LEVERAGING PROTEIN LANGUAGE MODELS FOR PREDICTION OF ANTIBODY PROPERTIES

A dissertation submitted in partial fulfilment
of the requirements for the MSc in Data Science

Matthew Balmforth

SEPTEMBER 17, 2023
BIRKBECK, UNIVERSITY OF LONDON
Department of Computer Science and Information Systems

Academic declaration

This report is substantially the result of my own work except where explicitly indicated in the text. I give my permission for it to be submitted to the TURNITIN Plagiarism Detection Service. I have read and understood the sections on plagiarism in the Programme Handbook and the College website.

The report may be freely copied and distributed provided the source is explicitly acknowledged.

Abstract

Antibodies demand intricate and costly engineering to obtain molecules with properties that enable them to enter the clinic and be delivered as treatments for disease. These properties are typically considered only after years of extensive discovery and can result in significant delays to patients. Earlier consideration of these properties is warranted, however computational costs of structural modelling for derivation of these properties is a limiting factor. Transformers, and in particular protein language models, offer a solution to this problem by encoding protein structure and biophysical properties in the model latent embedding space. To probe the extent to which protein language models can be used to derive antibody biophysical properties, two distinct language models were compared for their ability to generate antibody sequence representations for a transfer learning task. Computationally expensive structural modelling was used to derive aggregation propensity scores for a set of diverse antibodies, and a series of neural networks were trained for target value prediction. This research demonstrates that protein language models readily encode biophysical properties that can be used for training highly accurate supervised models. In addition, we show that these models can be deployed on large datasets for property prediction to flag problematic antibodies early in the discovery process.

Report contents

Table of contents

Table of figures

Table of tables

Acknowledgments

Sebastian Kelm and Mabel Wong provided access to code and resources to generate models and aggregation prediction scores without which I could not have developed the project to the extent that I did.

Thank you, Seb, for suggesting you could carry out structural modelling on so many antibody sequences. Thank you, Mabel, for the hard work you put in while Seb was away to provide me with the data I desperately needed to get this project off the ground. Without both of your help this project would have lacked a plot and would have been a struggle to write up.

Thank you to Phil for the encouragement and support at work.

Thanks to all friends and family and colleagues for listening to me complain about my masters. I promise not to complain any more.

1. Introduction

Antibodies are multifaceted proteins that have been engineered as therapeutic drugs for numerous diseases. Engineering these antibodies is complex and resource-intensive, requiring a detailed analysis of myriad sequences to discover sequences with optimal attributes, such as durability and specificity. Understanding these traits is pivotal for selecting antibodies with characteristics that will facilitate their development into real-world therapies. However, experimentally discerning these properties is demanding in terms of labour, time, and cost. Recent progress in protein language models (PLMs), underpinned by transformer architectures, stand poised to revolutionise drug research, facilitating computational predictions of antibody features including their hydrophobicity, shape, and function.

Enthusiasm for language models (LMs) has increased significantly, especially with the emergence of LM-driven tools, such as DALL-E and ChatGPT. Moreover, the democratisation of accessibility to LMs, thanks to open-source ventures like HuggingFace,[1] has resulted in greater uptake and significantly sped up their development. The realm of bioinformatics has harnessed these LMs, morphing them into PLMs, to execute various biologically useful tasks, ranging from deciphering structure,[2]–[4] to function projection,[5]–[7] and generative protein design.[8] Nevertheless, their use in forecasting antibody features is still in its infancy. The purpose of this project was to build on this body of research by creating a technique that couples PLMs to supervised learning for antibody property prediction.

Open-source antibody databases were queried to generate data for property prediction. These properties were used as target values to train machine learning algorithms. Feature extraction was carried out by sequence embedding of antibodies using two different language models. These models were compared for their ability to encode meaningful representations of antibodies for the purpose of property prediction. This study therefore represents a progression in the field of antibody development by improving the efficiency of antibody property prediction via cutting-edge computational techniques.

1.1. A brief background to antibodies

Proteins are intricate molecular structures that underpin all life. They perform a vast range of biological functions that are linked directly to their structure. The structure of a protein is governed by its amino acid building blocks that are assembled in a linear fashion and subsequently fold into a specific shape. There are 20 naturally occurring amino acids, each with their own biophysical properties, that make up the overwhelming majority of all proteins.

Antibodies are a class of protein that for the most part have a clearly defined structure that enables them to recognise and disable pathogenic targets (antigens) in the body. Antibody structure is composed of framework regions that act as scaffolding, onto which highly variable regions are mounted, dubbed complementarity determining regions (CDRs), which form the surface that recognises and engages an antigen (Figure 1). An antibody is actually composed of two proteins, a heavy chain and a light chain, across which the CDRs are spread and which assemble further into a complex of four proteins consisting of two heavy chains and two light chains (Figure 1). Since the ensemble of CDRs that bind to an antigen is highly variable between different antibodies, while the framework regions change minimally, modelling this interface is considered to be critical to understand antibody properties.

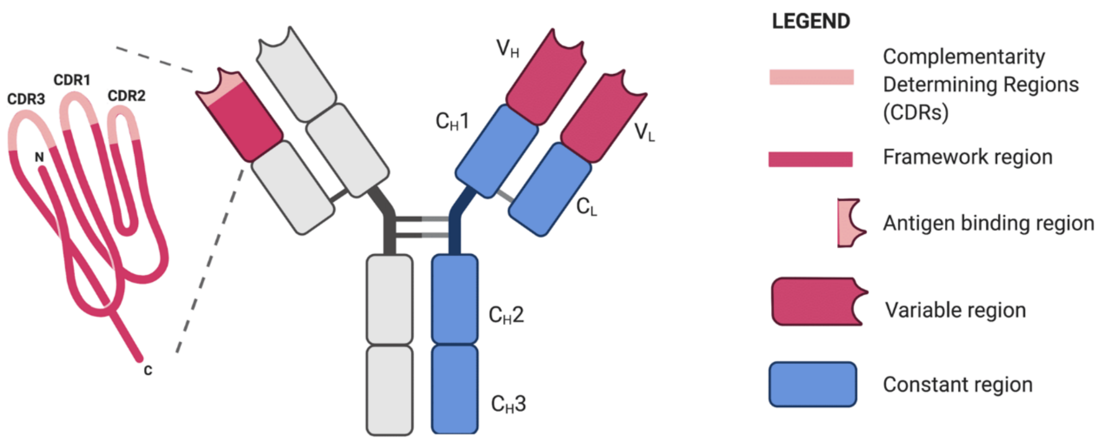

Figure 1. Representation of the structure of an antibody. The symmetric protein complex is composed of two heavy chains (VH, CH1, CH2, CH3) and two light chains (VL, CL). The antibody engages antigens via the complementarity determining regions (CDRs) that come together to form an antigen binding region, or paratope. Figure adapted from RapidNovor.[9]

Antibodies are often developed as drugs to treat disease, and are typically delivered to patients as an infusion. As such, the propensity for an antibody to self-associate and clump in a concentrated mixture, its aggregation propensity, is a property that must be fully understood in the drug development process. The state of the art for this prediction relies on first predicting the structure of the antibody, since its aggregation propensity is intrinsically linked to its shape.

Structural modelling of proteins has advanced significantly with the launch of DeepMind's AlphaFold.[10] AlphaFold has enabled highly accurate structure prediction to be carried out simply based on the amino acid sequence of a protein. Since the arrival of AlphaFold, other shape prediction frameworks, including Facebook's ESMFold,[11] have emerged. Central to all of these methods is the transformer.

1.2. Transformers and large language models

Over the last six years, transformers have emerged as the dominant neural network architecture in the field of natural language processing. Initially proposed by Vaswani et al. in 2017,[12] the transformer structure illustrated the potential of transformers for building large-scale language models that surpass traditional machine learning techniques across numerous benchmarks. By harnessing attention mechanisms, explained below, transformers enable the modelling of complex connections and dependencies within data. These advancements paved the way for the development of large-scale language models, such as GPT-3, equipped with billions of parameters and mimicking human-like text generation.[13]

Transformers introduced a paradigm shift by transitioning from sequential processing, where a sentence would be processed as a linear string of words, and moving to parallel processing, where a whole sentence would be evaluated in its entirety. This holistic approach to processing sentences offers a comprehensive context, enhancing the prediction accuracy for obscured words. As well as being more context-driven, this method also proved to be more resource-efficient than its predecessors, such as recurrent neural networks (RNNs) or long-short term memory neural networks (LSTMs), which previously set the benchmark for word prediction tasks.[14]

1.2.1. Transformer architecture

The Transformer architecture can be broken down into five key steps:

1. Tokenization and vectorisation

2. Input encoding
3. Target decoding
4. Linear and SoftMax transformations
5. Output of probabilities

Tokenization and vectorisation

The first step in the transformer pipeline is the process of text vectorisation, where words are encoded with vectors. This is a twostep process first involving tokenization, in which words are replaced with symbols, and subsequently vectorisation, where algorithms such as GloVe or word2vec encode the tokens with vectors that describe the semantic properties of the tokenized word. Position of a word in a sentence can also be encoded, and added to the word vector to denote positional index (Figure 2).

Training of the Transformer model usually occurs in a semi- or self-supervised fashion. When the Transformer was first introduced, its authors used an English-German sentence pair dataset for training. The model learns by adjusting weights to reduce the discrepancy between predicted and target values – in this context, either from English to German sentences or the other way around. This weight optimisation continues through backpropagation until the training concludes.[15] To streamline this, both the target and input sequences are processed concurrently, with both embedded using positional encoding, as highlighted in sections (1) and (3) of Figure 2.

Encoder stack

After input vectorisation, these vectors proceed to the encoder stack, illustrated in (Figure 2 (2)). This stack contains numerous layers of encoders, which consist in turn of multi-head self-attention mechanisms and a feed-forward neural network. This network refines the attention mechanism output through non-linear transformations before heading into another encoder or possibly a decoder layer.[16] Once processed through the encoder stack, it is then fed into the decoder stack.

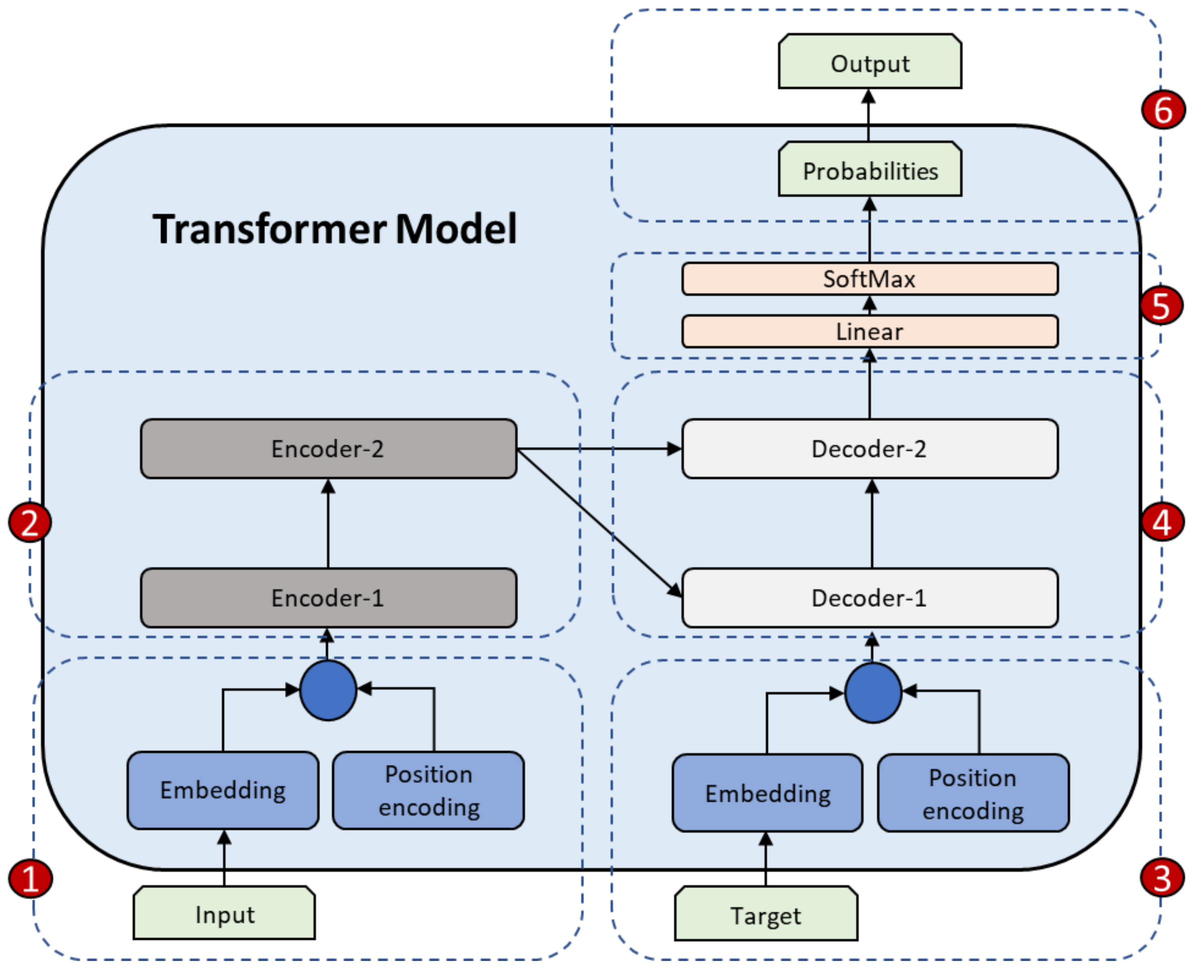

Figure 2. An overview of Transformer model architecture. The model has been divided into six logical sections of work.

Decoder stack and final transformations

The decoder stack is made up of multiple decoder layers. Each of these decoders is equipped with an attention layer, an encoder-decoder attention layer, and a feed-forward layer. This stack processes the target sequence embedding in conjunction with the encoded input to yield a concluding encoded depiction. This resultant depiction undergoes a series of transformations, notably a linear transformation and a SoftMax function, to produce a probability vector as its output.

Attention mechanisms

The encoder and decoder are underpinned by attention mechanisms, which are critical for learning the interplay between words, and therefore recognising their context.

Query, key and value vectors and attention scores

The concept of the attention mechanism revolves around the use of query, key, and value vectors. These vectors help derive attention scores, painting a coherent picture of relationships among words in a sentence. Throughout the training phase, weight matrices are honed to represent linear transformations of initial embeddings. Applying these transformations to fresh sequence encodings results in derivation of these three vectors. The correlation between word pairs emerges by taking the dot product of a specific query and the key vector of another word. This procedure repeats for all word pairings, resulting in attention scores. These scores are then scaled down by the square root of the vector dimension and standardized using the SoftMax function, resulting in attention weights.

The culmination of this process involves computing the weighted sum of the value vectors with the aforementioned attention weights. This produces a solitary vector that embodies a token within the full input sequence context.

Multi-head attention

Here, the three key vectors (query, key, and value) are divided into N separate vectors. Each of these vectors undergoes self-attention, passing through what is termed an 'attention head'. The framework is made up of numerous such heads, and every attention head produces its own output vector. Once all heads have processed, their outputs are fused into a single vector and undergo a linear transformation. This new structure aims to encapsulate the semantics and syntax of the processed word. Interestingly, the different head outputs can be harnessed to unravel varied semantic and syntactic nuances.[17], [18]

1.2.2. Large language models

BERT

Unveiled by Google in 2018, BERT (Bidirectional Encoder Representations from Transformers) represented a pioneering application of the Transformer model to large-scale language modelling.[18] Its development saw the emergence of a novel strategy called masked language modelling (MLM). Here, tokens are obscured at random and then predicted. The objective is to curtail the cross-entropy loss between the model prediction of this concealed token and the ground truth value. Given the implementation of MLM is inherently self-supervised, the extent of the training data is solely dictated by the length of the corpus.

The creators of BERT also implemented a bidirectional flow, such that BERT considers context both from the left and right of the sentence. This bidirectionality improves the model understanding of context as well as improving its ability to represent the relationships between words in a sentence, making it powerful for natural language processing (NLP).

T5

T5 (Text-to-Text Transfer Transformer) emerged in 2019 as another language model developed by Google.[19] Rather than following traditional language models, T5 zeroes in on converting one textual sequence into another, hence "text-to-text transfer". Post fine-tuning, this framework lends itself to an array of NLP applications, including language conversion, content abstraction, automated query response, or text classification.

Architecturally, T5 is underpinned by a sequence-to-sequence (Seq2Seq) model variant of the Transformer. This equips the model with the ability to capture dependencies within input-output sequences. Training of T5 operates on supervised learning, where the system is conditioned to correlate input sequences with corresponding outputs via an extensive paired data set. Another feature of T5 is task-centric pretraining, a pretraining step that involves honing the model for specific NLP challenges prior to its deployment. This type of specialisation positions it to deliver top-tier outcomes across diverse NLP challenges.

1.3. Protein Language Models

The power of NLP, accelerated by transformers, has now been harnessed to decipher the complex language of protein sequences through unsupervised learning. ProtT5 is a protein language model (PLM) derived from the T5 framework and enriched by extensive protein databases. Its prowess lies in its ability to assimilate protein-specific features, which it then applies to protein sequence annotations, matching the precision of cutting-edge methodologies.[20] Specifically, ProtT5 demonstrates a nuanced understanding of amino acids, predicting their secondary structures and pinpointing the animal kingdom from where the protein is expressed.

The methodology for sequence embedding follows a similar workflow to conventional LMs. Amino acids are categorised as words within the greater context of a protein sentence. Amino acids are tokenized and vectorised like words and given positional encoding. The T5 transformer then processes these vectors. The final sequence embeddings are gleaned from the last hidden state. Capitalising on this high-dimensional output, the authors of ProtT5 employed it for transfer learning within a convolutional neural network (CNN). This CNN, tailored for structure prediction, executed global average pooling, and churning out a fixed-size embedding independent of sequence length. The art of inductive transfer further improved their approach, driving the training of a neural network geared towards understanding overarching protein features, such as subcellular localisation.

Building on the successes of AlphaFold and ProtT5, Facebook released ESM-2,[21] the largest protein language model (as of September 2023) trained on 65 million proteins and implemented with 15 billion tunable parameters. ESM-2 outperforms ProtT5 in protein folding, with structural prediction comparable to AlphaFold in RMSD yet performing the computations 60-fold faster. However, exploitation of colossal datasets and parameters has been fairly criticised. These LLMs have sparked debate over sustainability given their demand for computational power. In an effort to democratise and streamline, the research community has pivoted towards building smaller models trained on quality over quantity. As an example of this, Google developed Ankh, a PLM that outperforms ESM-2 in seven pivotal protein function benchmarks despite requiring 90% fewer parameters.[22]

Table 1. Summary of the features of the different protein language models. Architecture of either encoder or encoder-decoder models, the dataset the models were trained on, the number of parameters i.e. the number of learned values for neural network weights, scaling factors and biases, number of attention heads, number of encoder layers in the encoder stack, embedding dimension, and top model are presented.

	ProtT5	ESM-2	Ankh large	AbLang
Architecture	Encoder/decoder	Encoder only	Encoder/decoder	Encoder only
Dataset	UniRef50	UniRef50	UniRef50	OAS
Parameters	3B	15B	1.15B	1.5B
Attention heads	32	40	16	12
Encoder layers	24	48	48	12
Embedding dimension	1024	5120	1536	768
Top model	CNN	MLP	ConvBERT	AbHead

1.3.1 AbLang

The PLMs described above have demonstrated use in describing the general language of proteins, and use large datasets such as UniProt and BFD for training.[20] Because of this broad training, they may be less applicable to highly specialised tasks such as describing antibodies. To resolve this, the Deane group developed the observable antibody space (OAS), a database reflecting all publicly available antibody sequences[23], [24] and used this large dataset to develop AbLang, a collection of two PLMs trained on either heavy chain (AbLang_heavy) or light chain (AbLang_light) sequences in the OAS.[25] AbLang, heavy or light, thus represents the first antibody-specific PLM. AbLang is available via the HuggingFace transformers library, enabling easy access to the tokenizer and pretrained model.

AbLang is based on the RoBERTa transformer encoder-decoder architecture,[26] and is trained solely on unpaired antibody sequences in a self-supervised manner to learn sequence representations. The model consists of two components – AbRep and AbHead. AbRep is the encoder module that creates representations of the input antibody sequences, while AbHead takes those representations and uses them to predict missing amino acids in the sequence. The AbRep architecture is identical to RoBERTa, a BERT-related LM, except for a learned, rather than a fixed, positional embedding layer with a maximum sequence length of 160.

The choice to opt for a learned positional embedding layer likely stems from the application of the model to backfill missing residues in antibody sequences. During learned positional encoding the position vectors are randomly initialised and learned during training, and thus the model learns the optimal position vector representations. Positional encoding requires more computation but allows the model to learn positional information and therefore predict where gaps may be and what should fill those gaps. Learned positional encoding comes at the cost of being constrained to sequences no longer than the maximum length they were trained on, hence the maximum sequence length stipulation of 160.

Sequence representations, or embeddings, are generated by passing antibody sequences into the tokenizer that replaces the amino acids with tokens and pads out all sequences so that the embeddings are padded out to the same length for each antibody sequence to a maximum of 160 amino acids. The padding is represented by special token IDs that are interpreted by the model and factored into the embedding.

The group compared AbLang to ESM-1b for prediction of a number of antibody features for which AbLang generally outperformed ESM-1b. While the authors recognise the potential for using AbLang to predict antibody features more broadly, the core focus of the paper exploited MLM to backfill antibody sequences in the OAS containing missing or ambiguous amino acids.

1.3.2. Ankh

Ankh is a protein language model developed by Elnaggar et al. that was designed for computational efficiency and performance on downstream prediction tasks. The model architecture follows an encoder-decoder design, inspired by ProtT5. Ankh was pretrained on the UniRef50 database of protein sequences in a self-supervised manner using a masking strategy that employs ConvBERT’s span-based dynamic convolution (see below for details). Two versions of Ankh were released – Ankh base, with 450 million parameters, and Ankh large, with 1.15 billion parameters.

Several optimisations were made in Ankh compared to previous protein language models that focussed on volume of training data and parameters to obtain state-of-the-art performance. Critically, the authors evaluated different masking strategies, architectures, and training datasets empirically in a series of experiments, leading to choices like span-based masking and training on the UniRef50 protein dataset.

As with AbLang, fixed positional encodings were ditched, although the authors selected relative, rather than learned, positional embeddings, enabling the model to handle longer sequences.

Gated-GELU activations were introduced in the encoder-decoder stacks as well as the top model to improve modelling capability. The GELU function provides a balance between the non-linearity of the ReLU (Rectified Linear Unit) and smoothness of the Sigmoid functions. The gated version of GELU further incorporates gating mechanisms, which allow the network to learn and control the flow of information.

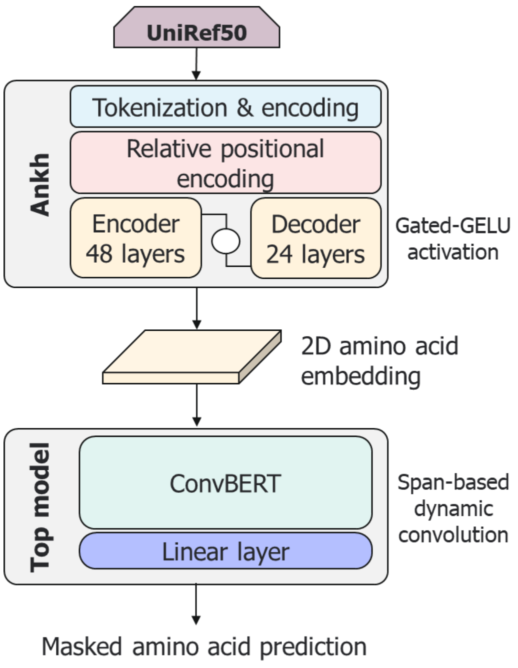

Figure 3. Overview of Ankh architecture for training. Ankh was trained on the UniRef50 dataset, which was tokenized and encoded with relative positional encoding. Encodings were passed into an encoder-decoder stack composed of 48 encoder layers and 24 decoder layers and transformed between layers with gated-GELU activation. Amino acid embeddings were transferred to the ConvBERT top model for supervised learning by span-based dynamic convolution followed by a final transformation with a linear layer to output probabilities for masked amino acids.

On a suite of protein prediction tasks, Ankh showed state-of-the-art performance on seven benchmarking tests while using ten times fewer parameters than previous top models like ESM-2 (Table 1). Ankh also showed greater computational efficiency, extracting features over seven times faster on long sequences. Importantly, two of the aforementioned benchmarking tests were for protein solubility and GB1 fitness, a test for predicting impact of amino acid substitutions on the folding and stability of a protein.[27]

ConvBERT

ConvBERT is a sophisticated language model developed by Jiang et al.[28] that integrates convolution operations into the transformer architecture. The goal was to improve the efficiency and performance compared to the language model BERT. Importantly, ConvBERT departed from the traditional self-attention mechanisms implemented in BERT by introducing mixed attention, whereby both convolution-based attention and self-attention are integrated into each attention head.

The convolution-based attention mechanism incorporates novel span-based dynamic convolution layers. These layers are specifically designed to capture and model the amino acid context within the local environment to provide improved precision (Figure 4). Rather than using static convolution filters, these layers operate by taking spans of tokens as input and dynamically generating dynamic kernels based on the context they present. This dynamic approach ensures that the model remains adaptive and responsive, within the input sequence, to the relevant biophysical properties an amino acid projects into its local environment, thus better capturing interactions between amino acids.

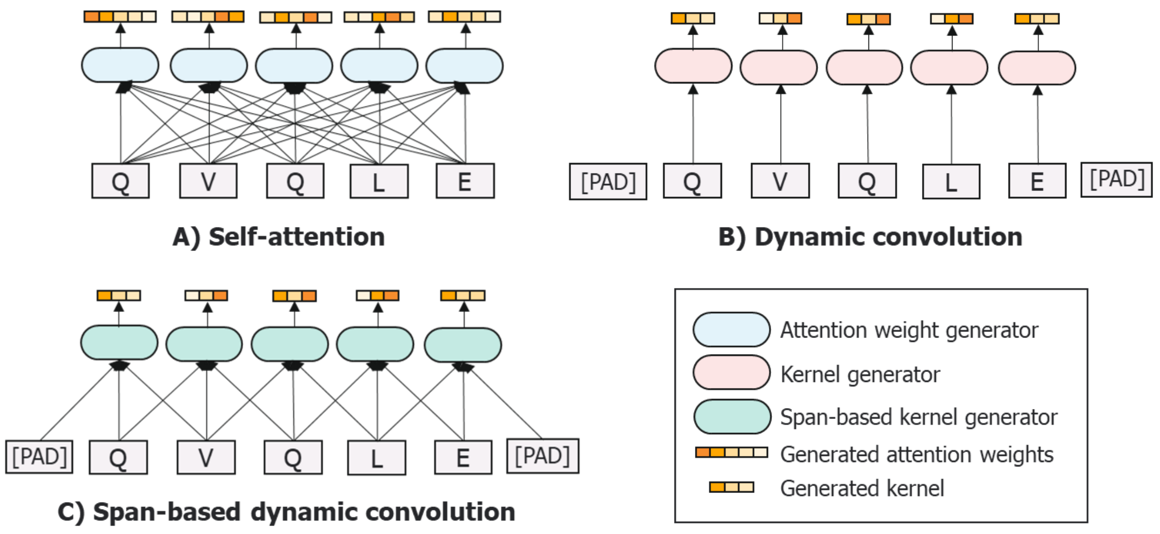
A) Self-attention
B) Dynamic convolution
C) Span-based dynamic convolution
Q
V
Q
L
E
[PAD]
Attention weight generator
Kernel generator
Span-based kernel generator
Generated attention weights
Generated kernel

Figure 4. Different methods for convolution kernel or attention weight generation. A) In self-attention every input token is used for attention weight generation, necessitating O(n^2) computational complexity. B) In dynamic convolution a single token gives rise to a single dynamic kernel, producing a vector representation that encodes all possible properties of that specific token. C) In span-based dynamic convolution the local environment of the token is sampled to generate dynamic kernels that encode the properties of the token given its local dependencies. Figure adapted from Jiang et al.[28]

To enhance computational efficiency, the creators of ConvBERT employed a so-called bottleneck strategy whereby token embeddings were projected into a lower-dimensional space before undergoing mixed-attention. This process served to reduce redundancy across multiple attention heads as well as streamlining model computations.

Another innovation resulting in efficiency gains was the introduction of grouped linear layers in the transformer feedforward network. This concept is borrowed from convolutional neural networks, where the neurons in between layers are typically divided into groups and process a subset of the features. This is in contrast to a standard feedforward layer, where every neuron processes every feature. By dividing the feedforward network into groups, the number of necessary parameters was severely reduced. Surprisingly, this reduction did not compromise the model's representational power, as evidenced by performance in the GLUE natural language understanding benchmark, where ConvBERT matched or exceeded state-of-the-art models like ELECTRA while incurring a significantly smaller training cost.

1.4. Predicting antibody solubility

1.4.1. ABodyBuilder2

Transformers in structural prediction have recently been applied to the realm of antibodies. Most recently, a deep learning framework developed by the Deane lab for predicting antibody 3D structures from sequence has been developed, called ABodyBuilder2, which uses a transformer architecture adapted from AlphaFold2.[29]

ABodyBuilder2 first represents each amino acid residue as a distinct node, characterised by its specific properties and a positional encoding reflecting its sequence location. Edges drawn between these nodes signify the relative distances between the residues. As the model processes the sequence, encoder blocks equipped with attention layers continually refine these node representations by gleaning contextual information from the entire sequence. A final layer of the model predicts the backbone's torsion angles, which are subsequently translated into all-atom coordinates.

AbodyBuilder2 was trained on ~7,000 available antibody structures sourced from the SAbDab database, yet another database created and maintained by the Deane lab.[30] The loss function underpinning its training was composed of a *Frame Aligned Point Error* component, which ensures that the predicted structures align closely with the known structures. Additional elements of the loss function penalised structural anomalies such as steric clashes, while also promoting improved sidechain geometry.

To bolster the accuracy and robustness of its predictions, ABodyBuilder2 employs an ensemble approach of four distinct models, with each making independent structural predictions. The outputs of each model are then averaged together to generate the final model. Final models are subjected to a final refinement step, whereby physics-based energy minimisation is used to adjust and remedy any steric clashes and further enhance the structural geometry.

On test evaluations, ABodyBuilder2 rivalled the performance of AlphaFold-Multimer, the state-of-the-art for structure prediction, whilst predicting structures up to 100 times faster. These high-fidelity structures predicted hold practical significance, enabling the computation of antibody biophysical properties such as aggregation propensity.

1.4.2. Using MAPT to predict aggregation

The monoclonal antibody aggregation prediction tool is an algorithm that takes an antibody structure as input, for instance from ABodyBuilder2, and derives a measure of hydrophobicity.[31] This measure is based on the solvent-accessible surface of amino acids with hydrophobic properties. Combined with the charge of the molecule, a score for aggregation propensity can be derived. This aggregation score is referred to as the MAPT score, and correlates extremely well with therapeutic antibodies for which experimental data is available.

MAPT functions by using a suite of packages, in particular the PSA package, which functions by reading the atom coordinates stored within the PDB files, and subsequently determining which amino acids have their sidechains protruding from the surface of the protein. Since a subset of the 20 possible amino acids are considered hydrophobic (phenylalanine, isoleucine, tryptophan, leucine, valine, methionine, alanine and glycine) a hydrophobicity score can be determined by counting the number of hydrophobic amino acids present at the solvent-accessible surface of the protein. This process is carried out by MAPT for both the heavy and the light chain, since they are considered to be separate molecules within each PDB file despite making significant contacts with each other, and thus carry separate hydrophobicity scores. The scores were then summed to obtain a hydrophobicity score across the whole antibody.

Net charge of the antibody is subsequently determined simply by calculating the charge of each amino acid at a given pH, summing the positive charges, and then subtracting the negative charges from the respective acidic amino acids.

MAPT score can then be determined using the derived information in conjunction with the equations outlined below.

$$H = (Hl + Hh - 300)/500$$

Hl and Hh are the light and heavy chain hydrophobicity scores, respectively.

$$N = 6 + (0.5 * Q7)$$

Q7 is the unweighted net charge at pH 7.4.

$$\mathrm{C} = 1 + (\mathrm{C}_{cr} - 17.5) \div 20$$

C_{cr} is the number of acidic amino acids in antibody.

$$MAPT\ score = H * N * C$$

MAPT is considered to be state-of-the-art for aggregation propensity prediction,[31] deriving nearly all of its information from sophisticated deep-learning and physics-based structure prediction tools such as ABodyBuilder2. However, despite being significantly faster than AlphaFold2, structure prediction is still relatively slow, approaching roughly one structure every 12 seconds on a graphical processing unit (GPU). In the era of big data, where deep sequencing is enabling the elucidation of millions of antibodies, tools that predict properties within those timeframes are unable to be deployed at scale. As such, algorithms capable of approaching this predictive capability, but on a significantly faster timescale are highly desirable.

1.5. Project aims

The aims of this project as stated in the initial proposal were to leverage large language models to embed antibody features, and to transfer those features to a machine learning model for a supervised learning task, with values for model training and evaluation to be derived using MAPT. Two language models, Ankh and AbLang, were to be compared to understand the impact of transferring embeddings from an antibody-specific model versus a generalised protein language model for the task of predicting antibody aggregation propensity.

2. Methodology

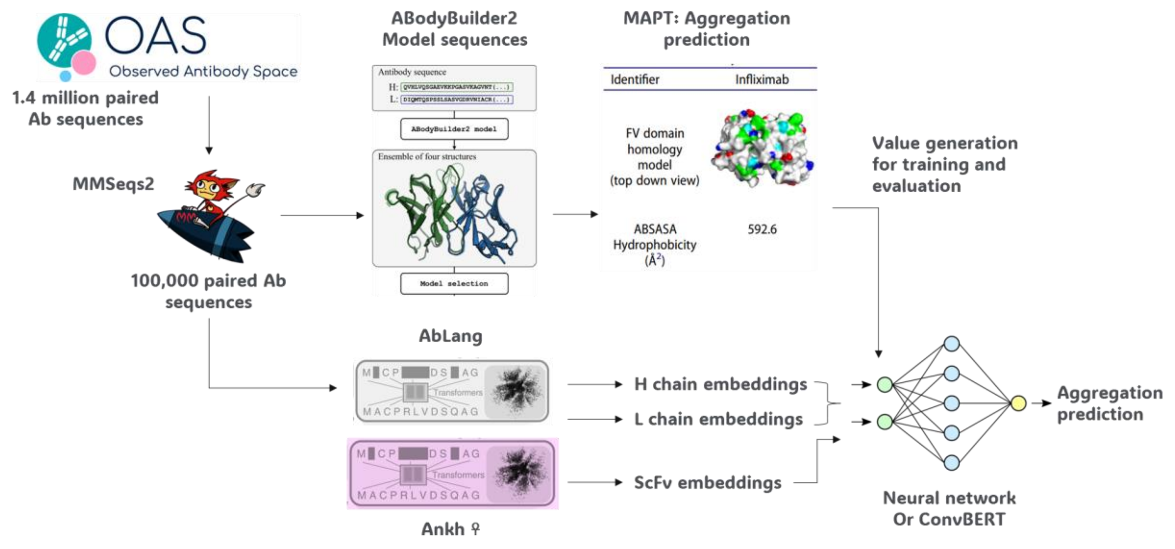
OAS
Observed Antibody Space
1.4 million paired Ab sequences
MMSeqs2
100,000 paired Ab sequences
ABodyBuilder2 Model sequences
Antibody sequence
ABodyBuilder2 model
Ensemble of four structures
Model selection
MAPT: Aggregation prediction
Identifier
Infliximab
FV domain homology model (top down view)
ABSASA Hydrophobicity (Å²)
592.6
Value generation for training and evaluation
AbLang
Transformers
MACPRLVDSQAG
H chain embeddings
L chain embeddings
ScFv embeddings
Ankh ♀
Neural network Or ConvBERT
Aggregation prediction

Figure 5. Project pipeline. Data pulled from OAS is fed into MMSeqs2 for clustering and sequence selection for 100,000 paired antibodies. Paired antibodies are structurally modelled using ABodyBuilder2, then assessed for aggregation propensity using MAPT. In parallel, paired sequences are embedded either into AbLang, or Ankh. Resulting representations are transferred either to an MLP framework for model training with aggregation propensity score, or coupled to ConvBERT for training and model prediction.

Computation was carried out on an Intel Xeon CPU @2.30 GHz with 2 vCPUs and 13GB RAM for all data extraction and preparation steps (sections 2.1 to 2.4).

2.1. Data extraction from OAS

Code available in extract_cluster.ipynb, data_preparation.py

Paired antibody sequences were pulled from the database as zipped csv files using a shell-script containing a set of *wget* linux commands made available by the database maintainers. Each file is composed of a line of metadata containing information such as the antibody species, followed by a dataframe of antibody sequences and their respective annotations, including many features that were not important for this study, including CDR and framework extractions. Redundant features were dropped. Files were parsed and filtered based on 'Species' to obtain human only antibody sequences. Additional parsed metadata included 'Run', which was added as a new column to each dataframe. All dataframes were merged, and a unique sequence identifier was generated for each antibody sequence pair by concatenating heavy chain ID, light chain ID and Run identifier. Unique IDs were forced by adding a suffix to any duplicated sequence ID.

ScFv sequences were generated by concatenating heavy and light chain amino acid sequences separated by a biologically relevant linker (SGGSTITSYNVYYTKLSSSGT).

2.2. Data cleaning

Code available in extract_cluster.ipynb, data_preparation.py.

Sequences were filtered based on ANARCI status containing the key phrase "Shorter than IMGT defined". Dataframes from human antibody files were concatenated into a single file with 1,410,689 paired antibody sequences.

2.3. Data processing using MMSeqs2

Code available in extract_cluster.ipynb, data_preparation.py.

MMSeqs2 was installed using conda and conda-forge and run on all 1,410,689 ScFv antibodies using the easy-linclust workflow for clustering large datasets with a minimum sequence identity of 66.4% and a minimum coverage for clustering of 80% based on the length of both the query and target.

2.4. Modelling and MAPT score generation

Code not available.

102,504 paired antibody sequences were modelled using ABodyBuilder2[29] on a single CPU (Intel(R) Xeon(R) Gold 6242 CPU @2.80GHz) and a single GPU (NVIDIA GeForce RTX 2080 Ti) with a walltime of 11.5 days, extrapolated from 30,000 sequences (4866 mins).

The resulting PDBs adopted IMGT numbering and were renumbered to Chothia convention using ANARCI[32] for MAPT score prediction. MAPT scores for each modelled antibody were determined from the renumbered PDB files using PSA (protein surface accessibility) from the JOY package[33] to calculate the surface hydrophobicity of the antibody model. This was parallelised on 16 CPUs (Intel(R) Xeon(R) Gold 6242 CPU @2.80GHz) with a walltime of 6 hours to renumber all 102,496 PDBs and 8.5 hours to derive MAPT scores.

2.5. AbLang

Code available in ablang_seq_embedding.ipynb, plm_manipulation.py.

AbLang Heavy and AbLang Light models and tokenizers were imported from the HuggingFace transformers library. Heavy and light chain amino acid sequences were tokenized and padded to max length. Encoded sequences were embedded using AbLang without gradient calculation, and the last hidden state was extracted. The last hidden state was averaged along the sequence length dimension to generate 1D sequence embeddings. To obtain sequence representations of the entire antibody, heavy chain and light sequences were embedded separately and concatenated to produce one sequence embedding with dimension 1536.

AbLang embeddings were computed either using an Nvidia V100 with 52GB RAM to generate tensors, or an Intel Xeon CPU @2.30 GHz with 2 vCPUs and 13GB RAM.

2.6. Ankh

Code available in ankh_seq_embedding.ipynb, plm_manipulation.py.

Ankh was not available through the HuggingFace transformers library and was installed as a separate package. ScFv or heavy chain sequences were encoded using the tokenizer and subsequently embedded using the Ankh large model. Ankh large embeddings were carried out with GPU acceleration on an Nvidia A100 GPU. Batch embedding for tensor generation was employed to overcome memory limitations, with batches of 100 sequences embedded, then saved to disk followed by GPU cache clearing in between batches.

2.7. MLP architecture

Code available in ablang_embedding_to_mapt_score.ipynb, ankh_embedding_to_mapt_score.ipynb , ML_manipulation.py.

The continuous values for aggregation propensity were normalised and standardised to have a mean of 0 and a standard deviation of -1. The dataset was then broken up into training and testing sets. The training set was composed of 45,000 sequences and their respective aggregation propensities. The testing set comprised 15,000 sequences held out from the training set for MLP model performance evaluation. All models were trained for 20 epochs and with a batch size of 64 unless otherwise stated. Models were trained on 80% of the training set, with 20% reserved for validation. The model was evaluated by predicting the testing set and comparing predictions to test values by extracting performance metrics including R^2 score, mean absolute error, root of the mean squared error, and Pearson's correlation coefficient. All MLP training was carried out on an Intel Xeon CPU @2.30 GHz with 2 vCPUs and 53GB RAM.

A simple feedforward regression neural network(MLP-simple) was assembled using the *Keras* library. The model consisted of a fully connected input layer of 512 neurons, activated with rectified linear unit (ReLu); a dropout layer for overfitting prevention ignoring 50% of the neurons in the network; and an output layer, activated with a linear function for continuous value prediction. The model was compiled using a *mean squared error* loss function and *Adam* was used for adaptive learning rate optimisation.

A more complex MLP (MLP-complex) was implemented based on the MLP-simple architecture. MLP-complex consisted of three hidden layers with 256, 128, and 64 neurons respectively and each activated with ReLu. The first two hidden layers additionally had dropout layers with 20% dropout. The final linear layer and model compilation followed MLP-simple.

A KFold consensus was built by splitting the training data into five folds and training MLP-complex on each fold. Model predictions were averaged between the models by taking the mean to obtain a consensus.

Batch size and epochs were optimised using MLP-complex, cycling through training and model prediction with seven batch sizes ranging from 16 to 1024 and 14 epochs training from 10 to 45,000.

Hyperparameter optimisation was carried out using Optuna. For AbLang embedding-trained MLPs (AbLang-MLPs) a variant of MLP-complex was used for optimisation experiments. The variant (MLP-to-opt) was composed of three hidden layers, each activated with ReLu, and three dropout layers. The hidden layers and dropout layers were separated by a batch normalisation function. The number of neurons and dropout rate were subjected to optimisation for 100 trials, with each model trained over 20 epochs and with an optimised batch size of 512. Neurons were optimised with a step of 128, 64 and 32 within ranges of 128-1024, 64-512, and 32-256 for hidden layers 1, 2, and 3, respectively. Dropout rates were optimised with a step of 0.1 and within a range of 10-70% dropout.

Ankh embedding-trained MLPs (Ankh-MLPs) were optimised using a complex architecture due to lack of improvement with simpler model frameworks. The architecture followed an initial hidden layer with number of neurons in the range 256-2048 (to be suggested by Optuna), with an activation choice between ReLU and Leaky ReLU. If Leaky ReLU was selected an value for the hyperparameter leaky alpha was suggested between 0.01 and 0.3. A dropout layer was also introduced within a range of 20-80% neuron dropout. Further hidden layers could then be added as needed with a maximum of five hidden layers permissible, with each subsequent layer from the first having a fixed ReLU activation. Finally the learning rate was also inputted as a tuneable parameter in the range of 1e-5 and 1e-2 with log steps.

2.8. ConvBERT for MAPT score prediction

Code available in convbert_to_mapt_score.ipynb.

ConvBERT model training and evaluation was carried out on an NVIDIA A100 Tensor Core GPU.

Ankh-base model and tokenizer were employed together with the Ankh ConvBERT for regression class (ConvBertForRegression) for model training. This class uses a modified version of the ConvBERT model[28] together with specific pooling mechanisms and a linear regression head to produce the final regression output. 10,000 paired antibody sequences with their normalised and standardised MAPT scores were used for training, with 25% of this data held out for validation. Further evaluation was carried out on an additional 10,000 ScFv sequences not present in the training or validation set.

The AnkhWithConvBertForRegression class was built to hold the base model (Ankh-base) and the regression model (ConvBertForRegression) and inherits nn.Module, the base class for all PyTorch models. The class method *forward* feeds the input IDs and attention mask from the tokenized sequences into the base model to obtain the model embedding. The last hidden state of this sequence representation is then extracted and added to the regression model.

A custom dataset class (CustomDataset) was built to hold the tokenized sequences and MAPT scores and adheres to the PyTorch Dataset interface, implementing __*len*__ and __*getitem*__ methods, a prerequisite for PyTorch datasets.

ConvBertForRegression was instantiated using default parameters suggested by the creators of Ankh. Ankh-base was used, whose last hidden state has an embedding dimension of 768. The input dimension of the regressor function was thus kept at 768. The number of attention heads was kept at the default of 4, a single hidden layer was implemented with a dimension of 384, a kernel size of 7, no dropout, and a maximum pooling step. The mean of the training values (0) was also included for computational speed.

ConvBERT training was performed using the TrainingArguments configuration in the HuggingFace Transformers library. A learning rate of 5e-5 and weight decay of 0.01 were used with the AdamW optimizer (an Adam variant with weight decay) that is automatically used by the TrainingArguments framework to tune learning rate for optimal fit. Warmup for stability of the initial training was implemented for the first 500 steps, whereby the learning rate was progressively increased from a low value to its maximum value (5e-5). Evaluation was carried out every 500 steps, with model checkpoints saved at the same interval, recording training and validation loss.

Following model training, models were saved and evaluated on an unseen test set. Evaluation was carried out by loading the trained model, and passing Ankh-base tokenized sequences into the model. The first dimension of the embedded output was saved as the MAPT score prediction. Predictions were compared to ground truth values and regression metrics such as R^2 score, mean absolute error, root of the mean squared error, and Pearson's correlation coefficient were extracted.

3. Results

3.1. Data extraction and preparation

The Deane lab in Oxford maintain and update a series of databases and webservers from which antibody data can be pulled. Of particular interest to the antibody research community is the Observable Antibody Space (OAS), a database of every antibody for which the sequence has been described.[34] The OAS has both unpaired and paired antibody data. The unpaired dataset is much larger, since it is comparatively trivial to bulk sequence heavy or light chains from immune samples (e.g. blood), while it is significantly more complicated to obtain single cell data from which paired heavy and light chains can be obtained. However, since both the heavy and light chain contribute to the binding of an antibody to an antigen, paired data should be studied to generate the most biologically meaningful models.

The OAS contains 1,777,462 paired antibodies (as of September 2023), which were pulled and extracted from the database using a script made available by the database maintainers. Importantly, each file contained metadata, including antibody species, which enabled filtering out of non-human antibodies. In total 1,487,957 human antibodies were extracted from OPIG. These were filtered further down to 1,410,689 antibody sequences by removing any sequences that were considered shorter than a standard antibody, i.e. the sequencing was incomplete and thus the ends of the antibody were missing. The data pulled from OPIG conveniently comes pre-annotated with such warnings.

One of the key steps in this project was to model a selection of antibodies using tools to predict the structure of the antibody, and subsequently use the structural data to infer properties that could be used as labels for transfer learning. However, structural modelling is a computationally intensive task, and would be impractical for 1.4 million sequences. A number of 100,000 was deemed more practical, given the computational resources available: a single Nvidia GeForce GPU for two weeks. As a result, it was desirable to reduce the dataset from 1.4 million to 100,000 paired antibodies. In order to obtain a generalisable model, it is often advisable to use a dataset that captures the true diversity of the data. Randomly picking 100,000 antibodies would not necessarily achieve that, and instead, MMSeqs2 was employed to group antibodies into 100,000 clusters based on sequence similarity, with each cluster containing its most representative sequence and thus capturing the diversity in the sequence data.

MMSeqs2 was run at a range of different cluster thresholds in order to reduce the dataset to the required number of paired antibodies for subsequent structural modelling. Eventually, a threshold of 66.4% similarity was reached, which gave 103,152 paired antibody sequences.

3.4. Implementation of Ablang

3.4.1. Clustering embeddings with dimensionality reduction and KMeans

Heavy and light chain sequence embeddings were generated for the set of antibodies obtained following clustering with MMSeqs2. In the Ablang paper,[25] the authors initially use the dimension reduction technique t-SNE to demonstrate that the language model has learnt useful sequence representations. They use antibody germline V-gene to colour classify their clusters and demonstrate that the clusters separate based on this germline identifier. These experiments were repeated here, reducing the 768 dimensions that AbLang returns from the sequence embedding to the number of components specified for the dimension reduction technique. For visualisation purposes the number of components is set to two. To better understand the information that the language model was capturing in the V-gene segmentation, three different dimension reduction techniques were applied.

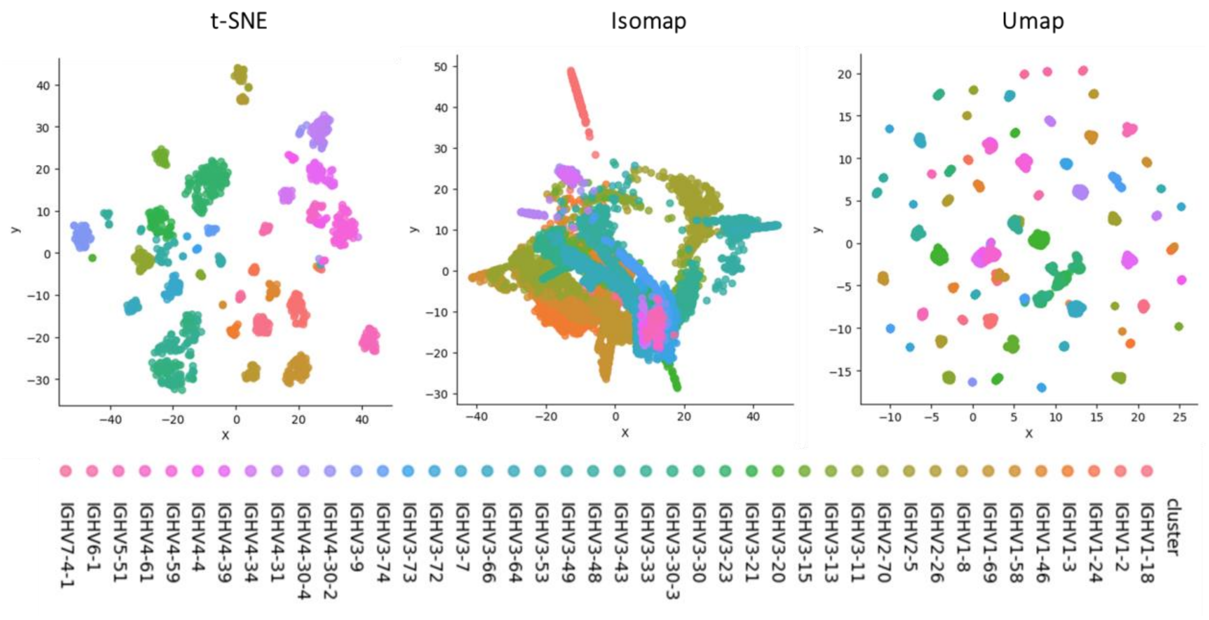

Figure 6. Dimension reduction of AbLang heavy chain sequence embeddings using three techniques: t-SNE, Isomap, and UMAP. Dimension reduced datasets visualised and coloured by germline V-gene

The first technique was t-SNE, as per the publication. However, while t-SNE is useful for visualisation of data, it is known to capture global data structures quite poorly,[35] and thus distance between well separated clusters may not be meaningful. To address this limitation, UMAP and Isomap were also implemented. UMAP aims to capture local and global data structures,[36] while Isomap seeks to preserve distances between clusters, and thus better reflects the underlying geometry of the data.[37]

Looking at the data by eye (Figure 6), the cluster segmentation is more pronounced in UMAP, whilst visual interpretation of the Isomap plot is challenging. To try to quantify the degree of separation using these dimension reduction techniques, a KMeans clustering algorithm was applied. A function for assessing data separation by KMeans was implemented and used to return cluster purity given an array of *K* clusters.

Table 2. Highest cluster purity for each dimension reduction model as calculated by KMeans. Dimensions reduced to two components with t-SNE and UMAP while Isomap tested at two and four components.

K clusters	Cluster purity	Dimension reduction model	Number of components
38	0.7619	t-SNE	2
44	0.7609	UMAP	2
36	0.626	Isomap	4
32	0.4459	Isomap	2

Table 3. Top five highest cluster purities as calculated by KMeans following dimension reduction with t-SNE, UMAP and Isomap.

K clusters	Cluster purity	Dimension reduction model	Number of components
38	0.7619	t-SNE	2
44	0.7609	UMAP	2
42	0.7527	UMAP	2
40	0.7521	UMAP	2
34	0.7236	UMAP	2

Table 2 shows the highest cluster purity obtained for each dimensionality reduction technique. While KMeans is able to obtain the highest cluster purity with t-SNE, the optimal number of clusters for t-SNE is 38, whilst the real number of clusters is 44. UMAP, meanwhile, has an optimal cluster number equal to the real cluster number. Additionally, UMAP outperforms t-SNE in terms of the cluster purity across a range of *K* clusters (Table 3). Isomap performs worse across the board, with a significant uptick in cluster purity (0.45 to 0.63) when the number of components is increased to 4. As mentioned above, Isomap attempts to preserve the distances between clusters. Due to this feature, Isomap performs better when it has more final components, since the larger the number of dimensions, the higher the likelihood that this distance has been accurately preserved. However, the general poor performance relative to the other techniques perhaps indicates that V-gene segment is not closely related to the underlying geometry of the embeddings.

Since Ablang embeddings represent underlying features of antibodies related to V-gene, it was considered whether a simple neural network could be trained to classify sequence embeddings by V-gene.

3.4.3. Predicting V-gene from sequence embedding

The model was derived from methods available from *Keras*, a python library for machine learning. A simple feedforward neural network architecture was employed, with an input layer, a hidden layer, and an output layer, with two dropout layers. The input layer was a fully connected dense layer with 512 neurons, and uses the ReLU activation function, which takes the input for each neuron and returns either the input or 0, whichever is greater. The dropout layers function to prevent overfitting by randomly ignoring some, in this case 50%, of the neurons in the network. The hidden layer was fully connected, with 256 neurons, and also made use of the ReLU activation function. Finally, the output layer returned the target classes, using the softmax activation function to convert the output scores to probabilities. The model was subsequently compiled using the loss function *categorical cross-entropy* and the adaptive learning rate optimisation algorithm, *Adam*.

This simple model was trained on 45,000 heavy chain embeddings, using one-hot encoded V-gene classes as training labels and a 20% validation split. The model was trained for 10 epochs, and then evaluated on the validation set, as well as a testing set of 5000 sequences held out specifically for testing.

Table 4. Classification report summarising key metrics from the feedforward neural network predicting V-gene ID

	Precision	Recall	F1-score	Accuracy	Tests
Accuracy				0.97	5000
Average	0.97	0.99	0.97		5000
Weighted average	1.00	0.97	0.98		5000

Table 4 summarises the key metrics from the classification model: precision, recall and F1-score. Precision is calculated as the sum of true positives divided by the sum of false and true positives, thus providing a measure of exactness of the classification. Recall is the number of true positives divided by true positives and false negatives, which provides insight into how many positive calls were missed. F1-score assesses both precision and recall, a particularly useful metric for imbalanced datasets. Imbalance is also addressed in the weighted average scores, which are weighted based on class balance. All of these metrics were greater than 97% on the testing set, indicating that a simple neural network could readily classify V-gene identity using heavy chain sequence embedding as features.

3.5. Implementation of Ankh

3.5.1. Clustering embeddings by dimensionality reduction and KMeans

When interrogated by t-SNE and UMAP, Ankh embeddings visually appeared to segment based on V-gene (Figure 7), although the separation was significantly less clear than the same projection using AbLang embeddings (Figure 6).

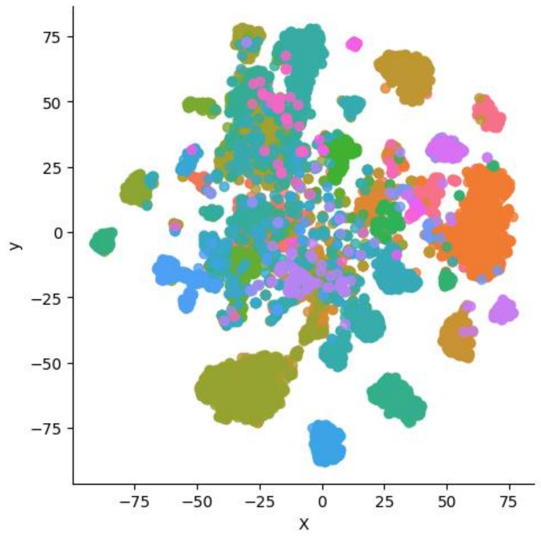

Figure 7. Dimensionality reduction of Ankh heavy chain sequence embeddings using t-SNE and coloured by V-gene segment.

This visual observation was supported following analysis by KMeans clustering, which returned clusters with significantly lower purity than the AbLang clusters (Table 5).

Table 5. Highest cluster purity for each dimension reduction model as calculated by KMeans. Dimensions reduced to two components with t-SNE and UMAP while Isomap was tested at two and four components.

K clusters	Cluster purity	Dimensionality reduction model	Number of components
36	0.6132	UMAP	2
36	0.6021	t-SNE	2
36	0.5352	Isomap	4
34	0.4448	Isomap	2

Despite the differences between the two language models, cluster purities of greater than 50% indicate that Ankh understands some underlying antibody structure to a degree. To see whether this purity was significant enough to train a machine learning model, a V-gene classifier was generated, using Ankh heavy chain sequence embeddings as features, and the V-gene ID as a class label to be predicted.

3.5.3. Predicting V-gene from sequence embedding

The neural network architecture was copied from the AbLang V-gene classifier, with the features being the only element that was changed. Despite the comparatively impure KMeans clustering on the feature compressed embeddings, the V-gene classifier still reached an accuracy of 90% (Table 6). The precision was high, at 89%, indicating that the false positive rate was low. Recall stood at 79%, indicating that the false negative rate was higher than the false positive rate. The weighted average of these scores, however, improves all metrics significantly.

Table 6. Classification report summarising key metrics from the feedforward neural network predicting V-gene ID

	Precision	Recall	F1-score	Accuracy	Tests
Accuracy				0.9	5000
Average	0.89	0.79	0.76		5000
Weighted average	0.92	0.9	0.9		5000

The weighted average is a useful way to assess classifier performance on imbalanced datasets, since the performance of a classifier on the minority class might not provide a representative measure of the model's overall performance. Instead, weighting the average ensures that the accuracy of prediction on classes that are very heavily represented is proportionally reflected in the model performance metrics. Figure 8 shows the heatmap for the confusion matrix of predicted versus actual classes. This heatmap visually helps to explain why the weighting significantly alters the performance metrics in the classification report. Some clusters e.g. 16, 37, 39, are very heavily represented, and the model is able to predict these with a high level of accuracy and relatively few miscalls.

That the average differs from the weighted average reflects a poor performance of the model on outliers, a problem that could be overcome with a larger training data set.

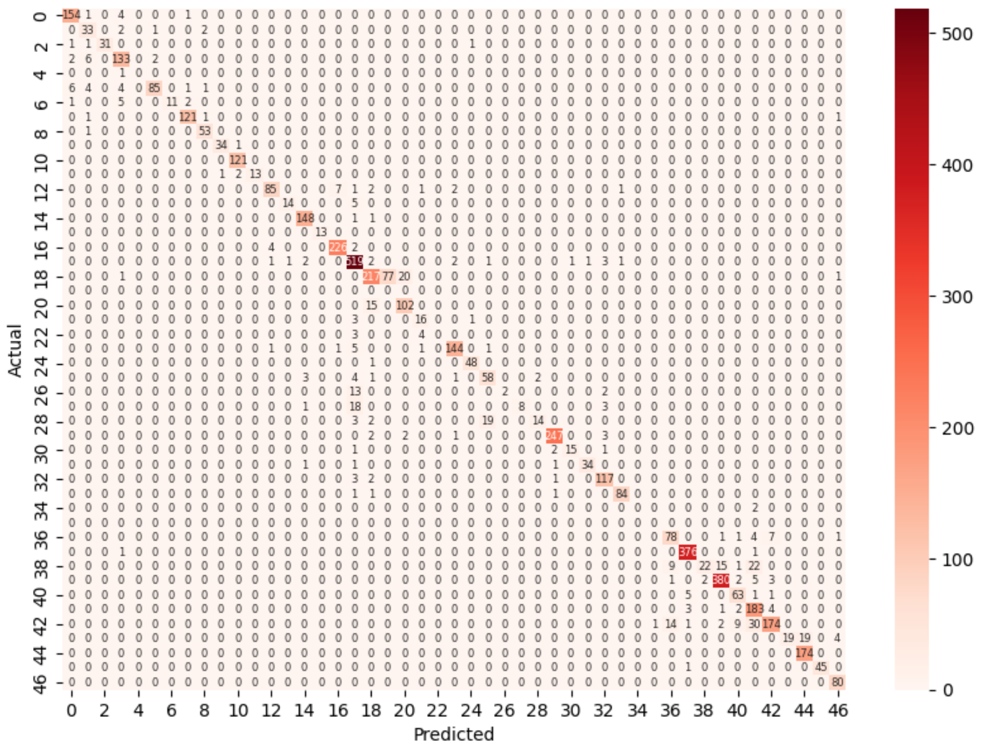

Figure 8. Heat map of a confusion matrix for V-gene classifier following prediction of class labels for 5000 heavy chain antibody sequences.

3.6. Generating MAPT scores

The ability of both AbLang and Ankh to encode an underlying property of antibodies in their embeddings led to the natural question of what other properties could potentially be encoded. Certain biophysical properties can be derived by modelling antibody structures, and subsequently extracting detail relating to surface exposed amino acids. This modelling is computationally expensive, and for predicting aggregation propensity it is the current state of the art. To generate data for machine learning this method was leveraged on antibody paired sequences. The subsequent property derivation was then used as a target value for training a suite of models to predict aggregation propensity given an antibody sequence embedding.

The deep learning framework ABodyBuilder2 was used to model 100,000 paired antibody sequences at an approximate model production rate of one 3D structure every 10 seconds for 11.5 days. This modelling process and subsequent surface hydrophobicity score derivation is described in Heads et al.[31] but was carried out at UCB BioPharma using proprietary code under patent filing, and as such has not been detailed in this project.

3.7. Predicting MAPT score from sequence embeddings

Having generated the aggregation propensity scores using MAPT, sequence embeddings for the same set of antibodies were used as features for transfer learning to a neural network. Since aggregation propensity is a continuous value, the neural network was built for regression. Sequence embeddings for both Ankh and AbLang were used as features for performance-based comparison of a protein versus antibody specific language model.

MAPT generates aggregation propensity values at two pHs and with the variable region in the context of two different antibody constant regions. For simplicity, the most relevant condition, IgG1 constant region at pH 7.4, was selected for prediction.

3.7.1. AbLang sequence embedding prediction

As described in an earlier section, aggregation propensity is measured across both the heavy and light chains. Since AbLang produces either heavy chain or light chain sequence representations, but not both together, the heavy chain and light sequences were embedded separately, and then concatenated, to produce one sequence embedding with dimension 1536. Concatenated embeddings for 45,000 sequences were fed in as features into a multilayer perceptron (MLP) composed of a single fully connected layer, together with the sequence aggregation propensity scores. For clarity, this MLP architecture is referred to as MLP-simple.

Table 7. Performance metrics for each neural network architecture trained on AbLang sequence embeddings and tested on held-out data.

Model	R^2 score	MAE	RMSE	Pearson
Simple	0.854	0.289	0.385	0.93
Complex	0.856	0.274	0.383	0.928
KFold	0.865	0.265	0.37	0.933
Optimised	0.904	0.211	0.3	0.952

The model yielded an R^2 score of 0.854 and a RMSE of 0.385 on the hold-out test set of 15,000 sequences. The MAE of 0.289 indicates the model deviates from the real values by around 0.3 units on average. These metrics quantify the ability of the model to capture the true relationship between sequence and aggregation propensity. The high R^2 score and low RMSE indicate the model fits well to the test data. Pearson's correlation coefficient of 0.93 further demonstrates the strong positive association between predicted and actual values.

Model complexity was increased in an attempt to improve the model predictions. Two more hidden layers were added in, with 256 or 128 neurons. Activation remained as ReLU for both hidden layers, but batch normalisation was added in after each layer to ensure that the outputs of the layers are normalised and standardised. This model architecture is referred to as MLP-complex. Despite the increased complexity, model performance only marginally improved. Further marginal gains were made by using the same model complexity, but by training the model iteratively on different folds of the dataset. The split was kept at 80:20 training to validation, but five models were trained, each using a different fold of the data. Using this KFold strategy, the entire training set was sampled by the sum of the five models. Taking the model consensus of resulted in reduction of MAE from 0.274 to 0.265.

Hyperparameters were then tuned to determine how much more improvement could be extracted from the three hidden layer model architecture. Batch size was optimised first, with batches ranging from 16 to 1024. Model performance correlates with increasing batch size but begins to plateau after 128 (Figure 9), with an optimum reached at 512.

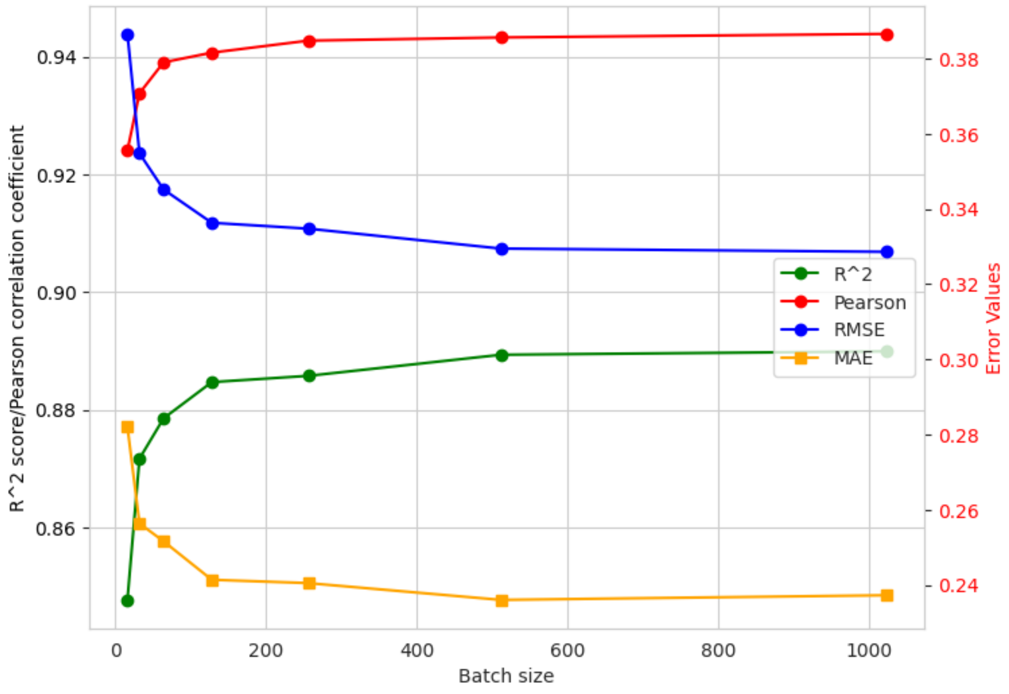

Figure 9. Batch size optimisation of an MLP (MLP-complex) trained on AbLang embeddings and MAPT scores. Seven MLPs trained for 20 epochs with variable batch sizes (x-axis). Regression parameters as a function of batch size plotted on left-hand y-axis (R2 score and Pearson correlation coefficient) and right-hand y-axis (root mean squared error (RMSE) and mean absolute error (MAE).

To tune the number of neurons and the dropout rate in each hidden layer, the hyperparameter optimisation tool, Optuna, was employed. This tool enabled simultaneous optimisation of these two hyperparameters, resulting in an optimal number of neurons and dropout rate for all three layers being reached after 31 trials. Incorporation of these hyperparameters, together with the optimised batch size, and increasing the epochs to 100, resulted in a significant increase in model performance. R^2 increased to 90%, with MAE and RMSE reducing to 0.211 and 0.3 respectively.

Figure 10 shows the predicted values for the optimised model plotted against the ground truth values for each sequence in the hold out test set. Whilst the majority of the sequences have aggregation propensity values between -1 and 2, the maximum values are above six. These ground truth extremes are also predicted by the model as extremes, with the predicted values correlating linearly with the ground truth values.

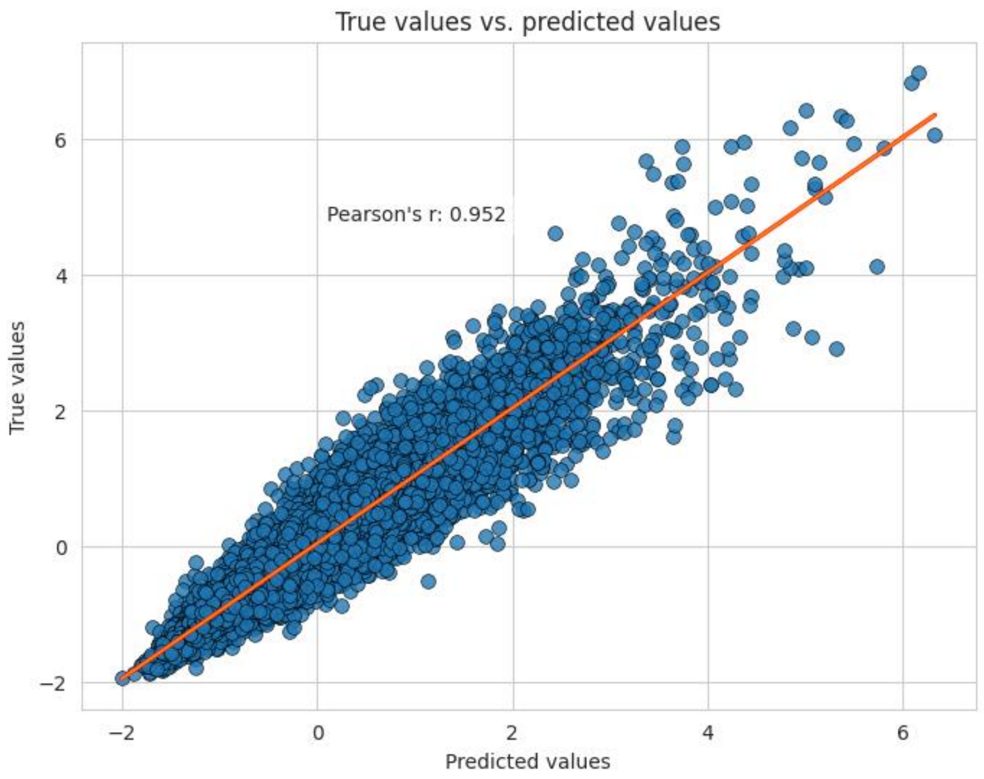

Figure 10. Optimised model predictions of aggregation propensity for 10,000 unseen sequences plotted against ground truth values with line of best fit shown in orange and Pearson correlation coefficient (Pearson's r) shown in line in graph (0.952). Optimised model is AbLang sequence embedding-trained MLP with hyperparameters tuned for optimal performance.

3.7.2. Ankh sequence embeddings prediction

Unlike AbLang, Ankh does not require an antibody to be split into heavy and light chains. As such, heavy and light chains could be represented together in the same sequence embedding with embedding dimension of 1536. The same neural network architectures used to predict aggregation propensity with AbLang sequence embeddings was trained on Ankh embeddings. These values are summarised in Table 8. As with the AbLang embeddings, Ankh embeddings resulted in high performance metrics, including an R^2 score of greater than 85%, and an MAE of 0.279, decreasing to 0.27 with MLP-complex, and further still to 0.256 with the KFold consensus model.

Table 8. Performance metrics for each neural network architecture trained on Ankh sequence embeddings and tested on held-out data.

Model	**R^2 score**	**MAE**	**RMSE**	**Pearson**
Simple	0.859	0.279	0.372	0.93
Complex	0.861	0.27	0.369	0.934
KFold	0.877	0.252	0.347	0.94
Optimise 1	0.862	0.284	0.368	0.939
Optimise 2	0.87	0.263	0.357	0.933

Model optimisation was carried out to improve the performance of the model. Optimisation was carried out with Optuna, using a three-layer neural network with variable hidden units and a fixed dropout rate after the first two hidden layers as the model to optimise. This architecture was used following initial tests that suggested this framework performed better than a simple fully connected layer. However, after 100 trials the optimal parameters did not lead to a significant improvement in model performance (Table 8), with a marginal improvement in R^2 score and RMSE but a poorer outcome for MAE compared to the simple and complex models. These observations indicate that while the optimised regressor may be handling outliers better than the other models (as indicated by R2 and RMSE), for the bulk of predictions the optimised model is making larger errors than MLP-simple and complex models.

To rectify this, a second optimisation experiment was carried out, this time loosening the architecture, and enabling Optuna to test multiple layers, hidden units, dropout rates and learning rates. This optimisation led to a moderate increase in model complexity, with the final optimised model containing three layers, each with a unique dropout rate, a learning rate, and a leaky ReLU activation for the first hidden layer. This model performed better than the previous three models, although notably was not as fit as the KFold consensus model.

The increase in predictive performance with the KFold consensus model led to the question of how much data is needed before predictions start to converge. This is an important question since derivation of biophysical parameters other than aggregation propensity do not currently have computational solutions. As such, experimental measurement is still the only way of establishing these properties. To create a functioning computational model a certain amount of training data would be needed, but the data volume that would be required is an unknown. To obtain a rough estimate, the MLP-complex architecture was trained from scratch iteratively on different volumes of data, ranging from 10 to 45,000 sequences with their respective aggregation propensity values. The regression metrics for the resultant models are summarised in Figure 11.

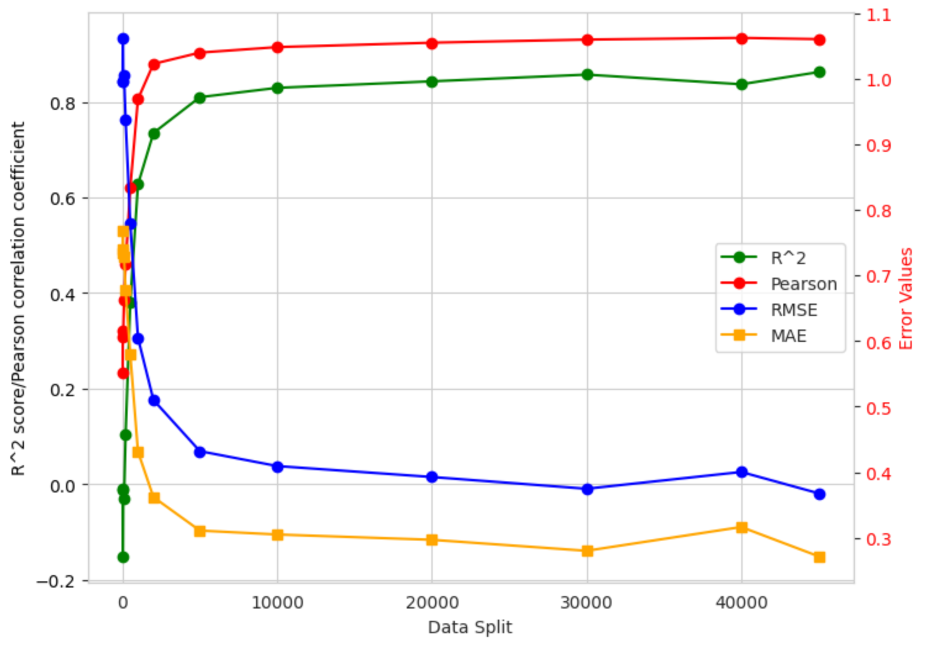

Figure 11. Volume of data plotted as a function of either R^2 score/Pearson correlation coefficient (left-hand y-axis), or error values (right-hand y-axis).

Convergence was attained after training on a dataset of 5,000 samples, with an R^2 score of greater than 0.8 and an MAE of less than 0.3. Evaluation was carried out on a dataset of 15,000 samples, as with the previous models.

3.8. Using ConvBERT for regression with Ankh

As discussed above, compressed sequence embeddings were sufficient features to train MLPs with high accuracy. However, a lot of granularity is lost upon compression from per-amino acid to sequence embeddings. For each sequence representation, a 2D tensor is initially generated that is then averaged along the sequence length dimension to yield a 1D sequence embedding. Since 2D arrays resemble image data, a convolutional neural network would be a suitable model to process this data and sample the context rich per-amino acid embeddings. As a result, after reaching a plateau of model performance for MLPs trained on Ankh sequence embeddings, the Ankh convolutional top model for regression was implemented to obtain further improvements to predictive performance. Due to resource limitations, access to a single Nvidia A100 GPU for less than 10 hours, the smaller Ankh base model was used.

Figure 12 shows the decrease in training and validation loss as training step increases. Notably, the validation loss did not start to increase prior to the end of the training.

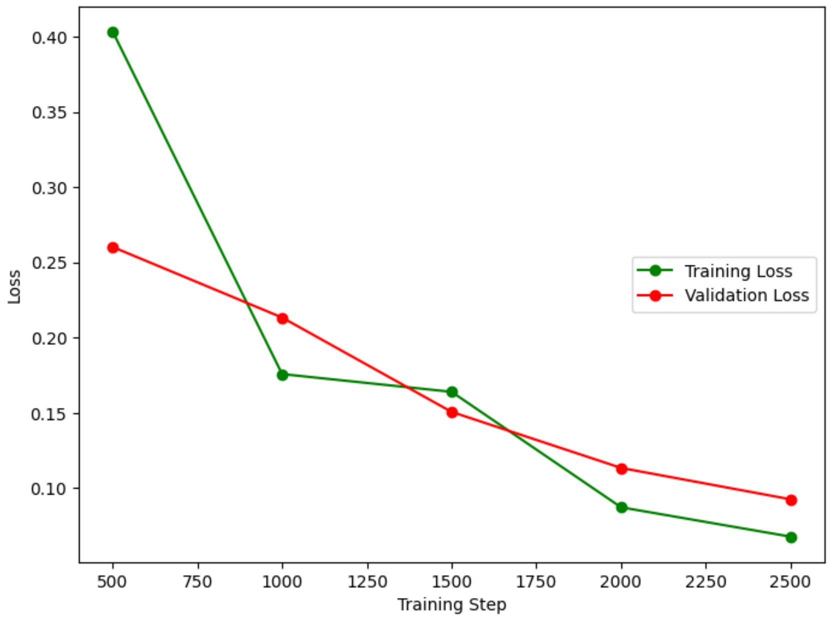

Figure 12. Training and validation loss during ConvBERT training on Ankh embeddings plotted over training steps.

After three epochs of training, the ConvBERT model achieved an R^2 score of 0.916 on the held-out test set, outperforming both the best Ankh-MLP (R^2 = 0.877) and AbLang-MLP (R^2 = 0.904) (Table 9). The mean absolute error (MAE) decreased to 0.215 from 0.263 and 0.211 for the Ankh and AbLang models respectively. Root mean squared error also improved from 0.347 to 0.291 with the more sophisticated model.

Table 9. Comparison of best performing models from Ankh and AbLang, showing regression performance metrics including R^2 score, means absolute error (MAE), root of the mean squared error (RMSE) and Pearson's correlation coefficient.

Regression metric	Ankh ConvBERT	Ankh Optimised 2	AbLang Optimised
R2 score	**0.916**	0.877	0.904
MAE	**0.215**	0.252	0.211
RMSE	**0.291**	0.347	0.3
Pearson	**0.957**	0.94	0.952

Figure 13 shows the strong correlation between predicted and actual aggregation propensity values for the test set sequences following training of ConvBERT. The linear fit closely approximates the ideal y=x line. This close fit is reflected in the high Pearson's correlation coefficient of 0.957 between predicted and true values.

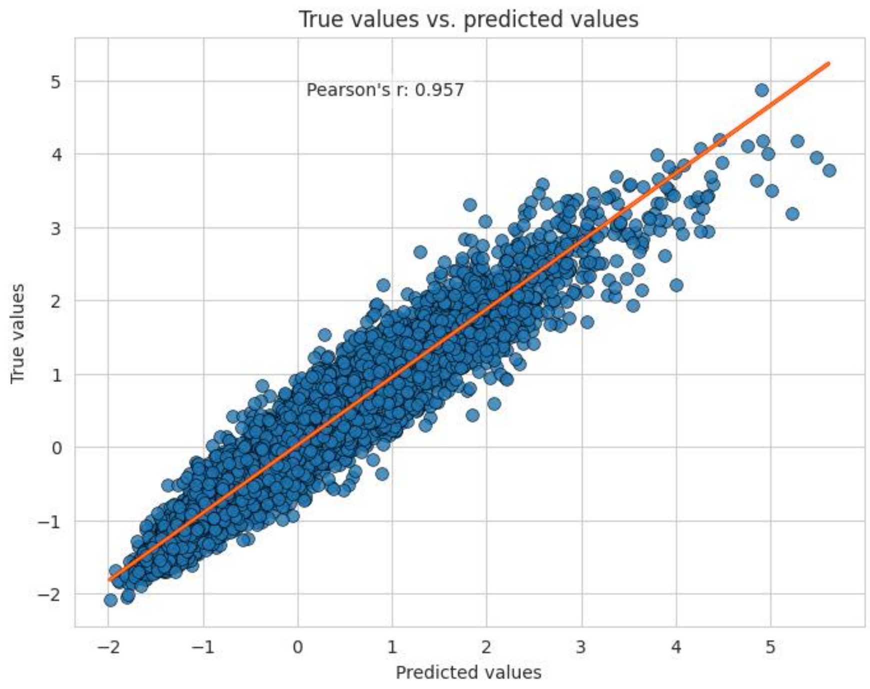

Figure 13. Predicted values after ConvBERT training compared to ground truth values. Line of best fit shown in orange and Pearson's correlation coefficient shown in text in graph.

4. Discussion

4.1. Data extraction and processing

The aim of this project was to build a regressor that could predict aggregation propensity from amino acid sequence using a protein language model. To achieve this, 1.4 million paired human antibody sequences were obtained and clustered. The program MMSeqs2 was used for clustering based on sequence similarity. By setting the similarity threshold to 65%, sequences were clustered into ~100,000 antibody sequence groups. From each cluster, the central representative sequence was selected. This clustering and representative selection approach ensured that as much of the diversity in the full dataset was captured within the smaller 100,000 sequence subset. If simple random sampling had been used instead, rarer sequences that lie at the extremes may have been inadvertently lost.

MMSeqs2 provides speed at the cost of some sensitivity. As such, it is possible that the diversity within this dataset is not at a maximum, however, it should be sufficient to ensure that the training data maintained good representation of the overall diversity. Although not carried out in this project, absolute similarity could have been tested by carrying out a sequence alignment on 1,000 sequences. Alignment algorithms such as ClustalW and Muscle may have been used to carry this out to determine how reliably MMSeqs2 sampled the diversity in the data.

4.2. Manifold learning and V-gene classification

As part of initial exploratory work into protein language models, sequence embeddings for antibody heavy chains were generated, using both AbLang and Ankh, and clustered and visualised using different manifold learning techniques. V-gene was selected as a feature to visualise, since segmentation on this feature using dimensionality reduction was presented as an application in the AbLang publication.

Both the antibody-specific AbLang and general protein language model Ankh were highly effective for V-gene classification, achieving testing accuracies above 90%. AbLang performed slightly better, which is expected given its specialized pretraining on antibody sequences. The high performance indicates these models have learned useful sequence representations encoding V-gene information. This provides a baseline validation of their utility for antibody property prediction tasks.

4.3. Modelling and MAPT score generation

Deriving MAPT aggregation propensity scores for 100,000 sequences was computationally intensive, taking over a week on a high-performance computer cluster. The initial modelling was carried out using ABodyBuilder2, an algorithm that uses deep learning to predict antibody structures. It generates an ensemble of four predicted structures for each antibody to improve accuracy, which further undergoes an additional refinement step using physics-based simulations to fix structural violations. ABodyBuilder2 has comparable structural modelling over AlphaFold-Multimer (the state-of-the-art protein folding prediction algorithm) and achieves the modelling 100 times faster.[29] Nonetheless, modelling 100,000 sequences is computationally time-consuming, and took 11.5 days of GPU computation.

The generated 3D models provided the atomic details needed to accurately calculate the surface hydrophobicity based on surface exposed amino acids and their precise geometries. The hydrophobicity metric was combined with sequence-derived charge scores to yield the aggregation propensity. The sum of these calculations goes beyond what could be inferred from sequence alone, prior to the implementation of this project.

4.4. Sequence embedding with Ankh and AbLang

ScFv sequence embeddings were generated using the 440 million parameter Ankh large model. AbLang by contrast has 14 million parameters, and as such can be run with relative ease on limited hardware. Larger models with more parameters require more computation per sequence, and indeed, the Ankh large model frequently exceeded the memory capacity of the A100 GPU when computing sequence representations. The memory requirements were compounded by the processing of ScFv sequences, which are double the size of either the heavy or light chain sequences alone, since longer input sequences require more computation to generate embeddings. Additionally, the embeddings generated by the larger model did not lead to significant improvements in the aggregation propensity prediction task when used as features for training compared to AbLang embeddings.

The Ankh large model trades off efficiency for representation power – it can generate richer embeddings but has higher resource requirements. The base Ankh model, with 450 million parameters, sacrifices some performance for greater efficiency than the large version, and could have been used to generate sequence embeddings. Given that the richer embeddings with the large model did not result in significant prediction performance increase over AbLang embeddings, there seemed little merit in comparing the base model sequence embeddings. However, Ankh base was subsequently used for generating per-residue embeddings for training the ConvBERT top model. Since per-residue embeddings contain an additional dimension along the sequence length, the size and memory requirements to store and process these are significantly higher. Hence, an Nvidia A100 GPU was needed to train ConvBERT.

4.4. MAPT score prediction

Both AbLang and Ankh sequence embeddings proved excellent features for predicting antibody aggregation propensity, achieving test R^2 scores above 0.8 and low error rates with an MLP composed of a single fully connected layer.

Notably, concatenation of the AbLang heavy and light chain embeddings did not obviously impact the ability of the model to predict aggregation propensity, compared to Ankh-MLPs. The context of the heavy and light chain together may be less important than initially hypothesised, or this context may add marginal predictive ability given a language model trained on paired antibody sequences. A second possibility is that the light chain and heavy chain only embeddings contain sufficient structural encoding that the interacting regions between heavy and light chains is implicit, and thus embedding *in tandem* is not necessary. Encoding of heavy and light chains separately using Ankh could be carried out as a fairer test to compare *in tandem* versus concatenated single chain embeddings.

Models trained on embeddings were hyperparameter tuned for optimal performance. Interestingly, while the Ankh-MLP performed better than an AbLang-MLP with the MLP-simple architecture, increasing complexity and hyperparameter tuning did little to improve Ankh-MLP, while AbLang-MLP demonstrated significant improvements with batch size increase, increase in model complexity, and dropout rate optimisation. Ankh-MLP seemed much more prone to overfitting, since hyperparameter tuning often resulted in model performance below MLP-simple models. AbLang sequence embeddings may be more suited to the task and less prone to overfitting since they capture more of the underlying antibody structure, thus enabling higher performance with more tuning and complexity. However, despite the issues with hyperparameter tuning, the strong performance of the general protein language model Ankh indicates this task may not require antibody-specialised representations. Indeed, ConvBERT trained on 10,000 Ankh per-residue embeddings and their respective MAPT scores outperformed the most accurate AbLang-MLP.

The comparable performance of Ankh and AbLang is perhaps not surprising in hindsight, since hydrophobicity, and thus aggregation propensity, is a property common to all proteins. The significance of this high rate of prediction is that language model embeddings evidently encode features required for estimating biophysical properties. In addition to this, structural context is evidently embedded in the sequence representations since the model is effectively able to predict surface hydrophobicity of the antibodies from embeddings. That protein language models learn general structural features of proteins is not new [38]–[41], but that these structural features can be extracted so readily to predict biophysical properties has not been as explored in the literature.

A notable feature of antibodies is that they have six short regions of exceptional diversity split across both heavy and light chains that together form the basis of target-binding in antibodies. The structure of the ensemble of these regions is exceptionally difficult to predict[42]–[44], yet this structure plays an important role in aggregation propensity.[45], [46] The ability of regressors trained on either AbLang and Ankh to accurately predict aggregation scores for a diverse set of antibodies implies that the sequence embeddings contain sufficient information for the impact on aggregation of these variable regions to be predicted. However, it is worth emphasising that the models presented in this project are able to accurately predict aggregation propensities from another computational model's predictions. Whilst it has been demonstrated that these biophysical properties are encoded in the latent space of a language model embedding, it is worth bearing in mind that any biases, errors, or inaccuracies resulting from imperfect prediction using the MAPT score prediction tool will be propagated into the embedding-trained neural network. The models generated in this project can thus only ever be as good as the initial predictions made using ABodyBuilder2, and due to the complexity of modelling the variable target-binding regions, there will invariably be some inaccuracies in the 3D structures generated, and thus the subsequent hydrophobicity scores.

The convergence analysis suggests that adequate accuracy can be attained with a surprisingly small training set of just 5000 sequences with their respective aggregation propensity measurements, thus lowering the barrier for generating training data to develop models predicting other antibody properties. Indeed, one of the principal obstacles to generating training data sets for supervised learning applications is that practical experimental data generation is costly, time-consuming, and not high throughput enough. Experimental measurements on 5000 antibodies lies at the upper end of feasibility but is possible given sufficient time.

4.5. ConvBERT and Ankh for aggregation prediction

Using the Ankh top model for the antibody aggregation prediction task improved performance over both the highest performing sequence embedding-based MLPs with only a quarter of the training samples. AbLang and Ankh-MLPs leverage sequence embeddings for predictions. Granularity is lost in the embedding compression where the 2D sequence representation is averaged along the sequence length dimension; this dimension contains information related to amino acid context within the protein sequence, thereby capturing the relationship between amino acids in a protein. ConvBERT samples the much richer 2D residue embeddings, and the convolutions that process the data make ConvBERT well suited to processing these arrays. These richer data sets may explain the ability of ConvBERT to improve on model performance and may also explain the inability to improve upon MLPs trained on Ankh sequence embeddings – the models may have reached a maximum performance based on the compressed features available.

While 10,000 training sequences were sufficient to significantly boost predictive accuracy through ConvBERT, additional paired antibody data could continue to improve performance. Additionally, given the success of training an MLP on AbLang sequence embeddings, training ConvBERT on concatenated AbLang per-residue embeddings may boost performance further still, although the high R2 score already achieved indicates the model is likely approaching optimal accuracy bounds, which are limited by the inherent noise in the initial ABodyBuilder2 computational predictions.

4.5. How does a generalised language model compare to an antibody-specific model?

Whilst the antibody-specific PLM AbLang outperformed Ankh in simpler MLP architectures, both models worked remarkably well for encoding sequence properties relevant to predicting biophysical characteristics like aggregation. Ankh's pretraining on a smaller, higher quality protein dataset may explain its competitive performance despite lacking antibody-specific context. In addition to having a performance edge over Ankh-trained MLPs, hardware requirements to generate sequence embeddings favour AbLang as the preferred model. However, the Ankh ConvBERT model makes both MLPs obsolete, and given access to a GPU and additional training data, the model likely pushes the bounds of reaching a near-optimal predictions.

4.6. Future work and finetuning with IA^3

Both Ablang and Ankh are amenable to fine-tuning. With additional time and resources, both models could be fine-tuned using the aggregation propensity scores to further improve model accuracy, although performance may be close to converging on a global minimum for current models based on their high R^2 scores, particularly with the Ankh top model. Instead, a solution needs to be sought that is able to make use of the limited experimental data available. Since experimental data related to solubility and aggregation could likely be generated on up to 500 antibodies, a model that is able to extract value on a small dataset is required.

Typical approaches to finetuning pretrained models involve adjusting all or many model parameters. When dealing with large models, this refinement becomes computationally intensive and requires huge amounts of disk storage for saving model checkpoints. The (IA)^3 methodology employs so-called parameter-efficient finetuning[47] to address these limitations while also enabling model finetuning on small datasets. This method retains the original pretrained model parameters while only adjusting a limited set of new parameters. Freezing the state of AbLang and Ankh is important, since they evidently capture many relevant sequence properties already. The new (IA)^3 model parameters would thus enable pretrained models to capture task-specific problems without needing extensive data storage requirements and huge computational overheads in the learning phase, since the base language model parameters are not updated.

Indeed, Yang et al. have recently demonstrated this lightweight finetuning for on a pretrained LM for secondary structure prediction.[48] They show that by introducing a small number of new parameters per transformer block and updating those alone during training gives rise to a finetuned model that outperforms existing benchmarks for secondary structure prediction.

5. Conclusion

The purpose of this project was to develop a computationally cheaper method for predicting aggregation propensity than the current state of the art – MAPT. It took ~12 days of GPU computation using MAPT to model and predict aggregation propensities for 100,000 paired antibody sequences. In contrast, generating sequence embeddings with language models like AbLang or Ankh required significantly fewer computational resources, achieving the sequence embedding on the same number of sequences in a matter of hours. All models trained achieved excellent predictive performance indicating that language models do not need to explicitly model the 3D structure, but instead embed useful sequence features in their sequence representations. AbLang's small size makes it fast and lightweight to generate embeddings, and thus is highly suitable to CPU deployment, while a more expensive Ankh-based transfer learning model achieved comparable performance trained on a quarter of the sequences.

This project has demonstrated the utility of language models in feature extraction for prediction of biophysical properties, and should be expanded to cover predictive models beyond aggregation propensity. Self-association, stability, immunogenicity, and liability prediction are all features of antibodies that require significant experimental and computational resource to measure. The framework proposed in this project could be readily applied to facilitate cheaper, faster prediction of antibody properties.

6. Bibliography

[1] S. M. Jain, “Hugging Face,” *Introduction to Transformers for NLP*, pp. 51–67, 2022, doi: 10.1007/978-1-4842-8844-3_4.

[2] R. Chowdhury *et al.*, “Single-sequence protein structure prediction using a language model and deep learning,” *Nature Biotechnology 2022 40:11*, vol. 40, no. 11, pp. 1617–1623, Oct. 2022, doi: 10.1038/s41587-022-01432-w.

[3] Z. Lin *et al.*, “Evolutionary-scale prediction of atomic-level protein structure with a language model,” *Science (1979)*, vol. 379, no. 6637, pp. 1123–1130, Mar. 2023, doi: 10.1126/SCIENCE.ADE2574/SUPPL_FILE/SCIENCE.ADE2574_SM.PDF.

[4] K. Weissenow, M. Heinzinger, and B. Rost, “Protein language-model embeddings for fast, accurate, and alignment-free protein structure prediction,” *Structure*, vol. 30, no. 8, pp. 1169-1177.e4, Aug. 2022, doi: 10.1016/J.STR.2022.05.001.

[5] S. Unsal, H. Atas, M. Albayrak, K. Turhan, A. C. Acar, and T. Doğan, “Learning functional properties of proteins with language models,” *Nature Machine Intelligence 2022 4:3*, vol. 4, no. 3, pp. 227–245, Mar. 2022, doi: 10.1038/s42256-022-00457-9.

[6] T. Bepler and B. Berger, “Learning the protein language: Evolution, structure, and function,” *Cell Syst*, vol. 12, no. 6, pp. 654-669.e3, Jun. 2021, doi: 10.1016/J.CELS.2021.05.017.

[7] A. Rives *et al.*, “Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences,” *Proc Natl Acad Sci U S A*, vol. 118, no. 15, p. e2016239118, Apr. 2021, doi: 10.1073/PNAS.2016239118/SUPPL_FILE/PNAS.2016239118.SAPP.PDF.

[8] N. Ferruz, S. Schmidt, and B. Höcker, “ProtGPT2 is a deep unsupervised language model for protein design,” *Nature Communications 2022 13:1*, vol. 13, no. 1, pp. 1–10, Jul. 2022, doi: 10.1038/s41467-022-32007-7.

[9] “Identifying CDRs by Antibody Sequencing - Rapid Novor.” https://www.rapidnovor.com/identifying-cdrs-antibody-sequencing/ (accessed May 04, 2023).

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

[11] Z. Lin *et al.*, “Evolutionary-scale prediction of atomic-level protein structure with a language model,” *Science (1979)*, vol. 379, no. 6637, pp. 1123–1130, Mar. 2023, doi: 10.1126/SCIENCE.ADE2574/SUPPL_FILE/SCIENCE.ADE2574_SM.PDF.

[12] A. Vaswani *et al.*, “Attention is All You Need.” 2017.

[13] T. B. Brown *et al.*, “Language Models are Few-Shot Learners,” *Adv Neural Inf Process Syst*, vol. 33, pp. 1877–1901, 2020.

[14] S. M. Jain, “Introduction to Transformers,” *Introduction to Transformers for NLP*, pp. 19–36, 2022, doi: 10.1007/978-1-4842-8844-3_2.

[15] “End-to-end Masked Language Modeling with BERT.” https://keras.io/examples/nlp/masked_language_modeling/ (accessed Apr. 16, 2023).

[16] "Transformers Explained Visually (Part 1): Overview of Functionality | by Ketan Doshi | Towards Data Science." https://towardsdatascience.com/transformers-explained-visually-part-1-overview-of-functionality-95a6dd460452 (accessed Apr. 19, 2023).

[17] "Transformers Explained Visually (Part 3): Multi-head Attention, deep dive | by Ketan Doshi | Towards Data Science." https://towardsdatascience.com/transformers-explained-visually-part-3-multi-head-attention-deep-dive-1c1ff1024853 (accessed Apr. 16, 2023).

[18] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, vol. 1, pp. 4171–4186, Oct. 2018.

[19] RaffelColin *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, pp. 1–67, Jan. 2020, doi: 10.5555/3455716.3455856.

[20] A. Elnaggar *et al.*, "ProtTrans: Toward Understanding the Language of Life Through Self-Supervised Learning," *IEEE Trans Pattern Anal Mach Intell*, vol. 44, no. 10, pp. 7112–7127, Oct. 2022, doi: 10.1109/TPAMI.2021.3095381.

[21] Z. Lin *et al.*, "Evolutionary-scale prediction of atomic level protein structure with a language model," *bioRxiv*, p. 2022.07.20.500902, Oct. 2022, doi: 10.1101/2022.07.20.500902.

[22] A. Elnaggar *et al.*, "Ankh ♀: Optimized Protein Language Model Unlocks General-Purpose Modelling," *bioRxiv*, p. 2023.01.16.524265, Jan. 2023, doi: 10.1101/2023.01.16.524265.

[23] A. Kovaltsuk, J. Leem, S. Kelm, J. Snowden, C. M. Deane, and K. Krawczyk, "Observed Antibody Space: A Resource for Data Mining Next-Generation Sequencing of Antibody Repertoires," *The Journal of Immunology*, vol. 201, no. 8, pp. 2502–2509, Oct. 2018, doi: 10.4049/JIMMUNOL.1800708.

[24] T. H. Olsen, F. Boyles, and C. M. Deane, "Observed Antibody Space: A diverse database of cleaned, annotated, and translated unpaired and paired antibody sequences," *Protein Sci*, vol. 31, no. 1, pp. 141–146, Jan. 2022, doi: 10.1002/PRO.4205.

[25] T. H. Olsen, I. H. Moal, and C. M. Deane, "AbLang: an antibody language model for completing antibody sequences," *Bioinformatics Advances*, vol. 2, no. 1, Jan. 2022, doi: 10.1093/BIOADV/VBAC046.

[26] Y. Liu *et al.*, "RoBERTa: A Robustly Optimized BERT Pretraining Approach," Jul. 2019, Accessed: Sep. 10, 2023. [Online]. Available: https://arxiv.org/abs/1907.11692v1

[27] N. C. Wu, L. Dai, C. A. Olson, J. O. Lloyd-Smith, and R. Sun, "Adaptation in protein fitness landscapes is facilitated by indirect paths," *Elife*, vol. 5, no. JULY, Jul. 2016, doi: 10.7554/ELIFE.16965.

[28] Z. Jiang, W. Yu, D. Zhou, Y. Chen, J. Feng, and S. Yan, "ConvBERT: Improving BERT with Span-based Dynamic Convolution," *Adv Neural Inf Process Syst*, vol. 2020-December, Aug. 2020, Accessed: Sep. 03, 2023. [Online]. Available: https://arxiv.org/abs/2008.02496v3

[29] 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,"

Communications Biology 2023 6:1, vol. 6, no. 1, pp. 1–8, May 2023, doi: 10.1038/s42003-023-04927-7.

[30] 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.

[31] J. T. Heads, S. Kelm, K. Tyson, and A. D. G. Lawson, “A computational method for predicting the aggregation propensity of IgG1 and IgG4(P) mAbs in common storage buffers,” *MAbs*, vol. 14, no. 1, 2022, doi: 10.1080/19420862.2022.2138092/SUPPL_FILE/KMAB_A_2138092_SM8158.DOCX.

[32] J. Dunbar and C. M. Deane, “ANARCI: antigen receptor numbering and receptor classification,” *Bioinformatics*, vol. 32, no. 2, pp. 298–300, Jan. 2016, doi: 10.1093/BIOINFORMATICS/BTV552.

[33] K. Mizuguchi, C. M. Deane, T. L. Blundell, M. S. Johnson, and J. P. Overington, “JOY: protein sequence-structure representation and analysis.,” *Bioinformatics*, vol. 14, no. 7, pp. 617–623, Jan. 1998, doi: 10.1093/BIOINFORMATICS/14.7.617.

[34] A. Kovaltsuk, J. Leem, S. Kelm, J. Snowden, C. M. Deane, and K. Krawczyk, “Observed Antibody Space: A Resource for Data Mining Next-Generation Sequencing of Antibody Repertoires,” *J Immunol*, vol. 201, no. 8, pp. 2502–2509, Oct. 2018, doi: 10.4049/JIMMUNOL.1800708.

[35] D. Kobak and G. C. Linderman, “Initialization is critical for preserving global data structure in both t-SNE and UMAP,” *Nature Biotechnology 2021 39:2*, vol. 39, no. 2, pp. 156–157, Feb. 2021, doi: 10.1038/s41587-020-00809-z.

[36] E. Becht *et al.*, “Dimensionality reduction for visualizing single-cell data using UMAP,” *Nature Biotechnology 2018 37:1*, vol. 37, no. 1, pp. 38–44, Dec. 2018, doi: 10.1038/nbt.4314.

[37] A. Saxena, A. Gupta, and A. Mukerjee, “Non-linear dimensionality reduction by locally linear isomaps,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3316, pp. 1038–1043, 2004, doi: 10.1007/978-3-540-30499-9_161/COVER.

[38] J. Vig, A. Madani, L. R. Varshney, C. Xiong, R. Socher, and N. F. Rajani, “BERTology Meets Biology: Interpreting Attention in Protein Language Models,” Jun. 2020, Accessed: Apr. 16, 2023. [Online]. Available: https://arxiv.org/abs/2006.15222v3

[39] J. Leem, L. S. Mitchell, J. H. R. Farmery, J. Barton, and J. D. Galson, “Deciphering the language of antibodies using self-supervised learning,” *Patterns*, vol. 3, no. 7, p. 100513, Jul. 2022, doi: 10.1016/J.PATTER.2022.100513.

[40] Z. Lin *et al.*, “Evolutionary-scale prediction of atomic-level protein structure with a language model,” *Science (1979)*, vol. 379, no. 6637, pp. 1123–1130, Mar. 2023, doi: 10.1126/SCIENCE.ADE2574/SUPPL_FILE/SCIENCE.ADE2574_SM.PDF.

[41] A. Rives *et al.*, “Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences,” *Proc Natl Acad Sci U S A*, vol. 118, no. 15, p. e2016239118, Apr. 2021, doi: 10.1073/PNAS.2016239118/SUPPL_FILE/PNAS.2016239118.SAPP.PDF.

[42] B. Abanades, G. Georges, A. Bujotzek, and C. M. Deane, “ABlooper: fast accurate antibody CDR loop structure prediction with accuracy estimation,” *Bioinformatics*, vol. 38, no. 7, p. 1877, Apr. 2022, doi: 10.1093/BIOINFORMATICS/BTAC016.

[43] J. A. Ruffolo, J. Sulam, and J. J. G. Correspondence, “Antibody structure prediction using interpretable deep learning,” *Patterns*, vol. 3, p. 100406, 2022, doi: 10.1016/j.patter.2021.100406.

[44] M. L. Fernández-Quintero *et al.*, “Challenges in antibody structure prediction,” *MAbs*, vol. 15, no. 1, 2023, doi: 10.1080/19420862.2023.2175319.

[45] S. J. Wu *et al.*, “Structure-based engineering of a monoclonal antibody for improved solubility,” *Protein Eng Des Sel*, vol. 23, no. 8, pp. 643–651, 2010, doi: 10.1093/PROTEIN/GZQ037.

[46] W. Li, P. Prabakaran, W. Chen, Z. Zhu, Y. Feng, and D. S. Dimitrov, “Antibody Aggregation: Insights from Sequence and Structure,” *Antibodies*, vol. 5, no. 3, Sep. 2016, doi: 10.3390/ANTIB5030019.

[47] H. Liu *et al.*, “Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning”, Accessed: Sep. 06, 2023. [Online]. Available: https://github.com/r-three/t-few

[48] W. Yang, C. Liu, and Z. Li, “Lightweight Fine-tuning a Pretrained Protein Language Model for Protein Secondary Structure Prediction,” *bioRxiv*, p. 2023.03.22.530066, Mar. 2023, doi: 10.1101/2023.03.22.530066.