1 Introduction

With the recent advances in ab initio protein folding, the problem of predicting protein 3D structure solely from its amino acid sequence is close to being solved [1], [2]. Yet, the problem of predicting multimeric protein complexes, that is the quaternary 3D structure of proteins consisting of several peptide chains, remains.

1.1 From predicting single protein structure to protein complexes

In 2018, DeepMind unveiled AlphaFold which was able to predict protein 3D structure with high accuracy [3]. In 2020 a new improved version of AlphaFold, called AlphaFold2, was shown to predict protein structures comparable to that of experimentally determined structures [2], [4].

AlphaFold2 utilizes a deep learning algorithm, divided into two stages. In the first stage multiple sequence alignments are made to infer spatial and evolutionary relationships between amino acids, while the second stage uses that information to predict the 3D structure [2].

Building on top of Alphafold2, AlphaFold-Multimer solves the folding of several proteins to form multimeric protein complexes. AlphaFold-Multimer follows the same principles as in AlphaFold but is optimized for and trained on protein structure complexes [5].

1.2 Antibodies – an important multimeric protein

An important multimeric protein in humans are the antibodies created by B-cells [6], [7]. Antibodies consist of several peptides which comes together to form a functioning unit. The variable chains, called the heavy and light chains, makes the antibody specific for an antigen [7].

1.3 Prediction of variable chains in antibodies

Recently, Abanades, B. et al. has presented a new set of deep learning models called ImmuneBuilder, where one of the models is ABodyBuilder2. This model has been trained to predict the structures and relative position of heavy and light chains given only their amino acid sequences. ABodyBuilder2 predicts the complexes of heavy and light chains through generating an ensemble of 4 possible structures, using 4 different deep learning models, and subsequently refining the model closets to the average of the 4 models [8], see Figure 1.

2 Materials and methods

2.1 Dataset

124 structures were picked from the protein data bank using the SAbDab database [9] and the following parameters: experimental method set to “X-RAY DIFFRACTION”, species set to “HOMO SAPIENS”, resolution cutoff set to “2.0”, R-factor cutoff set to "0.20”, and antigen type set to “Protein”, and all other parameters set as the default values.

2.2 Implementation

The implementation was made in Python 3.7 using the packages: pandas ver. 1.3.2, biopython ver. 1.18, NumPy ver. 1.21.6, PyTorch ver. 2.1.2, OpenMM ver. 7.7.0, PDBFixer ver 1.8.1, ANARCI ver. 1.3, HMMER ver. 3.3.2, pymol-open-source ver. 2.5.0 and ImmuneBuilder ver. 1.0.1.

PDB files were downloaded for all antibodies using a custom function with Bio.PDB, see section 7.1, and sequences of heavy and light variable domains were extracted using a custom function with Bio.PDB, see section 7.2.

Predictions of the variable domains were following done using the deep-learning model ABodyBuilder2 from the ImmuneBuilder package, see section 7.3.

2.3 Superposition and RMSD

Superposition of the predicted structure of the heavy and light variable domains onto the x-ray diffraction derived structures were done using PyMol and the RMSD of the superposition was extracted directly from here, see section 7.4. Depictions were made in PyMol using the script in section 7.5.

3 Results

3.1 Structure prediction

124 structures were downloaded, but only 73 structures contained chains named “H” and “L” for heavy and light chain, respectively. These 73 structures were run through the ABodyBuilder2 model and 71 were predicted a 3D structure. 2 failed due to the model not recognizing the chain’s sequence as that of a heavy or light chain.

3.2 Superposition and RMSD

A selection of the structures is depicted in Figure 2, with PDB id and RMSD for the superposition of the predicted heavy and light variable domains onto the experimentally determined structure. The mean (± S.E.M) RMSD of all superpositions were $0.503\ \text{Å}\ \pm 0.193$. See Table 1 for RMSD of each superposition.

4 Discussion

From the analysis, the low RMSD between the x-ray diffraction determined structure and the predicted structure, shows that ABodyBuilder2 is good at predicting the 3D structure of the variable chains in antibodies (protein multimeric complex) – at least for antibodies against protein antigens.

4.1 Strengths and limitations

The primary limitation of the ABodyBuilder2 model is the limited dataset of 3829 antibody structures used to train the model [8]. A bigger dataset would probably be needed to improve accuracy of the model.

But, with the results shown in this implementation and the results from the publication, the model is a big improvement in the field of antibody prediction, compared to earlier models. Though, the model is made to specifically handle proteins of the immune system rather than all types of proteins in biology.

Further, to comment on the result of this implementation, some of the PDBs used was also used in training the model, thus the results should be taken with a grain of salt.

4.2 Perspective

Prediction of antibody structures can potentially be the answer to create antibodies against currently uncurable diseases such as prion diseases. Prion diseases are characterized by the formation of protein aggregates in the brain, resulting from protein misfolding [10]. Thus, by creating variable chains specific for the misfolded protein, the immune system should be able to detect these proteins selectively and degrade them.

5 Conclusion

ABodyBuilder2 can predict the structure of variable heavy and light chains for antibodies specific for protein antigens to a degree comparable to experimentally determined structures.

6 Figures and tables

6.1 The principle behind ABodyBuilder2

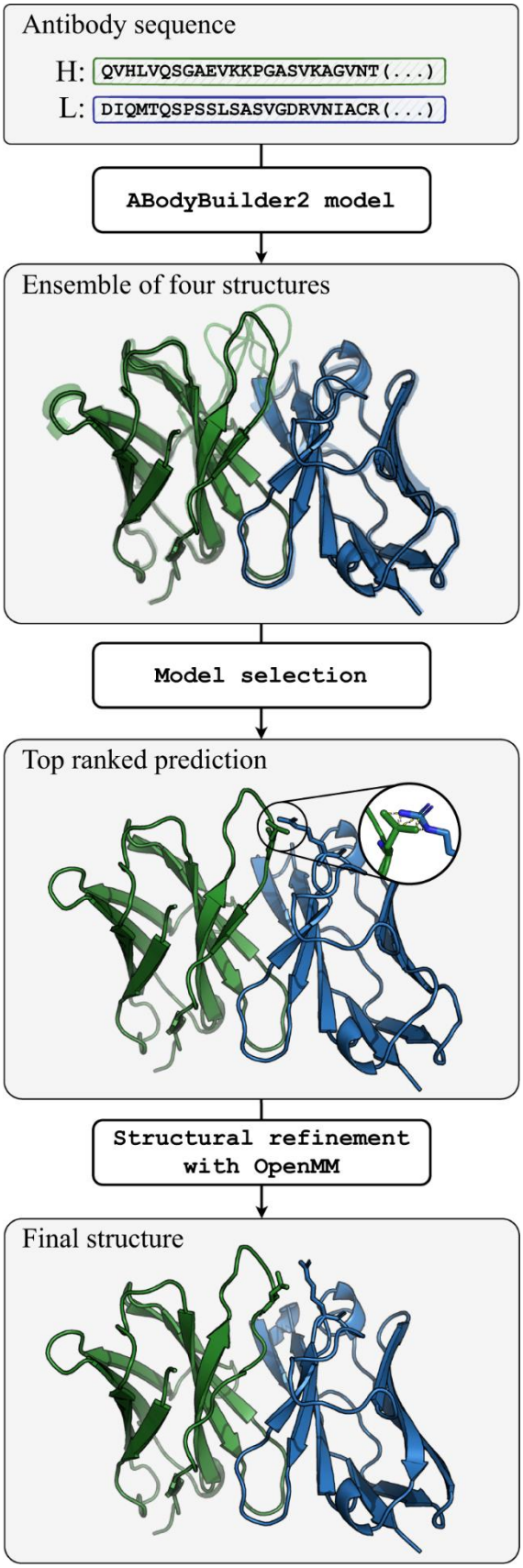

Figure 1 Model of the framework employed by ABodyBuilder2. The sequences of the variable heavy and light chains are used as input to the ABodyBuilder2 model, which parses the input into four deep learning models that creates an ensemble of four predicted protein structure complexes. The structure closets to the average of the ensemble is chosen and refined using OpenMM, which removes positions that are physically not possible (the steric repulsion principle) to yield the final structure. Figure from Abanades, B. et al. [8]

6.2 Representation of superposition

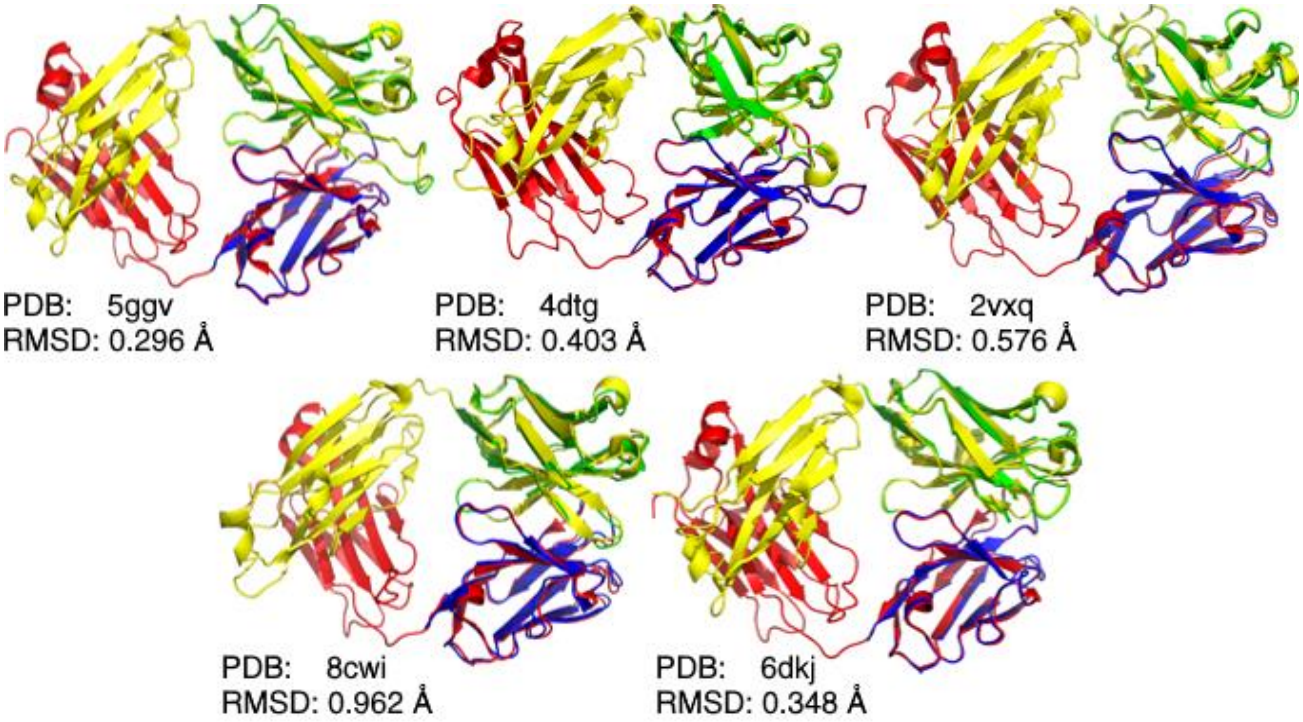

Figure 2 A selection of antibodies depicted using PyMol [11] with only their heavy and light chains depicted. Prediction of heavy and light chain for each heavy and light chain's variable region is superpositioned and RMSD for the superposition is shown. Yellow and red colored chains are the heavy and light chains, respectively, from the crystal structure, and the green and blue colored chains are the heavy and light chain, respectively, predicted using ABodyBuilder2 [8].

6.3 RMSD of superpositions

PDB_ID	RMSD	PDB_ID	RMSD	PDB_ID	RMSD	PDB_ID	RMSD	PDB_ID	RMSD
2uzi	0.423	4m62	0.432	5uea	0.342	7e9b	0.345	7sd5	0.834
2vxq	0.576	4nzr	0.605	5uek	0.306	7fcq	0.686	7so5	0.476
3l5x	0.368	4tsb	0.337	5vkd	0.508	7jmp	0.323	7str	0.755
3mxw	0.599	4xmp	0.611	6b0s	0.349	7kql	0.518	7vux	0.453
3p0y	0.626	4xvs	0.439	6cbv	0.374	7lfb	0.651	7wsl	0.399
3se8	0.581	4xvt	0.404	6dkj	0.348	7lm8	0.527	7z0x	0.713
3se9	0.714	4ydl	0.386	6iea	0.543	7lm9	0.393	8ahn	0.418
3sob	0.451	5ggv	0.296	6meh	0.470	7lsg	0.465	8bse	0.430
3u2s	0.425	5hi4	0.333	6mvl	0.479	7mzg	0.923	8cwi	0.962
4al8	0.530	5l6y	0.410	6svl	0.345	7mzi	0.673	8dfi	0.908
4dtg	0.404	5n7w	0.519	6vy4	0.646	7n3c	0.624	8ffe	0.432
4fqi	0.462	5ngv	0.353	6was	0.398	7n3d	0.363		
4h8w	0.303	5ob5	0.349	7amr	0.848	7neh	0.526		
4i77	0.360	5sy8	0.368	7b3o	0.628	7q0g	0.541		
4j6r	0.624	5ucb	0.428	7bz5	0.320	7rxp	0.751		

Table 1 RMSD of superpositioning predicted heavy and light chain onto the x-ray diffraction determined structure.

7 Code snippets

7.1 Implementation of downloading PDB files

```
def get_pdb_file(identifier, overwrite_files=False):
    """
    Download PDB file from the protein data bank and save it in structures folder as
    <identifier>.pdb

    Args:
        identifier (Str): 4-letter PDB identifier
        overwrite_files (Bool): Overwrite already downloaded structure, default=False

    Returns:
        None
    """

    # Check if structures folder exist
    if not os.path.isdir("./structures/"):
        os.mkdir("./structures/")

    # Check if file already exist
    if os.path.isfile("./structures/" + identifier + ".pdb") and not overwrite_files:
        return None

    # Else-if PDB file already exists, remove this file
    elif os.path.isfile("./structures/" + identifier + ".pdb") and overwrite_files:
        os.remove("./structures/" + identifier + ".pdb")

    # Instantiate PDBList
    pdbl = PDBList()

    # Ask BioPDB.PDBList to get PDB file
    pdbl.retrieve_pdb_file(identifier,
                           file_format="pdb",
                           pdir="./structures/",
                           overwrite=overwrite_files)

    # Rename the downloaded file
    os.rename("./structures/pdb" + identifier + ".ent", "./structures/" + identifier +
                                                                          ".pdb")

    # Return None
    return None
```

7.2 Implementation of extracting chain sequences using Bio.PDB

```
def get_chain_from_pdb(pdb_file, chain_names, choosen_model=0):
    """
    Extract a specific chain's sequence from a PDB file

    Args:
        pdb_file (Str): Name and location of PDB file
        chain_names (List): List chain(s) name(s) to extract
        choosen_model (Int): Which model number to get chain(s) from

    Returns:
        Dictionary of sequences for chains
    """

    # Get the PDB id from filename
    pdb_id = os.path.basename(pdb_file)

    # Instanciate PDBParser
    parser = PDBParser(QUIET=True)

    # Get structure and specified model of protein from PDB file
    structure = parser.get_structure(pdb_id, filename)
    model = structure[choosen_model]

    # Dictionary to store found chain
    chain_sequences = {}

    # Loop through the list of specified chain names
    for chain_name in chain_names:

        # List of found residues in current chain
        residues = []

        # Find all residue names in chain
        try:
            for residue in model[chain_name]:

                # Get the name of the current residue
                res_name = residue.get_resname()

                # If the residue is an amino acid, we add it to the list of residues
                # for the chain
                if res_name in amino_acids.keys():
                    residues.append(amino_acids[res_name])

        # If the chain doesn't exist we suppress the error and continue
        except KeyError:
            break

        # Put the found chain sequence into the list of found sequences
        chain_sequences[chain_name] = "".join(residues)

    # Return the chain sequences found
    return chain_sequences
```

7.3 Implementation of

```
# Loop through all antibody PDBs specified above
for antibody in antibodies:

    # Download PDB files using BioPDB
    get_pdb_file(antibody)

    # Construct file name variable of save location
    filename = "./structures/" + antibody + ".pdb"
    sequences = get_chain_from_pdb(filename, ["H", "L"])

    # If we were able to get a H and L sequences from PDB, the we accept it
    # Unfortunately, some of the PDBs in the dataset have named light and heavy chains
    # other names than H and L
    # I omit these from the exercise
    if "H" and "L" in sequences.keys():
        accepted_pdbs[antibody] = sequences

# Create an output directory, if it does not exist
output_location = "./predictions/"

if not os.path.isdir(output_location):
    os.mkdir(output_location)

# Predicted structures
predicted_structures = []

# Loop through all accept PDB files of antibodies
for antibody in accepted_pdbs.keys():

    # Set path of output file
    output_file = "./predictions/" + antibody + ".pdb"

    # If the structure have already been predicted, skip
    if os.path.isfile(output_file):
        predicted_structures.append(antibody)
        continue

    # Instanciate ABodyBuilder2
    antibody_builder = ABodyBuilder2()

    # Predict antibody and save it to the output file
    try:
        predict_antibody = antibody_builder.predict(accepted_pdbs[antibody])
        predict_antibody.save(output_file)
        predicted_structures.append(antibody)

    # In case we get an error from the prediction, we move on to the next antibody
    except AssertionError:
        continue
```

7.4 Implementation of PyMol to calculate RMSD

```
# Calculate RMSD of superpositioning predicted structure on crystal structure
# Loop through all predicted structures
for structure in predicted_structures:

    # Set object name of structures for use in PyMol
    crystal_name = "crystal_" + structure
    prediction_name = "prediction_" + structure

    # Set crystal and prediction file paths
    crystal_pdb = "./structures/" + structure + ".pdb"
    prediction_pdb = "./predictions/" + structure + ".pdb"

    # Load structures into PyMol
    cmd.load(filename=crystal_pdb, object=crystal_name)
    cmd.load(filename=prediction_pdb, object=prediction_name)

    # Select H and L chains
    select_crystal = crystal_name + " and chain H+L"
    select_crystal_name = crystal_name + "_chains"

    select_prediction = prediction_name + " and chain H+L"
    select_prediction_name = prediction_name + "_chains"

    cmd.select(select_crystal_name, select_crystal)
    cmd.select(select_prediction_name, select_prediction)

    # Superposition predicted structure onto crystal structure
    superposition = cmd.align(select_crystal_name, select_prediction_name,
                              object="Superposition")

    # Save RMSD value to dictionary
    rmsd = superposition[0]
    rmsd_values[structure] = rmsd

    # Delete everything in PyMol before we continue
    cmd.delete("*")

# Save RMSD values to file using Pandas DataFrame
rmsd_df = pd.DataFrame.from_dict(rmsd_values, orient="index", columns=["RMSD"])
rmsd_df.to_csv("./rmsd.csv", sep=";", index_label="PDB_ID")
```

7.5 PyMol script used to depict structures (filename: {PDB id}.pml)

```
# Load structures and hide everything
# Remember to change {PDB ID} to 4 letter id of PDB files
load .\structures\{PDB ID}.pdb, object=crystal
load .\predictions\{PDB ID}.pdb, object=prediction
hide

# Make selections of H and L chains in both the crystal structure and the predicted structure
select crystal_chains, crystal and chain H+L
select prediction_chains, prediction and chain H+L

# Show cartoon depiction of H and L chains in crystal structure and predicted structure
show cartoon, crystal_chains
show cartoon, prediction_chains

# Color H and L chains in crystal structure
color yellow, crystal and chain H
color red, crystal and chain L

# Color H and L chains in predicted structure
color green, prediction and chain H
color blue, prediction and chain L

# Superposition the predicted structure of H and L chain onto H and L chain of experimentally
# determined structure
align crystal_chains, prediction_chains, object=superposition
```

8 References

[1] R. Wu *et al.*, “Title: High-resolution de novo structure prediction from primary sequence,” 2022, doi: 10.1101/2022.07.21.500999.

[2] J. Jumper *et al.*, “Highly accurate protein structure prediction with AlphaFold,” *Nature*, vol. 596, no. 7873, pp. 583–589, Aug. 2021, doi: 10.1038/s41586-021-03819-2.

[3] A. W. Senior *et al.*, “Improved protein structure prediction using potentials from deep learning,” *Nature*, vol. 577, no. 7792, pp. 706–710, Jan. 2020, doi: 10.1038/s41586-019-1923-7.

[4] E. Callaway, “‘It will change everything’: DeepMind’s AI makes gigantic leap in solving protein structures,” *Nature*, vol. 588, no. 7837, pp. 203–204, Dec. 2020, doi: 10.1038/d41586-020-03348-4.

[5] R. Evans *et al.*, “Protein complex prediction with AlphaFold-Multimer,” 2022, doi: 10.1101/2021.10.04.463034.

[6] K. M. Yatim and F. G. Lakkis, “A Brief Journey through the Immune System,” *Clinical Journal of the American Society of Nephrology*, vol. 10, no. 7, pp. 1274–1281, Jul. 2015, doi: 10.2215/CJN.10031014.

[7] R. L. Stanfield, I. A. Wilson, and J. E. Crowe, “Antibody Structure,” 2014, doi: 10.1128/microbiolspec.AID.

[8] B. Abanades, W. K. Wong, F. Boyles, G. Georges, A. Bujotzek, and C. M. Deane, “ImmuneBuilder: Deep-Learning models for predicting the structures of immune proteins,” *Commun Biol*, vol. 6, no. 1, p. 575, May 2023, doi: 10.1038/s42003-023-04927-7.

[9] J. Dunbar *et al.*, “SAbDab: the structural antibody database,” *Nucleic Acids Res*, vol. 42, no. D1, pp. D1140–D1146, Jan. 2014, doi: 10.1093/nar/gkt1043.

[10] Y. Ma and J. Ma, “Immunotherapy against prion disease,” *Pathogens*, vol. 9, no. 3. MDPI AG, Mar. 01, 2020. doi: 10.3390/pathogens9030216.

[11] Schrödinger LLC, “The PyMOL Molecular Graphics System, Version 2.5.8.” Jan. 18, 2024.