel peptide sequences [11].

Autoregressive models tend to be uni-directional, meaning they analyze text in only one direction, like how a person might read a sentence in English from left to right. While this works well for plenty of natural language processing (NLP) tasks, the language of biology is not so simple. For instance, palindromes frequently show up in peptide sequences. The use of palindromes in a sequence seems to have biological significance. For example, palindromes have a four-fold higher likelihood of being found in alpha helices, a

type of fold in the secondary structure of a protein, compared to random sequences [25]. The classic example of palindromes in biology is CRISPR, which stands for clustered regularly interspaced short palindromic repeats. Indeed, palindromes in biology are so important that their use can even win Nobel Prizes, as Jennifer Doudna and Emmanuelle Charpentier did with CRISPR in 2020 [16]. Thus, particularly in biology, reading in a sequence from both directions could be important. Unsurprisingly, bi-directional models which examine a sequence from both directions outperform uni-directional autoregressive models for biological tasks. Furthermore, there was a significant increase in the performance of the uni-directional protein language model ProtTXL when it became the bi-directional model ProtXLNet [9]. While autoregressive models are great for sequence generation, they fall short in other tasks because they do not process language in a bi-directional way.

Autoregressive models can also fail to consider the larger context of a sentence and cannot always capture long-range dependencies [26]. Let's illustrate this with the following example from the 1987 film *The Princess Bride* [24]:

Vizzini: No more rhymes today, I mean it.
Fezzik: Anybody want a peanut?

If we obscure "mean it" from our sentence, autoregressive models have a more difficult prediction task compared to bi-directional models. Fezzik responds with "peanut" because it rhymes with "mean it." A model that reads the words around "mean it" therefore has an easier time of prediction because it has access to additional context that an autoregressive model does not (for a visual representation of this, see Figure 1.3). This concept is especially important within peptide sequences. Often one part of the sequence interacts with an entirely different part of the sequence to fold together [10] (for a visual, see Figure 1.4). Without knowing what falls to the right of a sequence and only considering the left of the sequence (the preceding characters), a model misses out on key information.

While autoregressive models work well for language tasks, biological language presents a unique challenge in that context especially matters, as does the direction the sequence is analyzed. To truly capture the intricacies of peptide sequences, we need a different model.

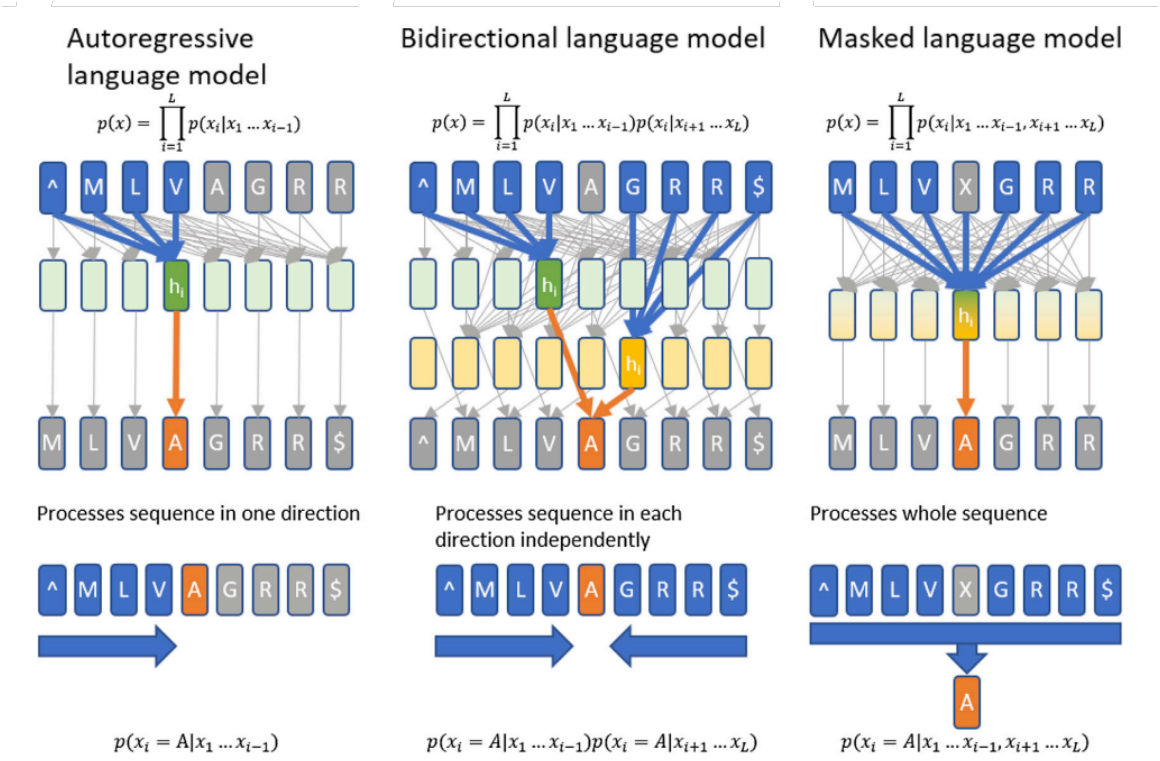


Figure 1.3. A summary of the typical objective functions and neural network architectures used to solve language problems in biology. Figure taken from [7].

1.5. ELMo and BERT: Tell Me How to Get, How to Get to a Good Embedding[4]

Autoregressive models consider the preceding words in a sequence because the objective of the model is to predict the next word. To get a model that wants to learn context, we need a different objective function. The typical objective function used to improve the contextual clues a model uses for prediction tasks is called an autoencoder. It works by randomly corrupting parts of a sequence and then the model trains by attempting to reconstruct this sequence.

To work with sentences, we first transform the words in the sentence into vectors in a process known as word embedding. Once a word is represented as a vector, it becomes easier to perform computations. A popular technique to produce these embeddings is ELMo (Embeddings from Language Models).

[4]To learn more about ELMo, BERT, and their friends, see the following link

ELMo uses a bidirectional approach, meaning it considers the words to the left and right to encode semantic meaning in the vector used to represent a word. Unlike previous approaches to word embedding, ELMo considers the whole input sentence when deciding upon how to embed a word as a vector [22].

ELMo seems particularly well-suited for biological applications. ELMo has previously been used to detect homonyms, pairs of words that have the same spelling but different meanings depending on context [17]. For instance, current is a homonym as it can be used in the context of current events or in the context of electrical current. Homonyms are especially important in biology. Because there are only 20 amino acids, amino acids tend to be reused in peptide sequences, often in different contexts [7]. Furthermore, homonyms of up to 7 amino acids in length have been found in proteins. These homonyms exhibit different structures in different proteins, similar to how homonyms in one sentence could have a different meaning in another [13]. ELMo's ability to consider contextual clues when computing word embeddings combined with its ability to distinguish between homonyms ultimately makes it well-suited for biological applications.

Using ELMo, we now have vector representations of words we wish to analyze. Thus, we can finally formulate our model. Because context matters, we wish to use an autoencoder model. Formally, we let X be a vector space of decoded messages and Y be a vector space of encoded messages. By encoded messages, we mean the vector has been corrupted such that certain tokens in the original sequence were replaced with filler tokens. By decoding a message, our model seeks to return the original uncorrupted sequence. We represent an encoder function as $E_\phi : X \to Y$ (E for encoder) and the decoder function as $D_\theta : Y \to X$ where ϕ and θ are respective parameters for our two functions. A perfect decoder, therefore, returns the original sequence after it has been encoded, i.e. $D_\theta(E_\phi(x)) = x$ for all $x \in X$.

We know, however, that perfect decoders are difficult to achieve. Instead, our decoder may return another vector $x' \in X$ after encoding, i.e. there exists $x \in X$ such that $D_\theta(E_\phi(x)) = x'$ for some $x' \in X$. Thus, we want to train our model such that it minimizes the distance between x' and x in the vector space X. To do this, we need a function. We can therefore define our loss function L as follows:

$$L := \frac{1}{N} \sum_{i=1}^{N} \|x_i - D_\theta(E_\phi(x_i))\|_2^2 ,$$

where N is the number of elements in X we choose to subset, i.e. we take $\{x_1, x_2, ..., x_N\} \in X$ as elements to compute our loss function [2].

Our model's task is to minimize L. In plain English, we wish to minimize the average distance between our original message and the reconstructed message.

To actually train our model, we use an over complicated version of a simple principle: guess, check, and correct. We guess weights to help our encoder and decoder functions recreate the original sequence. Then, we check how good these weights are by computing our loss function. If our loss is high, we readjust our weights through a process known as backpropagation [3]. This process continues as we train our model.

An example of one such model is BERT, a common type of masked autoencoder model used for natural language processing (NLP) tasks [14]. BERT utilizes a transformer architecture, a type of neural network, to train its model. The autoencoder objective function is *what* BERT uses to train; a transformer is *how* it trains [8]. Transformers have two main features: 1) positional encoding and 2) attention. In language, order matters. Positional encoding allows a transformer to store where in a sentence each word is located to better understand its meaning. Attention, as the name suggests, tells the model what to pay attention to. Think about a portrait of a person. There may be trees in the background, but clearly, the focal point is the person in the photo. In the same way, transformers use attention as a mechanism to determine what to focus on in a sentence so as to disregard the background and less important details.

When calculating attention, we use three inputs: queries, keys, and values, represented by the matrices Q, K, and V, respectively. A query is what word the model wants to focus on, similar to how Google search queries are used to pull up results. The query is then compared to keys, other words which may be similar to the query we wish to characterize. If the query and key are similar, their associated value has a higher weight. Formally, we compute attention using the following formula:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{Q \cdot K^T}{\sqrt{d_k}}\right)V,$$

where d_k is the dimension of the matrix K [27].

Recall that we previously embedded words as vectors. When we take the dot product of our query and key, we're really comparing the semantic meanings of the original words. Two words with similar semantic meanings will embed in Q and K similarly, meaning the coordinates of the vectors will be similar, resulting in a larger dot product. By extension, vectors associated with two words of different semantic meanings are orthogonal, and thus their final weighted value is zero. A softmax function is then applied to normalize the weights. Thus, our formula for attention can be intuitively thought of

as weighing final values based on the relative magnitude of semantic similarities between embedded words, i.e. pay attention to similar words to predict meaning.

Once we've trained BERT, we now have a decent description of semantic similarities between sentences, or for our purposes, peptide sequences. Once a model is trained on one task, it can be used further on seemingly unrelated downstream tasks in a process known as transfer learning. Bepler & Berger liken the process to how in the movie *The Karate Kid*, learning how to wax on and off offers transferable skills in karate [5]. Biological models of BERT have a stronger "intuition" for how information is encoded in peptide sequences which can then be used to predict protein structure and function in other tasks. This "intuition" is encoded by the weights used to make up the functions governing the model and allows for transfer learning. Transfer learning is especially useful when training data is lacking, as is the case with protein function prediction. The functions of many proteins are unknown, making it difficult to accurately train a prediction model. However, models trained on other tasks with plentiful training data available can then be used to predict protein function successfully [7]. For instance, transfer learning was used by Bepler & Berger to predict which parts of a protein are inside vs. outside the cell membrane based on an initial training task of embedding peptide sequences as vectors [6].

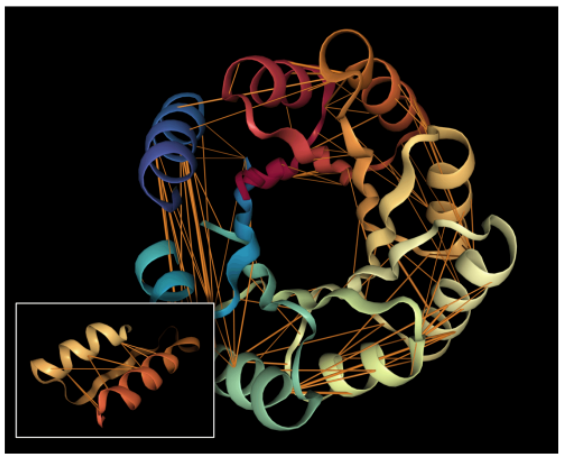
The problem with models like BERT is that this learned intuition is guarded by a black box —we don't know exactly why the model performs well at certain tasks. We only know that by learning the weights associated with accuracy in one task, accuracy transfers over to other tasks. To better understand the biology behind BERT's success, Vig et al. examined where BERT's transformer architecture places its attention [28]. In one instance, they examined contact maps, a matrix c where entry $c_{ij} = 1$ if amino acids i and j are in contact in the tertiary protein structure and $c_{ij} = 0$ otherwise. High attention weights above a threshold, θ, were then analyzed. For an n-length input sequence x in the model, the proportion, $p(c)$, of pairs of amino acids in contact with high attention can be calculated as follows:

$$p(c) = \frac{\sum_{x \in X} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} \cdot \alpha_{i,j}(x)}{\sum_{x \in X} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_{i,j}(x)},$$

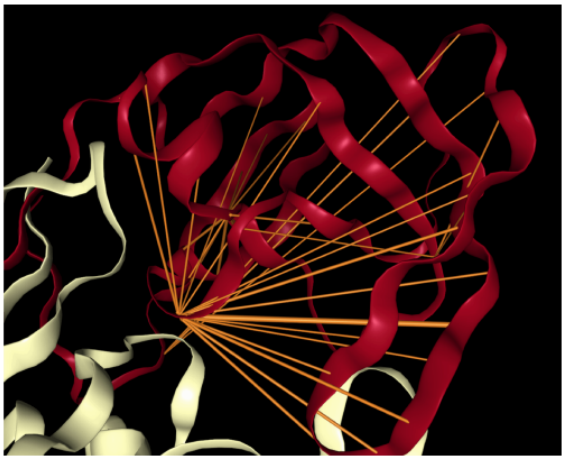
where $\alpha_{i,j}(x)$ is the attention weight for the input vector x between amino acids i and j.

The authors found that one variation of BERT for protein structure prediction,

ProtBERT, contained a layer in which over 50% of its attention was focused on contact maps. This is remarkable considering that ProtBERT is a language model without any spatial or topological information included in its dataset —it makes predictions purely based on peptide sequence. Furthermore, the model also focuses attention on amino acids which are far apart from one another in the peptide sequence but which are close together in the 3D structure (see Figure 1.4). Thus, the model is able to predict the linguistic context clues we wished to encode in our model.



(a) Attention in head 12-4, which targets amino acid pairs that are close in physical space (see inset subsequence 117D-157I) but lie apart in the sequence. Example is a *de novo* designed TIM-barrel (5BVL) with characteristic symmetry.



(b) Attention in head 7-1, which targets binding sites, a key functional component of proteins. Example is HIV-1 protease (7HVP). The primary location receiving attention is 27G, a binding site for protease inhibitor small-molecule drugs.

Figure 1.4. Amino acids which are far apart in a peptide sequence are able to coordinate with each other in 3D. Attention in transformer models captures structural interactions between different amino acids. Attention also appears to highlight binding sites on proteins, which is an important feature for determining molecules that can interact with the protein. Figure taken from [28].

To briefly summarize, ELMo is used to create vectors known as word embeddings which are used as inputs into autoencoder models like BERT. Because BERT is able to consider contextual information by the very nature of its objective function, it is able to capture valuable biological information as to how proteins fold and function. To aid in the training of this model, transformers are employed which use attention to disregard irrelevant information to improve accuracy. Despite these models being agnostic to topological data when training, their architecture appears to be invaluable in answering biological questions.

1.6. Final Thoughts

Deciphering the code found within each of our cells is no small task. DNA becomes amino acids which become proteins —a simple and foundational idea underpinning life that only supercomputers seem capable of beginning to tackle. Our own communication carries with it the language found inside our cells. What is language if not the cellular language giving itself the ability to speak? Advances in the mathematics of language modeling have only heightened our ability to see the similarities between the protein language and our own. The protein language possesses context and meaning, where even one fault in the language can mean the difference between life and death. As our protein language models advance, our ability to promote human flourishing advances too.

References

[1] A third of new drugs cause serious problems when more people take them, May 2017.

[2] Autoencoder, February 2023. Page Version ID: 1141727025.

[3] Multilayer perceptron, March 2023. Page Version ID: 1147594876.

[4] Vikram Alva, Johannes Söding, and Andrei N Lupas. A vocabulary of ancient peptides at the origin of folded proteins. *eLife*, 4:e09410, December 2015. Publisher: eLife Sciences Publications, Ltd.

[5] John G. Avildsen. The Karate Kid, 1984.

[6] Tristan Bepler and Bonnie Berger. Learning protein sequence embeddings using information from structure, October 2019. arXiv:1902.08661 [cs, q-bio, stat].

[7] Tristan Bepler and Bonnie Berger. Learning the protein language: Evolution, structure, and function. *Cell Systems*, 12(6):654–669.e3, June 2021.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, October 2018.

[9] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rihawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Debsindhu Bhowmik, and Burkhard Rost. 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*, 44(10), July 2021. Institution: Oak Ridge National Lab. (ORNL), Oak Ridge, TN (United States) Publisher: IEEE.

[10] Noelia Ferruz and Birte Höcker. Towards Controllable Protein Design with Conditional Transformers. 2022.

[11] Noelia Ferruz, Steffen Schmidt, and Birte Höcker. ProtGPT2 is a deep unsupervised language model for protein design. *Nature Communications*, 13(1):4348, July 2022. Number: 1 Publisher: Nature Publishing Group.

[12] Patrick J. Fleming, Haipeng Gong, and George D. Rose. Secondary structure determines protein topology. *Protein Science*, 15(8):1829–1834, 2006. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1110/ps.062305106.

[13] Derek Gatherer. Peptide Vocabulary Analysis Reveals Ultra-Conservation and Homonymity in Protein Sequences. *Bioinformatics and Biology Insights*, 1:BBI.S415, January 2007. Publisher: SAGE Publications Ltd STM.

[14] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollar, and Ross Girshick. Masked Autoencoders Are Scalable Vision Learners. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15979–15988, New Orleans, LA, USA, June 2022. IEEE.

[15] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, January 1989.

[16] Heidi Ledford and Ewen Callaway. Pioneers of revolutionary CRISPR gene editing win chemistry Nobel. *Nature*, 586(7829):346–347, October 2020. Bandiera_abtest: a Cg_type: News Number: 7829 Publisher: Nature Publishing Group.

[17] Younghoon Lee. Systematic Homonym Detection and Replacement Based on Contextual Word Embedding. *Neural Processing Letters*, 53:1–20, February 2021.

[18] Sayaka Mizutani, Edouard Pauwels, Véronique Stoven, Susumu Goto, and Yoshihiro Yamanishi. Relating drug–protein interaction network with drug side effects. *Bioinformatics*, 28(18):i522–i528, September 2012.

[19] David W. Mount. Bioinformatics Sequence and Genome Analysis. pages 281–335. Cold Spring Harbor Laboratory Press.

[20] Randall Munroe. Proteins, 2014.

[21] Alexander Pertsemlidis and John W. Fondon. Having a BLAST with bioinformatics (and avoiding BLASTphemy). *Genome Biology*, 2(10):reviews2002.1, September 2001.

[22] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations, February 2018.

[23] Barbara Quaresima, Maria C. Faniello, Francesco Baudi, Telma Crugliano, Maddalena Di Sanzo, Giovanni Cuda, Francesco Costanzo, and Salvatore Venuta. Missense mutations of BRCA1 gene affect the binding with p53 both in vitro and in vivo. *Oncology Reports*, 16(4):811–815, October 2006. Publisher: Spandidos Publications.

[24] Robert C. Reiner. The Princess Bride, 1987.

[25] Armita Sheari, Mehdi Kargar, Ali Katanforoush, Shahriar Arab, Mehdi Sadeghi, Hamid Pezeshk, Changiz Eslahchi, and Sayed-Amir Marashi. A tale of two symmetrical tails: Structural and functional characteristics of palindromes in proteins. *BMC Bioinformatics*, 9:274, 2008. Publisher: BioMed Central.

[26] Oskar Triebe, Nikolay Laptev, and Ram Rajagopal. AR-Net: A simple Auto-Regressive Neural Network for time-series, November 2019. arXiv:1911.12436 [cs, stat].

[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, December 2017. arXiv:1706.03762 [cs].

[28] Jesse Vig, Ali Madani, Lav R. Varshney, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. BERTology Meets Biology: Interpreting Attention in Protein Language Models, June 2020.