Group 2: Ab Initio Secondary Structure Prediction

Final Report

Group Members:
Arth Banka
Riti Bhatia
Sanchitha Kuthethoor
Sumeet Kothare

Abstract

Predicting protein structure using mathematical and computational methods has been an area of great interest for several decades now. This project explores two foundational secondary structure prediction algorithms, Chou-Fasman (CF) and Garnier-Osguthorpe-Robson (GOR), which were pivotal in the 1970s for leveraging statistical analyses. Additionally, we experimentally implemented a Hidden Markov Model (HMM). Our models predict four secondary structure categories: alpha helices, beta-sheets, turns, and coils. When compared to DSSP structure predictions for 50 proteins, Chou-Fasman achieved an average accuracy of 35.02%, GOR achieved 41.19%, and our experimental HMM surprisingly achieved an accuracy of 47.29%. By comparing predictions from each model through an interactive interface, we highlight a critical point in the evolution of computational methods that laid the groundwork for modern state-of-the-art AI-driven advancements like AlphaFold2. This study underscores the transformative role of traditional algorithms in the progression of protein structure prediction.

Introduction

Since the early 1900s, scientists have sought to understand the structure of proteins, recognizing the value of inferring function from structure. The very first three dimensional structure of a protein, myoglobin from sperm whales, was determined in 1958 through X-ray crystallography [10]- a breakthrough that took over two decades [11]. Researchers were optimistic that this feat would enable the prediction of all protein structures from a few examples, as seen with the symmetry of DNA's double helix. However, the complex structure of myoglobin shattered this hope, as it showed that every protein likely has a unique structure that is born out of various complex intermolecular interactions.

The field progressed significantly with Christian Anfinsen's pioneering experiment in 1961 [12], which demonstrated that given the right conditions, the information needed for a protein to fold into its functional structure is entirely within its amino acid sequence. This thermodynamic hypothesis laid the foundation for understanding protein folding and the relationship between the amino acid sequences and a protein's native 3D conformation. Predicting a protein's structure based on its primary sequence became a critical pursuit, driven by the resource and time intensive nature of experimental methods. This highlighted the need for computational and mathematical approaches to structure prediction.

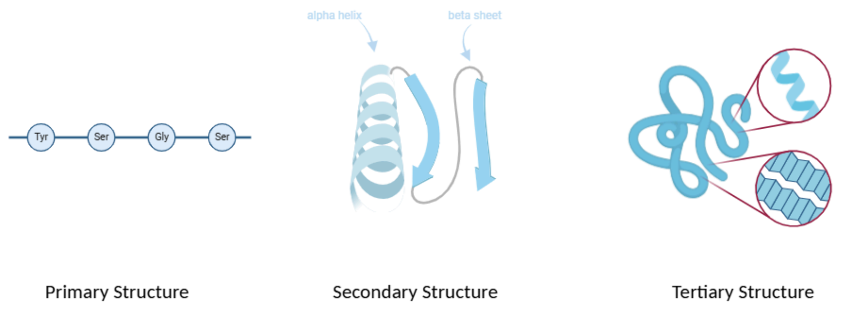


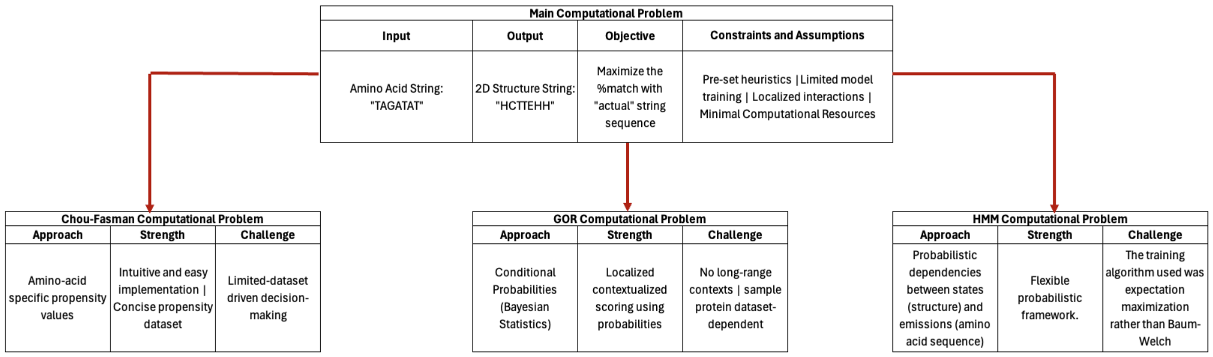
Figure 1**:** Cartoon representation of the three levels of protein structure

Protein folding involves hierarchical levels of structure: the primary structure is the amino acid sequence or polypeptide chain; the secondary structure is formed by hydrogen bonds formed between the amino acids resulting in local motifs such as alpha helices and beta sheets; and the tertiary structure involves interactions between the amino acid side chains resulting in a 3D folded conformation. Early efforts in protein structure prediction focused on secondary structure prediction, and some of the successful and pioneering algorithms that were developed in the 1970s were the Chou-Fasman (CF) and the Garnier-Osguthorpe-Robson (GOR) algorithm, which were based on statistical analysis of empirical data to perform its predictions. We have come a long way since then, with AlphaFold2 using AI-ML approaches to predict tertiary protein structure from an amino acid sequence very quickly and accurately.

In this project, we implemented the two foundational secondary structure prediction algorithms, Chou-Fasman and GOR, to appreciate their contributions to computational biology and transforming the field of protein structure prediction. We also extend our exploration by employing an experimental Hidden Markov Model (HMM). The models we have implemented predict 4 broad categories of secondary structures - alpha helices (H), beta-sheets (E), turns (T), and coils (C). Our implementation includes a user-friendly interface that allows users to input amino acid sequences and compare predictions from these three prediction methods. Our project aims to highlight the significance of computational approaches in protein structure prediction while exploring how traditional algorithms paved the way for modern machine learning models that have astounding capabilities today.

Computational Problem

In this project, we were motivated by a central theme: **predicting the secondary structure of proteins from their amino acid sequences** using straightforward, pre-defined heuristic methods. Rather than relying on modern, data-intensive machine learning approaches, we chose to revisit and implement earlier, "classic" algorithms—such as Chou-Fasman, GOR, and simple Hidden Markov Models (HMMs)—to assess their predictive accuracy against established "ground-truth" or industry-standard predictions (e.g., derived from DSSP or reliable online prediction tools). This historical and heuristic-driven perspective allows us to understand how early computational biology tackled a complex prediction problem without the luxury of advanced machine learning frameworks.



Main Computational Problem			
Input	**Output**	**Objective**	**Constraints and Assumptions**
Amino Acid String: "TAGATAT"	2D Structure String: "HCTTEHH"	Maximize the %match with "actual" string sequence	Pre-set heuristics |Limited model training | Localized interactions | Minimal Computational Resources

Chou-Fasman Computational Problem		
Approach	**Strength**	**Challenge**
Amino-acid specific propensity values	Intuitive and easy implementation | Concise propensity dataset	Limited-dataset driven decision-making

GOR Computational Problem		
Approach	**Strength**	**Challenge**
Conditional Probabilities (Bayesian Statistics)	Localized contextualized scoring using probabilities	No long-range contexts | sample protein dataset-dependent

HMM Computational Problem		
Approach	**Strength**	**Challenge**
Probabilistic dependencies between states (structure) and emissions (amino acid sequence)	Flexible probabilistic framework.	The training algorithm used was expectation maximization rather than Baum-Welch

Figure 2**:** Infographic depicting the Main Computational Problem and our "problem-chunking" to define the 3 approaches to solving the main problem.

Main Computational Problem:

Given an input amino acid sequence (a string of single-letter amino acid codes), produce an output string indicating the secondary structure state (e.g., H, E, T, C for helix, strand, turn, coil) for each residue. The objective is to maximize the percentage match between this predicted structure and the actual experimentally validated structure. We operate under minimal computational overhead and use models that rely on relatively simple statistical insights or state-based frameworks, without large-scale model training.

Methodology

Model 1: Chou-Fasman Model

The Chou-Fasman algorithm predicts the secondary structure of an amino acid sequence based on empirically derived propensity scores of each amino acid to participate in one of three structures- alpha helix, beta sheets, or turns. Amino acids are classified into categories that classify them as either "formers" or "breakers" of these secondary structures, and using this, the algorithm predicts the probability of a structure's occurrence at that residue. The original model reported a predicted accuracy of 70-80%, which changed to 50-60% when the algorithm was implemented computationally. This was likely due to the fact that the original paper also relied on the intuition of the scientists while they were making predictions[12][13].

Definitions

1. **Propensity**-

In the original study, the propensity for an amino acid, *i,* to adopt a particular structural conformation was calculated from the database of known structures with the following equations-

$$P(i|x) = \frac{Frequency\ of\ amino\ acid\ i\ in\ x}{Overall\ frequency\ of\ amino\ acid\ i}, \text{ where } x = \alpha \lor \beta \lor T$$

In other words, this equation is the conditional probability being normalized by the background probability of the amino acid.

For example if 25% of of the residues in helices are alanine, and 25% of the total amino acids in the data set are alanine, then the calculation would be - $P_{\alpha/\beta/T} i = \frac{25}{25} = 1$; so the propensity for alanine to be a helix would be 1.

2. **Former**- s amino acids that have a high propensity (usually over 1.0) to occur in a particular structure (either helix or sheet). These amino acids favor the formation of a particular structure.
3. **Breaker**- amino acids that have very low propensity to occur in a particular structure (either helix or sheet). They often disrupt the formation of a particular structure, and cannot be found at the nucleation sites.
4. **Bend probabilities table**- reflects how likely a sequence of four consecutive residues is to form a turn based on their observed frequencies in the experimentally determined structures (from the original paper).

Algorithm

Alpha Helix Prediction: (P_{α} = propensity score of a residue to form helix)

1. Find a nucleation site
   a. A sliding window of 6 amino acid residues is created and average P_α is calculated.
   b. "Formers" ($P_\alpha \geq 1.05$) will contribute 1 point to the former count, and "weak formers" ($P_\alpha \geq 1.00$) will contribute 0.5 point to the former count.
   c. "Breakers" ($P_\alpha \leq 0.69$) will contribute 1 point to the breaker count.
   d. If the average P_α of the 6 residue window is ≥ 1.03, the former count is ≥ 4, and the breaker count is < 2, then the window is favourable to be a nucleation site for a helix. If not, then slide along and repeat until a nucleation site is found.
2. Once a nucleation site is found, the window extends in either direction.
   a. Continue extending in either direction until average P_α falls below the 1.03 threshold, or, a set of tetrapeptide (i.e 4 consecutive) breakers are encountered.
3. Predict the entire window as a helix (H).
4. Continue sliding the window to find the next nucleation site.

Beta Sheet Prediction: (P_β = propensity score of each residue to form sheet)

1. Find a nucleation site
   a. A sliding window of 5 amino acid residues is created and average P_β is calculated.
   b. "Formers" ($P_\beta \geq 1.00$) will contribute 1 point to the former count.
   c. "Breakers" ($P_\beta \leq 0.75$) will contribute 1 point to the breaker count.
   d. If the average P_β of the 5 residue window is ≥ 1.05, the former count is ≥ 3, and the breaker count is < 2, then the window is favourable to be a nucleation site for a helix. If not, then slide along and repeat until a nucleation site is found.
2. Once a nucleation site is found, the window extends in either direction.
   a. Continue extending in either direction until average P_{P_β} falls below the 1.05 threshold, or, a set of tetrapeptide (i.e 4 consecutive) breakers are encountered.
3. Predict the entire window as a sheet (E).
4. Continue sliding the window to find the next nucleation site.

Turn Prediction: (P_t = propensity score of each residue to form turn)

For a 4 residue window:

1. Calculate average P_t .

2. Calculate bend probability $p_t = f_i \times f_{i+1} \times f_{i+2} \times f_{i+3}$.
3. If the average P_t is ≥ 1.00 and the bend probability (p_t) is ≥ 0.000075, and the f_{i+1} and f_{i+2} individually have $P_t \geq 0.5$, predicts the region as a turn (T).
4. Continue sliding the window until the conditions are satisfied.

Resolve Overlapping Regions:

- If regions predicted as “Helix” or “Sheet” or “Turn” overlap, the prediction with the highest average propensity score will be the predicted structure for the region.

All unassigned residues get the default structure of “Coil”.

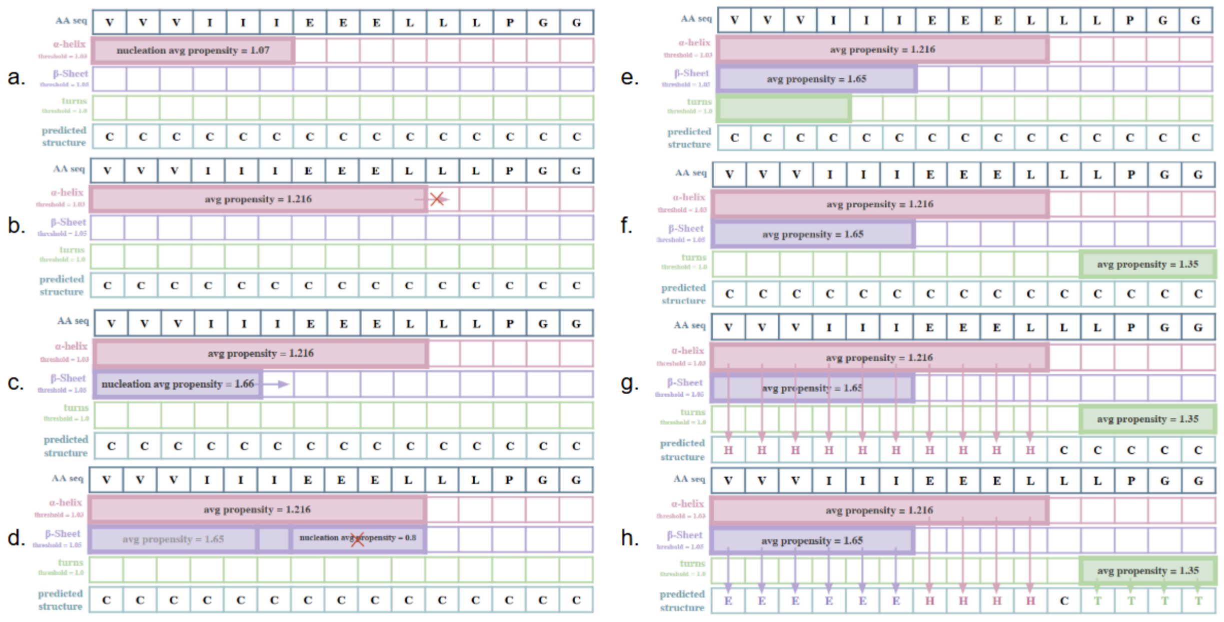

a.
b.
c.
d.
e.
f.
g.
h.
AA seq
α-helix
β-Sheet
turns
predicted structure
V V V I I I E E E L L L P G G
nucleation avg propensity = 1.07
avg propensity = 1.216
nucleation avg propensity = 1.66
avg propensity = 1.65
nucleation avg propensity = 0.8
avg propensity = 1.35
C C C C C C C C C C C C C C C
H H H H H H H H H H C C C C C
E E E E E E H H H H C T T T T

Figure 3. A figure depicting the overall Chou Fasman algorithm (note: steps have been skipped). **a.** Shows the favourable helix nucleation site, **b.** Shows the final extended helix window, **c.** Shows the favourable sheet nucleation site, **d.** Shows the sliding window looking for more nucleation sites (in this case it was unfavourable), **e.** Looking for turn regions (propensity score not added for simplicity) **f.** Shows found favourable turn regions, **g.** Shows updating predicted structures onto the final prediction structure, **h.** Shows overlaps being handled based on higher propensity scores.

Challenges & Limitations

Limited Dataset and Generalizability:

The algorithm relies on empirical data obtained from observed frequencies of amino acids in a small set of 29 known protein structures. As a result, its rules and propensities may not generalize well to other proteins. If the scores were derived from a larger and more diverse set of protein structures, the accuracy could potentially improve.

Ignores Long-Range Interactions:

The Chou-Fasman algorithm considers only the local sequence context, focusing on the immediate neighbors of amino acids to predict secondary structure. It does not account for long-range interactions, which play a significant role in protein folding.

Simplified Structure Categories:

The algorithm predicts only four broad categories of secondary structures—alpha helices (H), beta-sheets (E), turns (T), and coils (C). This simplification fails to capture the nuanced and diverse structures found in most proteins, such as intricate loop formations or irregular motifs.

Overlapping Sequence Handling:

In regions of overlapping sequence propensities, the algorithm replaces structures based solely on higher average propensities. This simplistic approach overlooks the biological reality that secondary structures do not transition abruptly (e.g., from a helix to a sheet) without intermediary spacer regions, such as bends or coils, to accommodate these transitions.

Accuracy Constraints:

The Chou-Fasman algorithm achieved an average accuracy of 35.02% in our study, lower than the reported 50% in earlier studies. This discrepancy may arise because many online implementations use updated datasets with refined propensity values, adjusted thresholds, and improved handling of overlapping sequences. Modern versions address these limitations, resulting in more biologically realistic predictions and higher accuracy.

Model 2: Garnier-Osguthorpe-Robson (GOR) Model

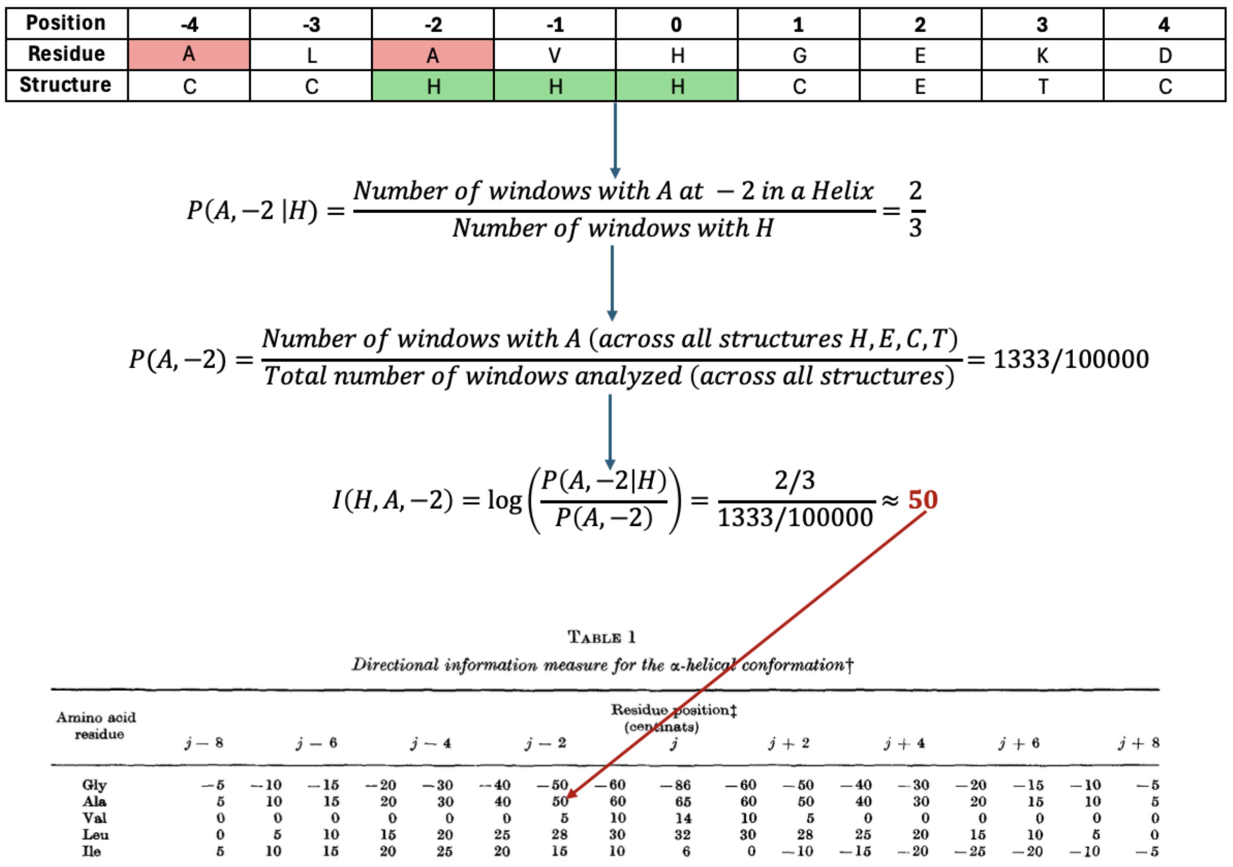
Algorithm

Overview of the GOR Method

The GOR (Garnier-Osguthorpe-Robson) method is a statistical approach used for predicting protein secondary structures based on amino acid sequences. It utilizes information theory and Bayesian statistics to calculate the propensity of each residue to adopt a particular secondary structure—Helix (H), Beta-strand (E), Turn (T), or Coil (C)—by considering the influence of neighboring residues.

Figure 4 below illustrates the calculation of an information score for Alanine (A) from a collection of proteins using a short sample sequence. Initial version of GOR used 26 proteins with known secondary structures with subsequent GOR model versions included upwards of 226 proteins. The residue sequence in Figure 4 represents one protein from such a database of proteins. First, we calculate the conditional probability, P(A,-2|H), of finding an Alanine residue at a position of -2 (second-last residue to a central residue at position 0) in a structural segment that forms a helix. As in Figure 4, this is calculated by dividing the number of windows with A at

-2 position with a helix at the central residue (red cells) by the number of windows with a helix structure (green cells). Secondly, we calculate the probability, P(A,-2), of finding Alanine in position -2 across all structures in the protein database. Lastly, the logarithm (to any base used consistently) of the division of these two probabilities is calculated to derive a unique information score for Alanine found at position -2 in a helical structure. This information score is then placed in a table specific to helices. The GOR authors chose a maximum of eight residues before and after a central residue to calculate such information scores and tabulate them as shown in Figure 4. We use four such tables, for each of the four structures, in our structure prediction process.



Position	-4	-3	-2	-1	0	1	2	3	4
Residue	A	L	A	V	H	G	E	K	D
Structure	C	C	H	H	H	C	E	T	C

$$P(A,-2\,|H) = \frac{Number\ of\ windows\ with\ A\ at\ -2\ in\ a\ Helix}{Number\ of\ windows\ with\ H} = \frac{2}{3}$$

$$P(A,-2) = \frac{Number\ of\ windows\ with\ A\ (across\ all\ structures\ H, E, C, T)}{Total\ number\ of\ windows\ analyzed\ (across\ all\ structures)} = 1333/100000$$

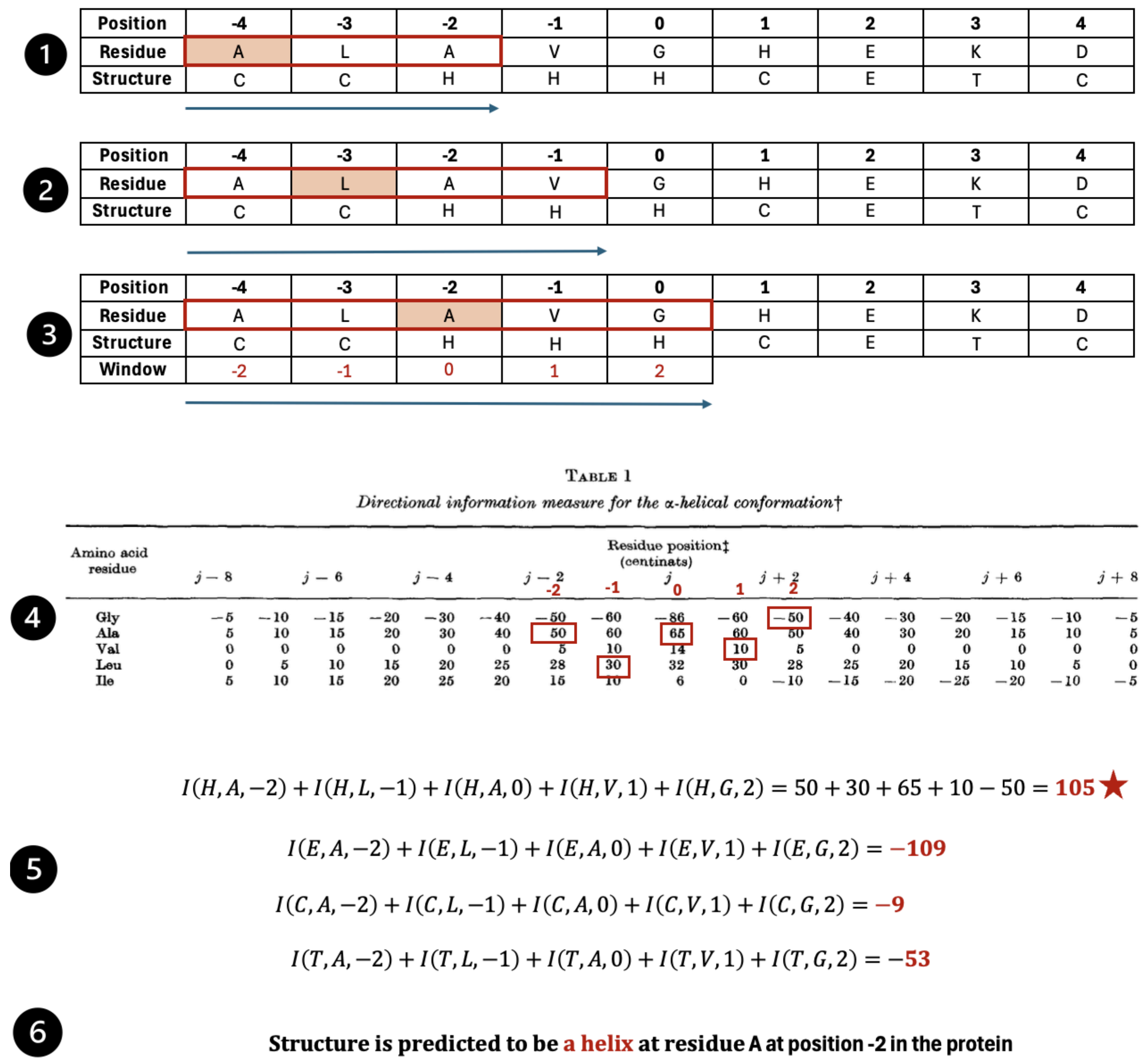
$$I(H,A,-2) = \log\left(\frac{P(A,-2|H)}{P(A,-2)}\right) = \frac{2/3}{1333/100000} \approx 50$$

TABLE 1

Directional information measure for the α-helical conformation†

Amino acid residue	Residue position‡ (centinats) j − 8		j − 6		j − 4		j − 2		j		j + 2		j + 4		j + 6		j + 8
Gly	−5	−10	−15	−20	−30	−40	−50	−60	−86	−60	−50	−40	−30	−20	−15	−10	−5
Ala	5	10	15	20	30	40	50	60	65	60	50	40	30	20	15	10	5
Val	0	0	0	0	0	0	5	10	14	10	5	0	0	0	0	0	0
Leu	0	5	10	15	20	25	28	30	32	30	28	25	20	15	10	5	0
Ile	5	10	15	20	25	20	15	10	6	0	−10	−15	−20	−25	−20	−10	−5

Figure 4: Calculating the information values using probabilities of finding an amino acid in a particular structural segment.



1

Position	-4	-3	-2	-1	0	1	2	3	4
Residue	A	L	A	V	G	H	E	K	D
Structure	C	C	H	H	H	C	E	T	C

2

Position	-4	-3	-2	-1	0	1	2	3	4
Residue	A	L	A	V	G	H	E	K	D
Structure	C	C	H	H	H	C	E	T	C

3

Position	-4	-3	-2	-1	0	1	2	3	4
Residue	A	L	A	V	G	H	E	K	D
Structure	C	C	H	H	H	C	E	T	C
Window	-2	-1	0	1	2				

4

TABLE 1

Directional information measure for the α-helical conformation†

Amino acid residue	Residue position‡ (centinats) $j-8$		$j-6$		$j-4$		$j-2$ -2	-1	j 0	1	$j+2$ 2		$j+4$		$j+6$		$j+8$
Gly	−5	−10	−15	−20	−30	−40	−50	−60	−86	−60	$\boxed{-50}$	−40	−30	−20	−15	−10	−5
Ala	5	10	15	20	30	40	$\boxed{50}$	60	$\boxed{65}$	60	50	40	30	20	15	10	5
Val	0	0	0	0	0	0	5	10	14	$\boxed{10}$	5	0	0	0	0	0	0
Leu	0	5	10	15	20	25	28	$\boxed{30}$	32	30	28	25	20	15	10	5	0
Ile	5	10	15	20	25	20	15	10	6	0	−10	−15	−20	−25	−20	−10	−5

5

$$I(H,A,-2)+I(H,L,-1)+I(H,A,0)+I(H,V,1)+I(H,G,2)=50+30+65+10-50=\mathbf{105}\bigstar$$

$$I(E,A,-2)+I(E,L,-1)+I(E,A,0)+I(E,V,1)+I(E,G,2)=\mathbf{-109}$$

$$I(C,A,-2)+I(C,L,-1)+I(C,A,0)+I(C,V,1)+I(C,G,2)=\mathbf{-9}$$

$$I(T,A,-2)+I(T,L,-1)+I(T,A,0)+I(T,V,1)+I(T,G,2)=-\mathbf{53}$$

6

Structure is predicted to be a helix at residue A at position -2 in the protein

Figure 5: Stepwise calculation of structural information scores and final structure prediction

Figure 5 illustrates the sliding window approach and shows the information scoring that leads to the prediction of a specific structure at a central residue. As mentioned above, GOR references 8 amino acid residues before and after a central residue leading to a window that contains 17 residues. This window was determined by the GOR authors as sufficient to capture the local influence that neighboring amino acid residues enforce on a central residue. Figure 5 illustrates this approach assuming a shorter 5-residue window. In Step 1, the window begins with the first residue in the entire protein sequence being considered as the central residue with only 2 amino acid residues after this residue being considered. Since the residues to the left of this central residue (orange cell) don’t exist, they are ignored or assumed to contribute a score of 0. In Step 2, as the central residue moves to the next residue to the right, the window expands with one residue available to the left. This expansion of the window continues until we reach the maximum window length of 5 as in Step 3. Step 3 also shows the Window indices from -2 through 2 including the central residue. Subsequently, for each of the amino acid residues in Step 3 with the central residue, A, at index 0 (protein position -2), information scores from the table for a helix are looked up (Step 4) and summed as shown in Step 5. This process is performed for beta sheets, coils, and turns by looking up the respective residues' information scores from the respective structure tables. Finally, the maximum information score out of the four calculated for each structure is chosen and the corresponding structure is predicted to be the structure at the central residue.

Limitations

Limited to Local Interactions in GOR:

The use of a fixed window size means that long-range interactions are not considered. Given a 3-dimensional folding of proteins, the entire strand of the protein sequence may dictate the structure in a particular region. For example, a region 20 amino acids further from the N-terminus may be impacted by a region 100 amino acids further from the N-terminus if the two regions come into proximity due to the folding of the strand. Additionally, the method assumes that the contributions of neighboring residues are independent and additive, which may not capture all the complexities of protein folding. Hence, the local interactions assumed in GOR underestimate non-localized effects.

Prediction Accuracy

As a statistical method, the GOR algorithm cannot account for all factors influencing protein structure, leading to potential inaccuracies. The quality of the prediction depends on the dataset used to derive the information tables. Biases in the dataset can affect results. For example, the updated GOR method by Garnier, Gibrat, and Robson used a curated protein database containing 267 proteins with residue length longer than 50 residues. This difference selection of the protein dataset skews the calculation of the probabilities and their derived information values. Capturing a large enough sample set of every residue occurring at each of the indices in the reading window for every conformation is intractable. This intractable issue leads to differences in approaches leading to an uneven comparison of prediction results across studies.

Edge Effects

At the beginning and end of the sequence, the window extends beyond the available residues. Optionally, padding the sequence with dummy residues or adjusting the window size may address this issue but this may introduce artifacts. For example, the structure score prediction for the first 8 residues may differ from the subsequent residues' structure score predictions since the reading window extends beyond the available residues leading to an assumption of a zero contribution. That is, fewer than 17 residues contribute to the structural prediction of the first 8 residues resulting in an under-defined overall prediction.

Simplification of Secondary Structures

The method reduces secondary structures to four states (H, E, T, C), which may oversimplify the diversity of actual protein conformations. This also introduces complexity during validation since different approaches, tools, and classifications by researchers result in an unequal comparison of the outcomes.

Experimental Model: Hidden Markov Model

Algorithm Overview:

Secondary structures, such as helices, sheets, turns, and coils, form the building blocks of proteins. Our project employs a Hidden Markov Model (HMM) to predict these secondary structures based on sequences of amino acids. This probabilistic approach is inspired by the ability of HMMs to capture sequential patterns and transitions between observable data (amino acids) and hidden states (secondary structures). By using the Viterbi algorithm, the model predicts the most probable sequence of secondary structures for a given protein sequence.

The HMM represents the protein secondary structure prediction problem as a sequence of states and symbols:

1. **States:** These correspond to secondary structures (Helix, Sheet, Turn, Coil).
2. **Symbols**: These represent the observable amino acid residues (20 types).

The model comprises:

1. **Transition Probabilities:** Define the likelihood of moving from one state (e.g., Helix) to another (e.g., Sheet).
2. **Emission Probabilities:** Represent the likelihood of observing a particular amino acid given a state (e.g., probability of Alanine in a Helix).
3. **Initial Probabilities:** Indicate the likelihood of starting in each state.

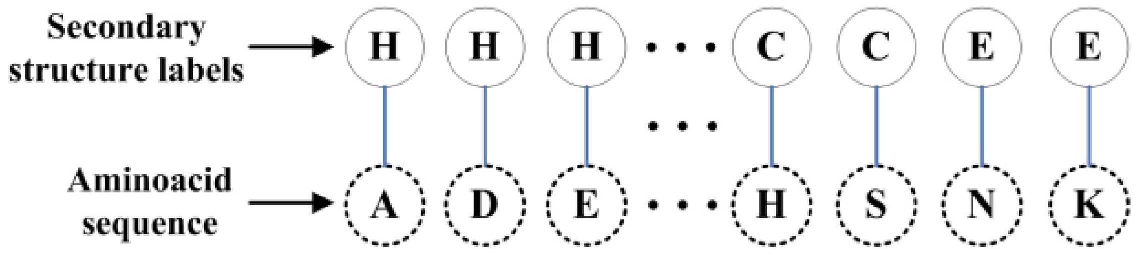


Figure 6: The figure above shows how our model would generate predictions (although we will be using it for decoding rather than generation). This image is borrowed from Ding et. al , 2012 [8].

To predict the secondary structures, the Viterbi algorithm is employed. This dynamic programming approach computes the most probable sequence of hidden states by:

1. Initializing the probabilities based on the starting states and the first observation.
2. Iteratively computing the probabilities of subsequent states while keeping track of the most likely transitions.
3. Backtracking through the computed probabilities to reconstruct the optimal sequence of states.

Why use an HMM for secondary structure prediction?

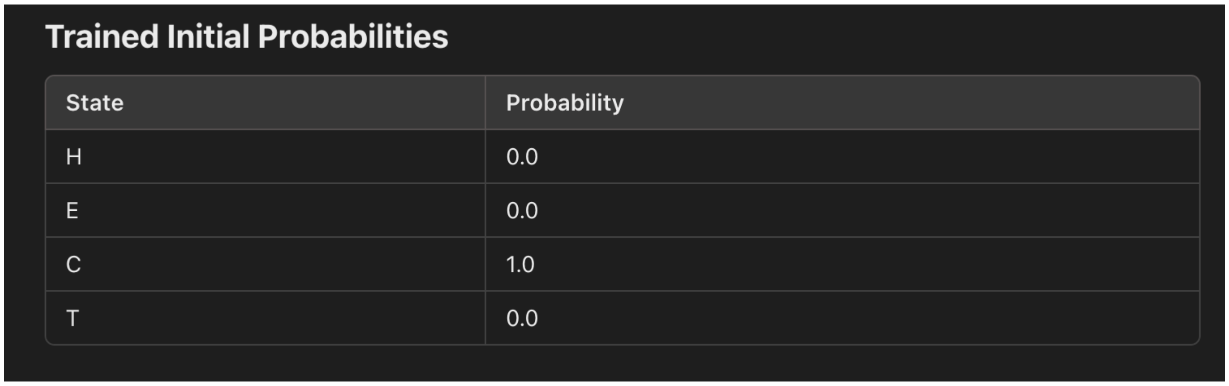
We chose to use a Hidden Markov Model (HMM) for predicting the secondary structure of proteins because it offered a unique approach compared to traditional methods like Chou-Fasman and GOR. We abstracted this model from what we learnt about markov chains in Math class and we found existing literature implementing this approach as well ([6], [7] and [8]). These conventional models employ a sliding window technique, analyzing the context of an entire segment of amino acids to make predictions. In contrast, an HMM focuses on the current amino acid residue and its immediately preceding state. This difference intrigued us, as we wanted to explore how an HMM could potentially uncover correlations between specific residues and their corresponding secondary structures. Additionally, we hypothesized that an HMM might inherently learn the transition rules between different structural states without explicit enforcement. While we haven't imposed any structural rules in our model yet, we believe that the HMM's ability to learn patterns from sequential data could lead to insightful discoveries in protein structure prediction.

Training the model: Justification for fine tuning the parameters

We chose a 4-state model for ease of implementation. We implemented the Expectation maximization algorithm to train the HMM rather than Baum Welch since we wanted to train the model on labelled data to identify the relationship between secondary structures and predicted amino acid residues.

The Expectation-Maximization (EM) algorithm implemented for training the Hidden Markov Model (HMM) is a powerful iterative approach designed to optimize the model parameters based on labeled sequences. At a high level, the algorithm operates by alternating between two main steps: the Expectation (E) step and the Maximization (M) step. In the E-step, the algorithm calculates expected counts of various events, such as state transitions and symbol emissions, using the provided labeled data. These counts are derived by analyzing how often each state and transition is observed in the training sequences. During the M-step, these expected counts are then used to update the HMM's parameters, including initial probabilities, transition probabilities, and emission probabilities, ensuring they reflect the observed data more accurately. This process repeats iteratively until convergence is achieved or until a predefined number of iterations is completed. The implementation also handles edge cases, such as zero counts, by assigning uniform probabilities to maintain numerical stability. Overall, this approach allows the HMM to learn from labeled sequence data and adjust its parameters to better model the underlying patterns in protein secondary structure prediction.

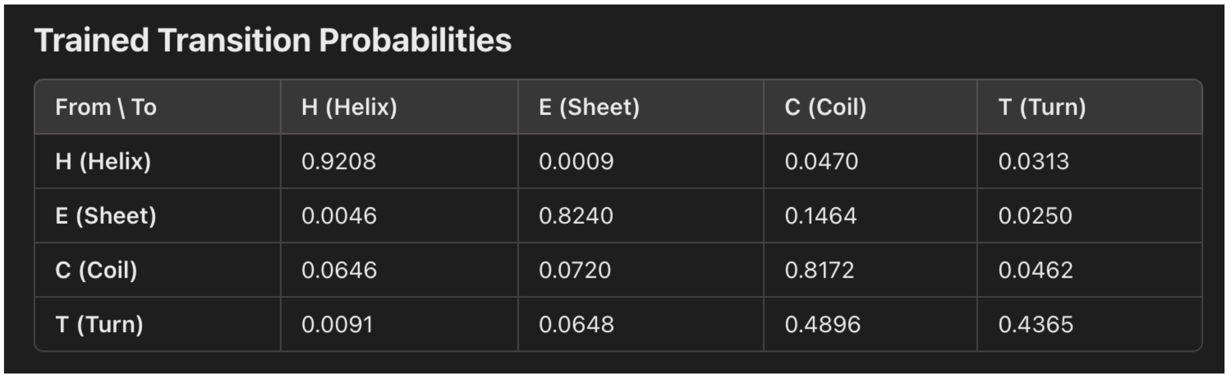
The values for the initial probability and transition probability we got from the EM training were as follows:



Trained Initial Probabilities

State	Probability
H	0.0
E	0.0
C	1.0
T	0.0

Table 1: Initial Probabilities learned from training are depicted in the figure above. In our dataset all the proteins started from C and that is reflected in the values learned from training.



Trained Transition Probabilities

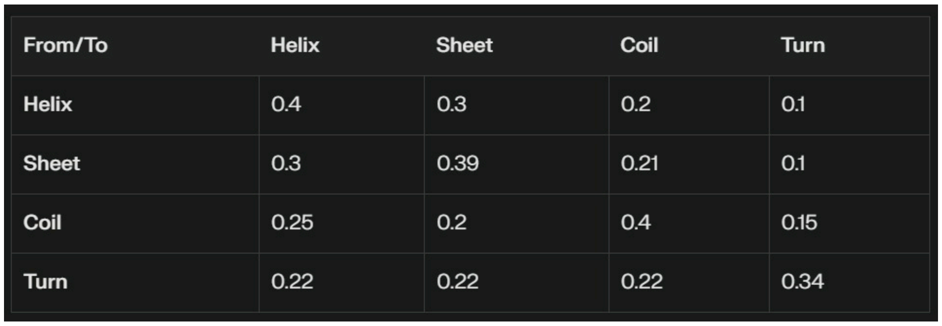
From \ To	H (Helix)	E (Sheet)	C (Coil)	T (Turn)
H (Helix)	0.9208	0.0009	0.0470	0.0313
E (Sheet)	0.0046	0.8240	0.1464	0.0250
C (Coil)	0.0646	0.0720	0.8172	0.0462
T (Turn)	0.0091	0.0648	0.4896	0.4365

Table 2: Transition Probabilities learned from training are depicted in the figure above.

The values we obtained from training were quite similar to those reported in the 3-state model of the PRT-HMM implemented by Ding et al., 2012 . However, these trained initial and transition probabilities were not directly used in our final model. Instead, we fine-tuned these parameters to produce more coherent and realistic predictions. During training, we observed that the model exhibited a strong bias towards self-transitions, often resulting in repetitive predictions such as long stretches of the same state (e.g., "HHHHH"). To address this issue, we drew inspiration from the transition probabilities used by Martin et al. (2006) and normalized the transition values to reduce this bias.

Fine-tuning the transition probabilities in a Hidden Markov Model (HMM) rather than using the trained values directly is a strategic decision aimed at enhancing the model's predictive accuracy and reliability. During the training phase, the model learns probabilities based on the given data, which can sometimes result in overfitting or an unrealistic bias towards certain transitions, such as excessive self-transitions. This can lead to repetitive predictions, such as long sequences of the same state, which do not accurately reflect biological reality. By fine-tuning these probabilities, we can introduce a level of normalization that reduces this bias while still maintaining the inherent preference for self-transitions where appropriate. This adjustment helps to create a more balanced model that is better suited for generalization across different datasets. Additionally, fine-tuning allows us to incorporate insights from previous research and empirical observations, such as reducing transition probabilities for less common states like turns, which might be underrepresented in training data. This process ensures that the HMM remains robust and adaptable, providing more coherent and realistic predictions of protein secondary structures. This also ensures that the emission probabilities learnt from the EM training can actually showcase the correlation between the amino acid residues and secondary structure predictions.

While self-transitions remained the most likely option, their dominance was tempered to allow for more varied state transitions. Additionally, the probabilities for turn transitions were significantly reduced, reflecting their sparse presence in both the training data and biological reality. Both the trained and fine-tuned parameters exhibited very low probabilities for turn transitions, which led to the HMM rarely predicting turns in practice.



From/To	Helix	Sheet	Coil	Turn
Helix	0.4	0.3	0.2	0.1
Sheet	0.3	0.39	0.21	0.1
Coil	0.25	0.2	0.4	0.15
Turn	0.22	0.22	0.22	0.34

Table 3: Transition Probabilities finally used in the HMM are depicted in the figure above.

For the emission probabilities we have directly used the values we learnt from EM training. Once again we found that the emission probabilities we learned from training were very similar to the ones obtained in the paper by Ding et. al, 2012 [8][1].

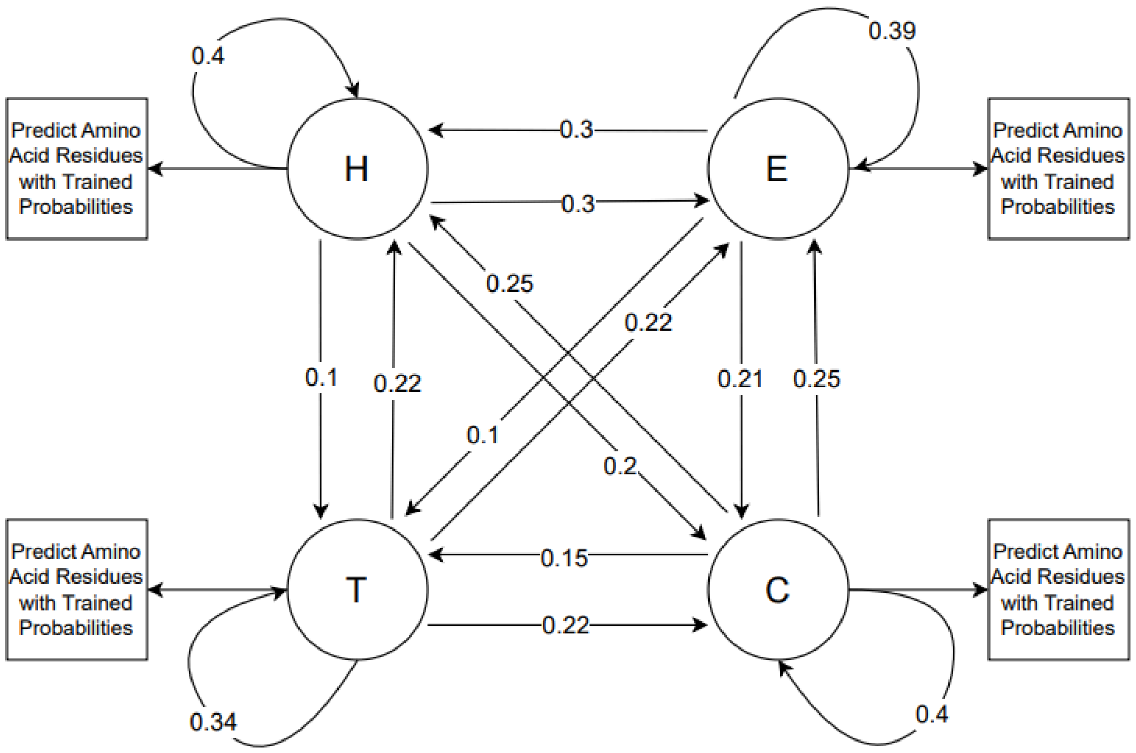

0.4
0.39
Predict Amino Acid Residues with Trained Probabilities
H
0.3
0.3
E
Predict Amino Acid Residues with Trained Probabilities
0.25
0.22
0.1
0.22
0.21
0.25
0.1
0.2
Predict Amino Acid Residues with Trained Probabilities
T
0.15
0.22
C
Predict Amino Acid Residues with Trained Probabilities
0.34
0.4

Figure 7: The state transition graph for our HMM model is depicted in the figure above.

Implementation

The HMM approach has been translated into a structured program through object-oriented principles and dynamic programming.

The HMM was implemented in Go using a modular design, where the core logic is encapsulated within an HMM struct. This struct contains:

1. State and symbol definitions.
2. Transition, emission, and initial probability matrices.
3. Functions for initialization and prediction.

[1] Our Emission Probabilities table can be found in the Appendix

The Viterbi algorithm is implemented as a method on the HMM struct. It processes a given sequence of amino acids and outputs the most probable sequence of secondary structures.
Additionally our implementation uses logarithmic probabilities for numerical stability, as multiplying many small probabilities can lead to underflow errors in floating-point arithmetic.

Input/Output: The program accepts a string of amino acid residues (e.g., "HAAAG") and outputs a string representing the predicted secondary structure (e.g., "HTHHC", where H = Helix, T = Turn, C = Coil).
The Viterbi algorithm predicts the most probable sequence of secondary structures for a given sequence of amino acid residues using a dynamic programming approach. It operates by considering possible paths through the states of the Hidden Markov Model (HMM) and selecting the sequence with the highest probability.

Viterbi Algorithm

1. Initialization: The algorithm starts by initializing the score matrix for the first amino acid in the sequence. Each state's score is computed using the initial probabilities and the emission probability of the first amino acid.
2. Recursion: For each subsequent amino acid in the sequence, the algorithm updates the score matrix by considering all possible previous states. It computes the maximum probability of transitioning from any previous state to the current state while emitting the current amino acid. These probabilities are stored in the score matrix. The backpointer matrix tracks the state that gave the maximum score.
3. Termination: After processing the entire sequence, the algorithm identifies the state with the highest probability in the last column of the score matrix. This state indicates the most likely final state in the sequence.
4. Backtracking: Using the backpointer matrix, the algorithm traces back from the last state to the first, reconstructing the most probable sequence of secondary structures.
5. Justification: The score matrix ensures that all possible paths are considered, while the backpointer matrix ensures that only the optimal path is traced back.
   The algorithm uses the transition and emission probabilities learned from training to make predictions based on the amino acid sequence. This process allows the Viterbi algorithm to efficiently find the most likely sequence of secondary structures, balancing both sequence-specific emissions and transitions between structural states.

Challenges

Implementing a HMM for predicting protein secondary structures presented several challenges, which are reflected in the Go code

1. **Efficient State Management:** Managing the dynamic programming matrices for large sequences required careful optimization to ensure the code remained efficient and comprehensible.
2. **Numerical Stability:** Implementing the algorithm in log-space required additional care to ensure that all computations (addition and multiplication) adhered to logarithmic rules.
3. **Data Constraints:** Real-world data often exhibit correlations that a simple HMM cannot capture. For instance, some secondary structures depend on broader sequence contexts beyond adjacent residues.

Limitations

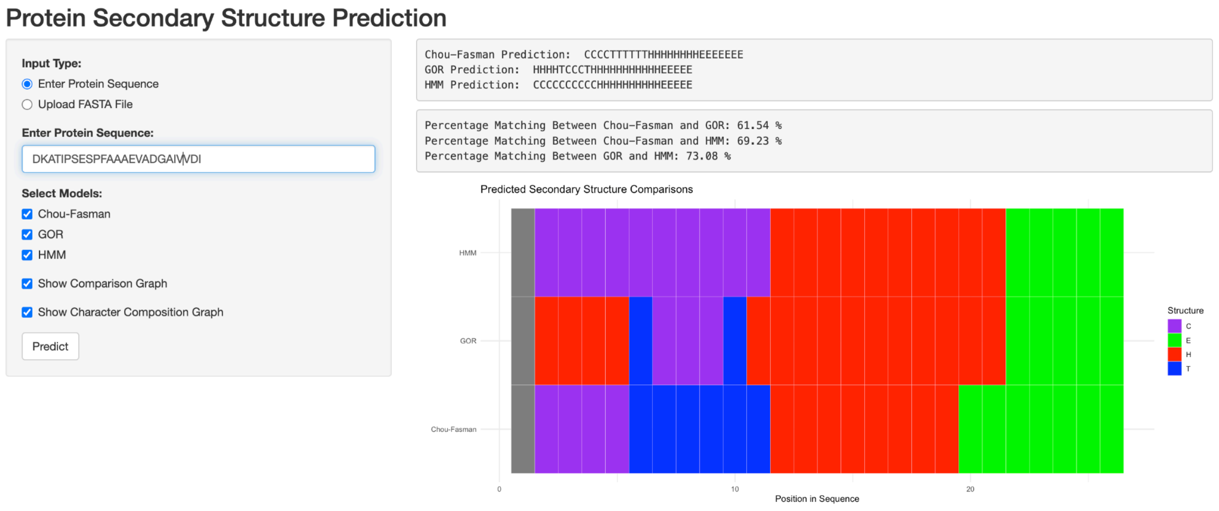
The current implementation has several limitations:

1. **Single-Residue Emission:** The model considers each amino acid independently for emission probabilities. This simplification ignores correlations between adjacent residues, which could improve prediction accuracy. Extending the model to use pairs or triplets of amino acids would increase the "codebook size" (e.g., from 20 to 400 for pairs) but also require significantly more data and computational resources.
2. **Lack of Structural Coherence:** The HMM predicts secondary structures independently of global constraints, which may lead to biologically implausible predictions. For instance, it does not enforce compatibility between successive states or ensure structural coherence across the protein sequence.
3. **Model Parameters:** Training the HMM using Baum-Welch might allow us to use various latent models within each state rather than just a 4-state model as we have right now. The model was currently trained by Expectation Maximization, in the future we can also train using the Baum-Welch algorithm.
4. **Turn Predictions:** The HMM rarely predicts turns, even though they are a defined state. This occurs because the transition probabilities to turns are very low, reflecting their scarcity in the training data. This limitation could be addressed by developing a more refined model in the future.

User Interface

Implementation

A Shiny app was developed to accept user inputs in the form of protein sequences or FASTA files, generating predicted structures for the three models: Chou-Fasman, GOR and HMM. It compares match percentages across these sequences and provides a visual representation of the comparison. Additionally, the app displays the percentage composition of each structural character (H, E, C, T) for each of the models.


Protein Secondary Structure Prediction

Input Type:
- [✓ Enter Protein Sequence]
- Upload FASTA File

Enter Protein Sequence: DKATIPSESPFAAAEVADGAIVVDI

Select Models:
- [✓ Chou-Fasman]
- [✓ GOR]
- [✓ HMM]
- [✓ Show Comparison Graph]
- [✓ Show Character Composition Graph]

```
Chou-Fasman Prediction:  CCCCTTTTTTHHHHHHHHEEEEEEE
GOR Prediction:  HHHHTCCCTHHHHHHHHHHHEEEEE
HMM Prediction:  CCCCCCCCCCHHHHHHHHHHEEEEE
```

```
Percentage Matching Between Chou-Fasman and GOR: 61.54 %
Percentage Matching Between Chou-Fasman and HMM: 69.23 %
Percentage Matching Between GOR and HMM: 73.08 %
```

Figure 8: RShiny User Interface, showing the input protein sequence, and the outputs. The outputs include the predicted sequences from all three models, the percentage of matching characters between them, and a visual comparison of the matching percentage.

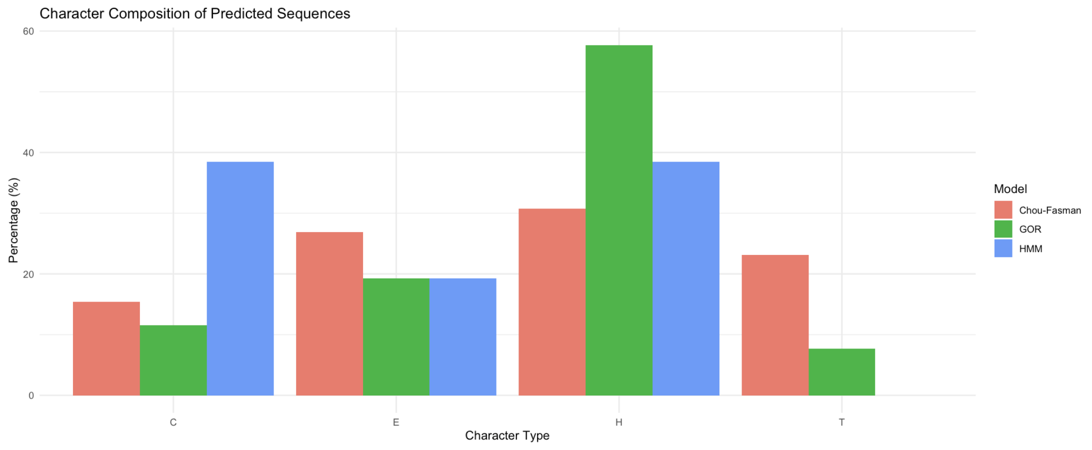


Figure 9: RShiny User Interface, showing the character composition (percentage of H, E, C and Ts) of all three predicted sequences.

Secondary Structure Validation with DSSP

Metrics and Method

DSSP (Dictionary of Secondary Structure in Proteins) is a widely used algorithm which was developed by Wolfgang Kabsch and Chris Sander. It is designed to assign secondary structure elements to the amino acids of a protein based on its atomic-resolution coordinates. The algorithm identifies intra-backbone hydrogen bonds using an electrostatic model, which helps in determining the secondary structures such as helices, sheets, and loops.

In our project, we used the SecNet 2018 test dataset to validate our outputs. The SecNet 2018 test dataset is a collection of protein structures that were determined and released in 2018. This dataset was specifically designed to evaluate secondary structure prediction models, such as SecNet, by using DSSP-derived information. The dataset uses DSSP to assign secondary structure elements to the protein chains included in the dataset. DSSP provides the annotations of secondary structures and categorizes them into eight types (DSSP8) or reduces them to four broader classes (DSSP4). Since our outputs for our 3 algorithms classified the secondary structures into 4 classes, we used the DSSP4 category[5].

Unlike predictive models, DSSP does not predict secondary structures from primary amino acid sequences; instead, it extracts this information directly from the 3D coordinates of proteins. This makes DSSP an invaluable tool for validating secondary structure prediction models, such as our own, as it provides a reliable reference for comparing our predicted structures against experimentally determined ones. By using DSSP data, we can assess the accuracy and effectiveness of our three computational models designed to predict protein secondary structures from amino acid sequences.

We illustrate the metrics and methods used in our validation of our model outputs against gold standard DSSP sequences via an example computation using simple dummy amino-acid sequences below.

Structure Types Considered:

- **H**: Helix
- **E:** Beta-sheet
- **C:** Coil
- **T:** Turn

Ground Truth sequence from DSSP database: H E **T** T C **C** E E
(Indices: 1 2 3 4 5 6 7 8)

Model Prediction (example, from GOR): H E **E** T C **H** E E
(Indices: 1 2 3 4 5 6 7 8)

Visually, we can easily identify that there are two positions in these sequences that differ; however, longer sequences require additional metrics to quantify differences more effectively. The two sequences are compared head-to-head in the table below.

Position	Ground Truth	Model Output	Correct?
1	H	H	Yes
2	E	E	Yes
3	T	E	No
4	T	T	Yes
5	C	C	Yes
6	C	H	No
7	E	E	Yes
8	E	E	Yes

Table 5: Table to explain how "matches" are decided.

We calculated percentage Accuracy, Precision, Recall, and F1-Score values, described in the table below, to robustly validate our models' outputs with the gold standard sequence outputs.

Metric	Formula	Conveys	Ideal When
Accuracy	$\left(\frac{\text{Correct Predictions}}{\text{Total Predictions}}\right)\times 100\%$	Overall correctness of the model's predictions across all structures (how often a model is correct).	Structures are balanced (one structure isn't the most frequent)
Precision	$\frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}\times 100\%$	Correctness of positive predictions for a specific structure (i.e. a Helix is a Helix).	Minimizing false positives is crucial.
Recall	$\frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}\times 100\%$	Ability to identify all actual positives for a specific class (how many of 30 T's in the standard sequence did the model correctly identify?)	Minimizing false negatives is crucial.
F1-Score	$2 \times \left(\frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}\right)\times 100\%$	Balance between precision and recall for a specific class (balances identifying all positive instances and that those instances are correct)	When both false positives and negatives matter.

Table 6: Metrics for model performance statistics and their significance

Using the metrics defined above, the model correctly predicted six out of eight structures with a 75% accuracy when comparing the example model sequence output with the ground truth output.

$$Accuracy = \left(\frac{6}{8}\right) \times 100\% = 75\%$$

Unlike accuracy, the next three metrics are applied to gauge the model performance per structure. The model correctly predicted a helix (H) at position 1 (True Positive), incorrectly predicted an H at position 6 instead of the actual C (False Positive), and there were no positions that were H but were predicted as another structure (False Negative). This presents the following metrics for a helix.

$$Precision(H) = \frac{1}{1+1} \times 100 = 50\%$$
$$Recall(H) = \frac{1}{1+0} \times 100 = 100\%$$
$$F1\,Score(H) = \frac{1}{1+1} \times 100 \approx 66.67\%$$

Note that F1-Scores are usually reported as fractions between 0 to 1 but we reported them here in percentages for consistency with the other metrics. We calculated these hypothetical metrics similarly for the remaining three structures as shown in the summary table below.

Structure Type	Precision (%)	Recall (%)	F1-Score (%)
H	50	100	66.67
E	75	100	85.71
C	100	50	66.67
T	100	50	66.67

Table 7: Precision, Recall, and F1-Score metrics for each structure type in the example above.

For a helix, 50% precision indicates that the model predicts a helix 50% of the time; this may indicate a high false positive rate. Secondly, a recall of 100% indicates that all actual helices are identified correctly. Lastly, an F1-Score balances the previous two metrics and indicates a low precision with perfect recall. These metrics are calculated by our code in R for a sequence of any length and are depicted visually using bar graphs in our RShiny Validation Application interface.

Validation Results

In this section, we present the actual validation results for the actual outputs from our three models when compared to the gold standard DSSP database's "ground truth" sequence structure strings. We sampled 50 protein sequences from the DSSP database and extracted their corresponding "ground-truth" structures. Using the calculations illustrated and metrics described in the examples above we validate our models' performance. The graphs below are generated using our RShiny Validation Application.

Figure 10 below plots the Accuracy of the three models for each of the 50 protein amino acid sequences from the dataset. The figure conveys the average accuracies of approximately 35.02 $\pm$ 9.69% for CF, 41.19 $\pm$ 10.63% for GOR, and 47.29 $\pm$ 17.32% percent for HMM models. These percentages indicate the number of structures in a sequence that the model correctly predicts when compared to the ground truth value. While GOR shows a greater average accuracy than CF, they both display comparable standard deviations indicating a similar spread in the predictions of the two models. The HMM model shows the highest prediction accuracy but the accompanying higher standard deviation indicates the large variance in the model's predictions. The higher variance observed in HMM predictions may result from its complexity and sensitivity to variations in data, whereas CF and GOR models are simpler and more consistent but less accurate overall. Hidden Markov Models rely on probabilistic transitions and emissions to predict sequences, which means that small changes in the input data or training set can lead to different outcomes. This variability is particularly pronounced when the training data is sparse or unbalanced, as is often the case with secondary structure prediction, where certain states like turns are underrepresented. As a result, while HMMs are powerful tools for capturing patterns in sequential data, their predictions can exhibit higher variance, especially in cases where the training data does not adequately represent all possible scenarios.

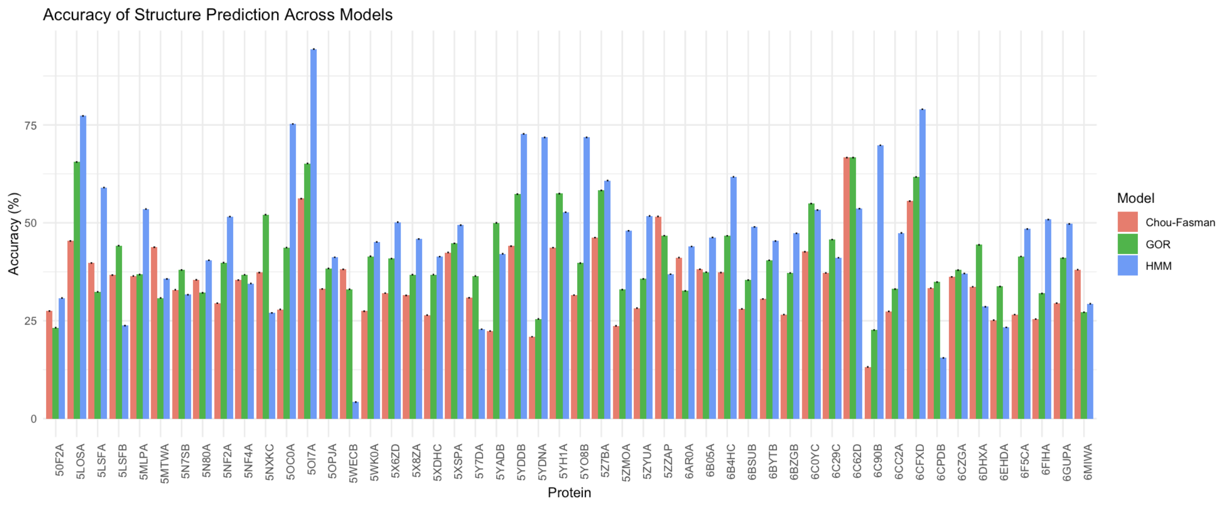
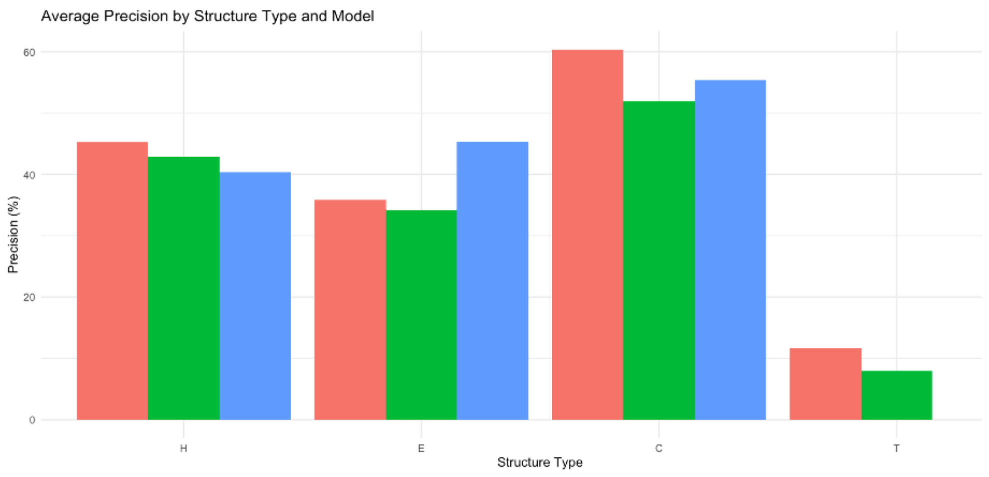


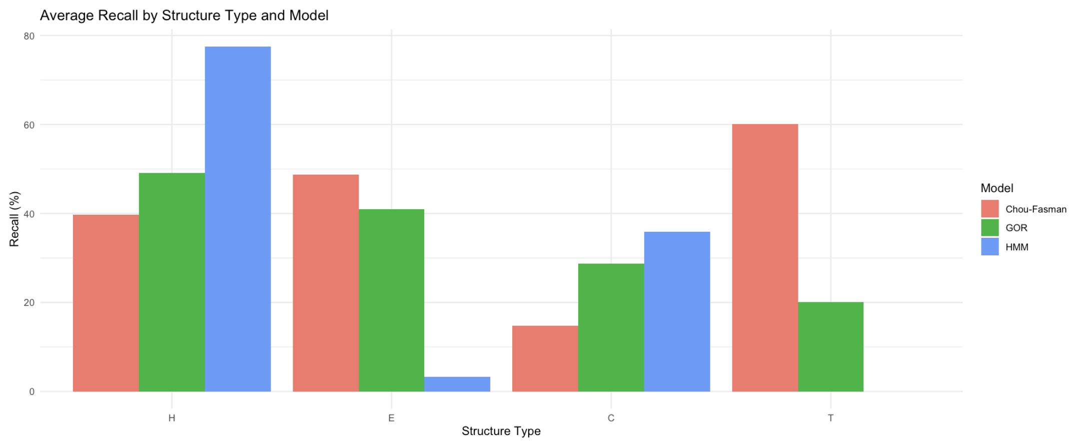
Figure 10: Bar graph showing the Accuracy(in %) of the three structure prediction models across 50 protein sequences.

Figures 11a., 11b., and 11c., below plot the per-structure metrics of precision, recall, and F1-Scores. All three models show comparable precision values for predicting helices (H), beta sheets (E), and coils (C). Most notably, the three models are better at correctly predicting coils, all with greater than 50% precision, than compared to other structures. Conversely, CF and GOR are poor at predicting turns (T) whereas the HMM model fails at predicting turns. Low to non-existent precision for turns indicates that the models have a high false positive rate for predicting turns and that many of their positive predictions are incorrect for turns. The difficulty in predicting turns by the HMM model is a result of the low probability assigned to turn transitions which is a consequence of the low representation of turns in training data, leading to poor model performance on this structure type.

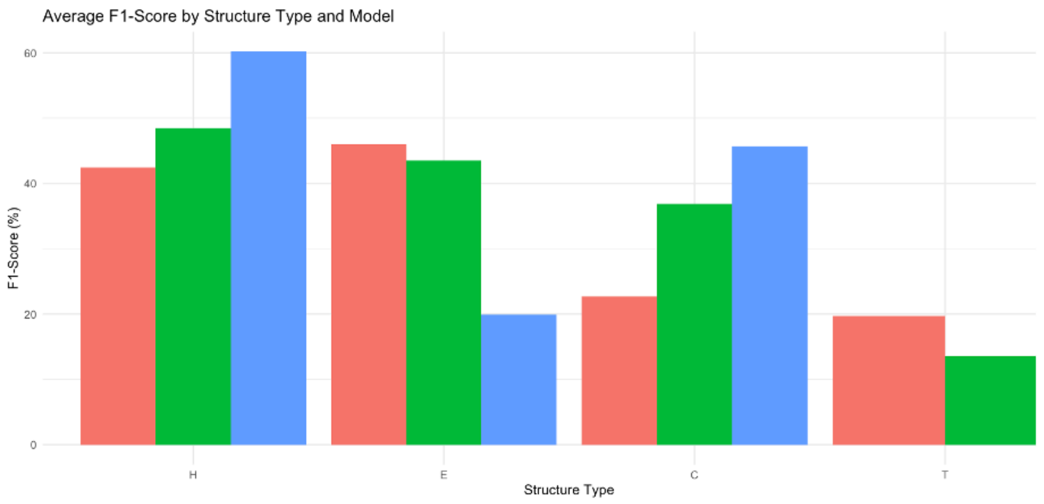
The helix recall value is the highest for the HMM model, at about 79% indicating a low false negative rate and that this model correctly predicts helices about 79% of the time. This high recall value combined with the observation of a low recall for beta sheets and nonexistent precision and recall values for turns may also suggest that the HMM model preferentially predicts helices. The CF and GOR models have low but expected and similar recall values except for the turns wherein CF greatly outperforms GOR. The F1-Scores harmonize the precision and recall observations into one statistic. A low F1-Score may convey that a model struggles in either precision (high false positives) or recall (high false negatives) or both; the ideal situation arises when the F1-Score is as close to 100% as possible. The preference of HMM for helices may be due to a higher frequency of helices in training data, while CF's better performance on turns could reflect its simpler rule-based approach capturing certain turn-specific features better than GOR or HMM.



a.



b.



c.

Figure 11: **a.** Bar graph depicting the average precisions of predicting H, E, C and Ts across 3 models for 50 proteins. **b.** Bar graph showing the average recall for each structure type, across 50 proteins. **c.** Bar graph depicting the average F-1 scores of each structure type across the three models for 50 proteins.

Figure 12 below plots the percent accuracy values for each model for each of the 50 proteins from the DSSP dataset against each protein's respective length of residues. This figure establishes the Law of Large Numbers wherein longer length residues convey an accuracy closer to the average accuracy of a particular model resulting in a seemingly Gaussian distribution. This trend is mostly because longer sequences provide more data points for statistical averaging, hence reducing the variability in accuracy measurements across different proteins compared to shorter sequences where any random fluctuations have a larger impact on results.

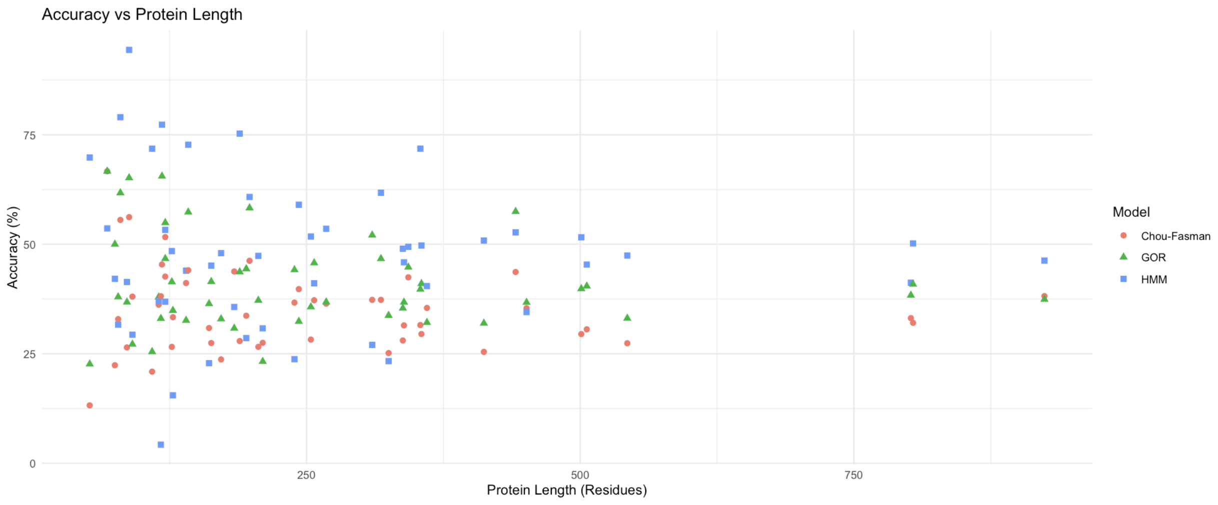


Figure 12: Scatterplot of the average accuracy of each model plotted as a function of the protein length.

Conclusion

In this study, we evaluated three classical models—Chou-Fasman (CF), Garnier-Osguthorpe-Robson (GOR), and Hidden Markov Model (HMM)—for protein secondary structure prediction. By comparing the CF, GOR, and HMM methods, we have demonstrated the varying degrees of accuracy and limitations inherent in each approach. The Chou-Fasman model, while pioneering, shows lower accuracy due to its reliance on static propensity values (from a small dataset) without considering sequence context. In contrast, the GOR method leverages local sequence context through statistical analysis, resulting in improved prediction accuracy. However, its dependence on local interactions limits its ability to account for long-range structural influences. The experimental HMM approach, promising due to its probabilistic framework, surprisingly outperformed both previous models. However, it exhibited high variance, reflecting the need for parameter optimization and training on larger datasets. Future directions include implementing the advanced versions of CF and GOR models, and refining the HMM with advanced training algorithms like Baum-Welch. All of the models could also be enhanced by integrating modern machine learning techniques to improve accuracy and generalizability.

Despite these challenges, the exploration of these classical algorithms provides valuable insights into the progression of computational biology and sets a benchmark for evaluating more advanced machine learning models like AlphaFold2. The field has seen remarkable advancements with machine learning models achieving up to 84% accuracy in secondary structure prediction, while AlphaFold by DeepMind has revolutionized tertiary structure prediction with an astonishing 99% accuracy, setting a benchmark for AI-driven approaches. Emerging trends include using attention-based architectures and large language models (LLMs), which offer faster and potentially more accurate predictions. Hence, this project emphasizes the importance of integrating historical methods with contemporary technologies to enhance our understanding and prediction of protein structures.

Citations

1. Prevelige, P., & Fasman, G. D. (1989). Chou-Fasman Prediction of the secondary structure of proteins. Prediction of Protein Structure and the Principles of Protein Conformation, 391–416. https://doi.org/10.1007/978-1-4613-1571-1_9

2. Garnier, J., Osguthorpe, D. J., & Robson, B. (1978). Analysis of the accuracy and implications of simple methods for predicting the secondary structure of globular proteins. *Journal of Molecular Biology*, 120(1), 97–120.

3. Garnier, J., Gibrat, J. F., & Robson, B. (1996). GOR method for predicting protein secondary structure from amino acid sequence. *Methods in Enzymology*, 266, 540–553.

4. Gibrat, J. F., Garnier, J., & Robson, B. (1987). Further developments of protein secondary structure prediction using information theory. *Journal of Molecular Biology*, 198(3), 425–443.

5. Anfinsen CB. Principles that govern the folding of protein chains. Science. 1973 Jul 20;181(4096):223-30. doi: 10.1126/science.181.4096.223. PMID: 4124164.

6. Kabsch, W. and Sander, C. (1983) ‘Dictionary of protein secondary structure: Pattern recognition of hydrogen‐bonded and geometrical features’, Biopolymers, 22(12), pp. 2577–2637. doi:10.1002/bip.360221211.

7. Asai K, Hayamizu S, Handa K. Prediction of protein secondary structure by the hidden Markov model. Comput Appl Biosci. 1993 Apr;9(2):141-6. doi: 10.1093/bioinformatics/9.2.141. PMID: 8481815.

8. Martin, J., Gibrat, JF. & Rodolphe, F. Analysis of an optimal hidden Markov model for secondary structure prediction. BMC Struct Biol 6, 25 (2006). https://doi.org/10.1186/1472-6807-6-25

9. Ding, Wang et al. "PRT-HMM: A Novel Hidden Markov Model for Protein Secondary Structure Prediction." 2012 IEEE/ACIS 11th International Conference on Computer and Information Science (2012): 207-212.

10. Potter, J., Lutsky, A., Nandakumar, R., & Katiyar, S. (2023). *Predicting secondary structure protein folding*. Pittsburgh, PA USA.

11. Kendrew, J. C., Bodo, G., Dintzis, H. M., Parrish, R. G., Wyckoff, H., & Phillips, D. C. (1958). A three-dimensional model of the myoglobin molecule obtained by X-ray analysis. *Nature*, *181*(4610), 662–666. https://doi.org/10.1038/181662a0

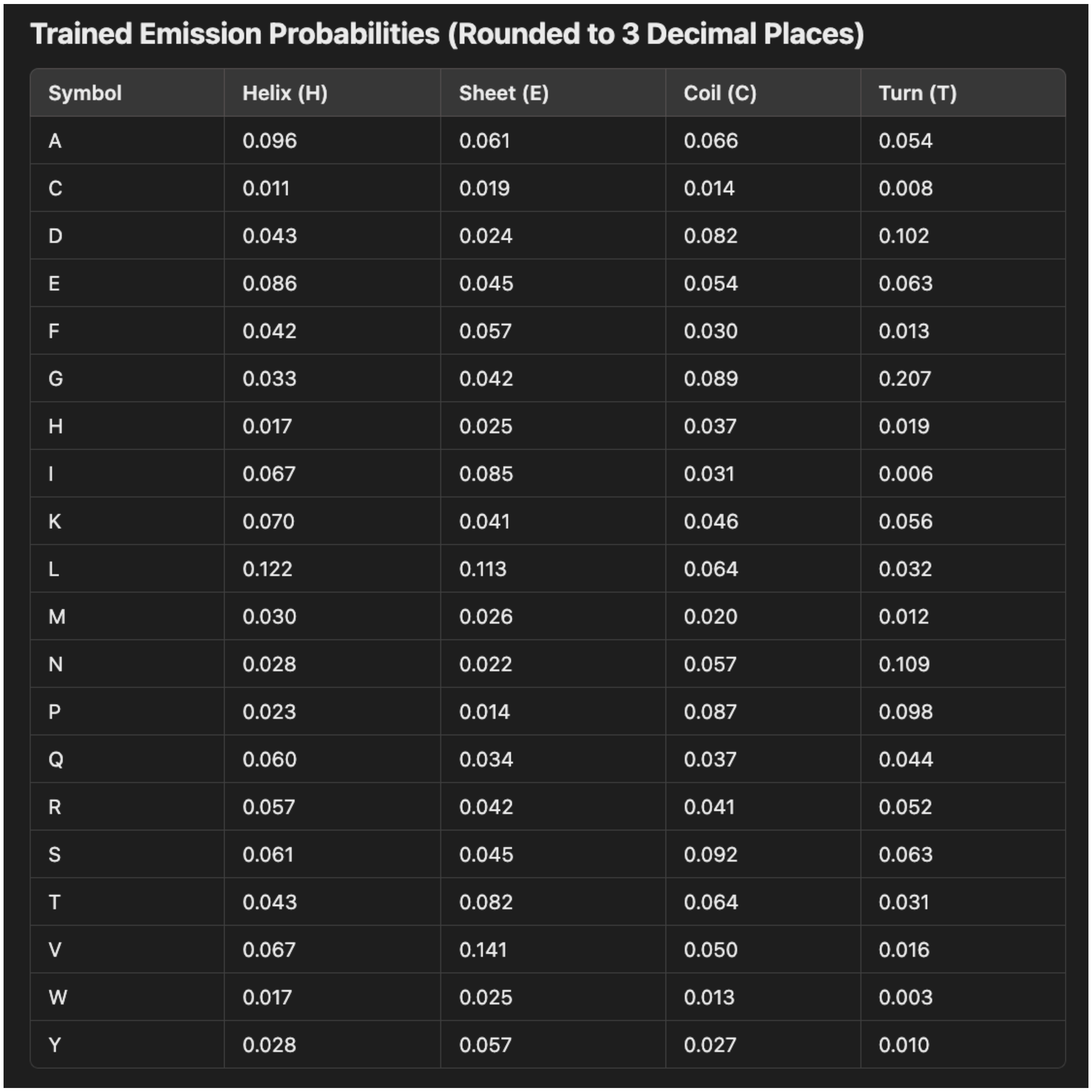
12. McDonough, M. (2024, November 22). *Did ai solve the protein-folding problem?*. Harvard Medicine Magazine. https://magazine.hms.harvard.edu/articles/did-ai-solve-protein-folding-problem

13. Chou, P. Y., & Fasman, G. D. (1978). Empirical predictions of protein conformation. *Annual Review of Biochemistry*, *47*(1), 251–276. https://doi.org/10.1146/annurev.bi.47.070178.001343

Contributions

Arth Banka: Worked on the HMM code from research, to implementation and testing functions
Riti Bhatia: Worked on the Chou-Fasman code from research to implementation and testing functions
Sanchitha Kuthethoor: Worked on the visualizations using RShiny- the user interface as well as the validation app. She also worked on validation of the three models against DSSP predictions.
Sumeet Kothare: Worked on the GOR code from research to implementation and testing. Sumeet also worked with Sanchitha on the visualizations and validation.

Appendix



Trained Emission Probabilities (Rounded to 3 Decimal Places)

Symbol	Helix (H)	Sheet (E)	Coil (C)	Turn (T)
A	0.096	0.061	0.066	0.054
C	0.011	0.019	0.014	0.008
D	0.043	0.024	0.082	0.102
E	0.086	0.045	0.054	0.063
F	0.042	0.057	0.030	0.013
G	0.033	0.042	0.089	0.207
H	0.017	0.025	0.037	0.019
I	0.067	0.085	0.031	0.006
K	0.070	0.041	0.046	0.056
L	0.122	0.113	0.064	0.032
M	0.030	0.026	0.020	0.012
N	0.028	0.022	0.057	0.109
P	0.023	0.014	0.087	0.098
Q	0.060	0.034	0.037	0.044
R	0.057	0.042	0.041	0.052
S	0.061	0.045	0.092	0.063
T	0.043	0.082	0.064	0.031
V	0.067	0.141	0.050	0.016
W	0.017	0.025	0.013	0.003
Y	0.028	0.057	0.027	0.010

Table: Emission Probabilities learned from training and finally used in the HMM are depicted in the figure above.