Allergens prediction: Internship memo

Pierre Boyeau

July 2018 - September 2018

Contents

1 Introduction

Context and objectives Proteins are macromolecules defined as sequences of amino-acids. Amino-acids are chemical compounds and 20 kinds of them exist naturally.

Some proteins can be the cause of strong organisms reactions, also called *allergies*. Such proteins have diverse origins like:

- animals (mammals, fish, insects)
- plants (vegetables, fruits, flowers)
- fungus

Being able to successfully predict the allergen character of proteins is crucial. First, determining if a protein can cause allergies is difficult and time-consuming. It also has applications in GMO design and control purposes, as biologists and control agencies both want to know if a GMO can cause allergies *before* it is produced. Detecting allergens using machine learning techniques is the objective of a 3-year project that just started, in which my laboratory is involved in, in cooperation with biologists. As an intern, my objectives were to establish rigorous protocols to evaluate models performance, to find pertinent ways to featurize protein data for allergens prediction and to design and compare models based on existing literature and on our needs.

How to model this problem? Let Ω be the set of all possible amino-acids. Let $X = \{s_i, i \in [\![1, N]\!]\}$ be a set of protein sequences, where $\forall i \in [\![1, N]\!], s_i = x_i^1 x_i^2 \dots x_i^{l_i} \in \Omega^{l_i}$ Our goal is to predict is, given protein sequence s_i to predict $y_i \in \{0, 1\}$ the boolean representing if the protein is an allergen.

This task is not easy. Proteins sequences have varied lengths, from a few dozens to thousands of amino-acids. As one can expect, it is how amino-acids come up one after another that can determine if the protein may cause allergies. Handling various lenghts and capturing the sequential character of proteins sequences are necessary steps in order to successfully predict allergens.

The biologists we worked with were also interested in discovering which specific parts of allergens sequences cause allergies. Such subsequences, called *epitopes* are very hard to find.

2 Data and performance measurements

Data used for this project was given to us by the biologists we worked with. It consists of $\approx 12K$ protein sequences in total, including only $2K$ allergens. The data was manually checked and cleaned. The dataset also gave information regarding which organism (Genre, Species) produced the proteins, allowing us to ensure that proteins (both allergens and non-allergens) came from a variety of organisms.

To measure algorithms performance, we used cross-validation on custom splits. Random splits could be dangerous for several reasons. The major problem with the data is that some pairs of proteins were very similar. For instance, only a few amino-acids would be different in the different sequences. In particular, the influence of proteins of species which only had one label (all positives or all negative) could lead to misleading results. During cross-validation, such instance could increase models performance, simply because the algorithm learned that those similar sequences should have the same label.

For that reason, to avoid this pitfall, we created 10 species-specific splits[1] that were used in cross-validation.

Because this project is just starting, discussions with biologists did not allow us to set up a custom metric that fit their needs. For that reason, and because the data is imbalanced, we used ROC-AUC score as a performance metric.

3 Parallelism of our task with NLP

Our problem statement underlines a lot of similarities between our task and text classification as we work with variable-lenght sequences. Hence, we can wonder what canh be archived by applying methods traditionnally applied to text sequences to proteins. This is what we will focus on in this part.

3.1 Allerdictor

The Allerdictor model described in [1] is one of the rare papers that deal with allergenicity prediction. Instead of representing a sentence by its constituent words, it represents each protein by its *overlapping* k-mers.

k-mers can be seen as some sort of words. Given a sequence $x_i^1 x_i^2 \dots x_i^{l_i}$, they correspond to $\{x_i^j x_i^{j+1} \dots x_i^{j+k-1}, j \in [\![1, N-k+1]\!]\}$. A^k corresponds to the set of all possible k-mers. While all possible k-mers do not appear naturally, the number of k-mers can be very important.

To extract features from these, we can use the frequencies of each k-mer in proteins sequences. While not described in the allerdictor paper, we used instead TFIDF word representation that yielded better results. Either way, these features have fixed-size and do not depend on the protein lenghts, which allow us to work with classifiers of our choice. Because of the high dimensionality and sparsity of this problem, we use a L1-penalized linear SVM model as classifier.

This simple method to classify proteins has a lot of benefits. First, interpreting our model results is very easy because we can interpret the SVM weights as allergen-inducing scores for all k-mers. It is also very easy to implement and to train, and for that reason constitute a great benchmark. On the other hand, this method is handicapped by the fact that it uses fixed-length subsequences. We cannot extract variable length motifs. k is also restrained to take relatively small values, as the number of $k-mers$ rapidly explodes.

3.2 Word2Vec inspired

Another technique applied to classify sentences is to compute a representation of each of its words and to agglomerate them (often through summation) to get a vectorial representation of sentences we can use for classification using the classifier of our choice.

Some works apply this idea to protein sequences [2][3], but not on allergens classification.

[1] Each split contains proteins whose species does not appear in other splits

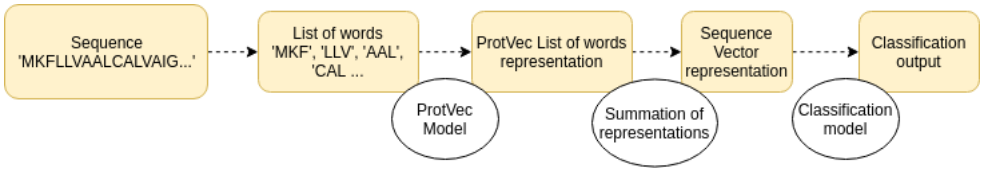

Figure 1: Word2Vec-inspired model

The basic idea is the following:

1. We extract pseudo-*words* from the protein sequence, and we compute their associated vectorial representation
2. These *word* representations allow us to compute a protein representation as a fixed-sized, low dimensional vector.
3. We can use the classifier of our choice based on those features.

Learning k-mer representation "Words" of the protein sequences as the non-overlapping k-mers. To get a meaningful, compact representation of these "words", we use the popular *Word2Vec* model. We will train a neural network on a very big corpus of protein sequences that will learn how to represent k-mers in a vectorial form.

To do so, the architecture of Word2Vec[4] we used is based on the skip-gram architecture. To put into a nutshell, we will predict surrounding k-mers given a k-mer using trainable embeddings of the words. These embeddings are the representations we are interested in, and that we will use to represent k-mers.

Our Word2Vec model was trained using *gensim*[5], and we used the UniProt dataset[6], containing more that $500K$ protein sequences to train this model. Indeed, our allergens/non-allergens dataset is too small to train this network.

Classifying proteins Once Word2Vec trained on UniProt, we can use them for our task. For each protein s in our dataset, let $E(s)$ be the set of all non-overlapping k-mers, and $w2v : w \mapsto w2v(w) \in \mathbb{R}^d$ be the function that associates a vectorial description of each k-mer.

We define the vectorial representation of a protein s as:

$$\sum_{w \in E(s)} w2v(w)$$

and we use Gradient Boosting to classify allergens.

4 Recurrent Neural Networks

4.1 Motivations

As we will show later, previously described methods achieve acceptable performance. That being said, they face by their construction some major obstacles. While they have similar properties, k-mers are not words, and proteins are not sentences. Allerdictor fails at extracting allergenicty-inducing motifs, as we are stuck with fixed-lenghts motifs, and in any case relatively short motifs. In reality, epitopes can have very varied lenghts. Thus, it is difficult to give a proper definition of "words" in our case.

Described methods also fail to take sequence properties into account. They are position-insensitive in the sense that relative positions of k-mers are not taken into account for prediction. Discussions with the biologists also underlined that the mecanisms that caused allergy were very complex and could be linked to a variety of properties, from chemico-physical attributes of amino-acids to the 3D structure of the protein. These properties are not taken into account by previous models.

These observations and the fact that these methods have been successfully used in proteomics [7][8] encouraged us to consider RNN-based methods for our classification task. These works also showed that such methods could be applied even with relatively small datasets such as our own.

As an alternative to LSTM, Temporal Convolution Networks (TCN) have also been experimented, following [9] guidelines. Unfortunately, these networks hyperparameters were difficult to tune, and overall results were not as good as the ones the authors of [9] observed.

4.2 Architecture

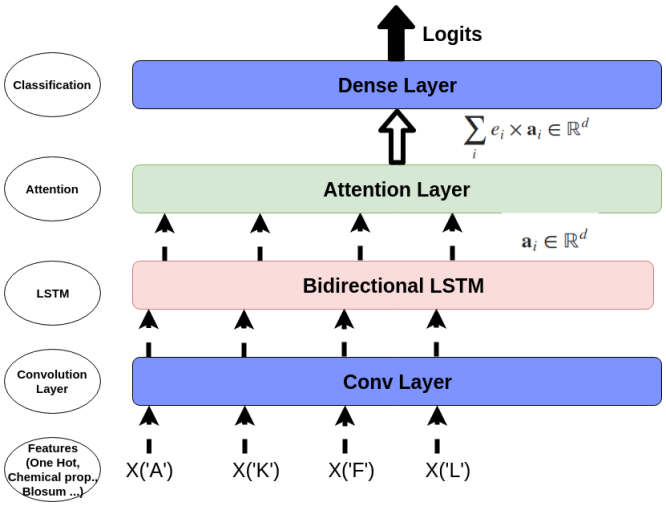

Figure 2: RNN model architecture

Above figure describes the model general architecture, that we will now describe in detail.

Inputs Extracting good amino-acid features plays a crucial part in obtaining good performance with our model. Each protein $s = x^1 \dots x^l$ will be represented by 2 inputs:

- $I_{static} \in \mathbb{R}^{l \times n_{static}}$ is a matrix containing one-hot encoding, physico-chemical properties and other features specific to each amino-acid. The matrix is *static* in the sense that 2 amino-acids, found at different positions in a protein or in different proteins will have the same features. Based on [7][8] and on research conducted on our own, these features include:
  - Amino acids one-hot encoding
  - BLOSUM80 matrix
  - Physico-chemical properties (hydrophobicity, and electric properties, 3D-geometry properties ...) found in several papers [10][11][12]

  As this project is just starting, biologists have not checked in detail these features, but are working on it. While this representation may not be perfect, it is an attempt to give a great overall representation of amino-acids properties.

- $I_{dynamic} \in \mathbb{R}^{l \times n_{dynamic}}$ containing PSI-BLAST [13] PSSM (Position-score specific matrices). Each position features do not depend solely on which amino-acid it corresponds to, but to the overall sequence. PSI-BLAST PSSM are score matrices obtained from alignments of the protein with proteins

in a reference database. These features were added to the model, inspired by [7], as we observed they drove ROC AUC scores up very significantly for our task.

We also set a maximum sequence length so that all sequences have the same length L_{max}. That way, unreasonably long sizes are capped, and we will be able to perform batch training. In practice, shorter sequences are 0 left-padded and we truncate the longer sequences as illustrated below.

We set $L_{max} = 1000$ such that most proteins verify $l < L_{max}$ (97% of proteins have shorter lengths.)

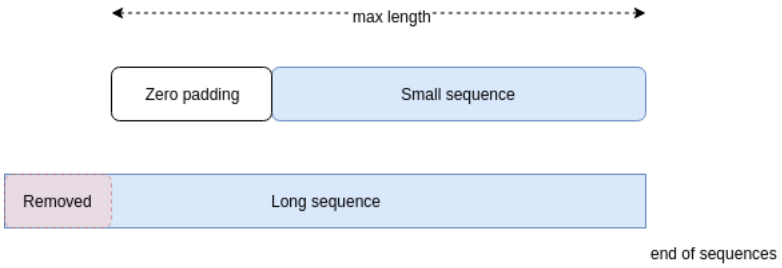

Figure 3: Processing different lengths

Convolutional layer After being concatenated, I_{static} and $I_{dynamic}$ are fed to a 1d-convolutional layer. This layer serves several purposes. First, it allows us to control the dimension of the tensor that will be fed to the LSTMs layers, and thus have leverage on the model complexity. Convolution also help to smooth signal. Second, it helps obtaining better features by taking into account amino-acids surroundings (through convolution). While they could be of interest performance-wise, we did not consider set-ups with more than one layer of convolutions for reasons we will explain later.

Bidirectional LSTM layer This layer is the core of the architecture, as it is the one that allows us to handle to sequential nature of the inputs. Contrary to original RNN, LSTM networks[14] are designed to capture long-term sequence dependencies.

Let $x = [x_1, \dots x_{L_{max}}] \in \mathbb{R}^{L_{max} \times n_{neurons}}$ be the output of the convolutional layer. We will describe the mecanisms that allow LSTM to handle sequences, by describing how a single, forward LSTM works[15]. For each of its cells, at each position t, will iteratively be computed and stored 2 values C_t and h_t, based on previous values computed at position $t-1$. C_t is the cell *state*: it stores a "memory" of what happened at previous positions. h_t is the output of the cell at position t, an hidden representation that we will later use for allergen prediction. These values are updated the following way:

$$\begin{cases} C_t = f_t * C_{t-1} + i_t * \tilde{C}_t & \text{(A)} \\ h_t = o_t * \tanh C_t & \text{(B)} \end{cases}$$

where f_t, i_t, o_t are *gates* that define the way we will update C_t and h_t, and that are defined as:

$$\begin{cases} f_t = \text{sigmoid}(W_f[h_{t-1}, x_t] + b_f) \\ i_t = \text{sigmoid}(W_i[h_{t-1}, x_t] + b_i) \\ o_t = \text{sigmoid}(W_o[h_{t-1}, x_t] + b_o) \end{cases}$$

and where $\tilde{C}_t = \tanh W_C[h_{t-1}, x_t] + b_C$ represents the new candidate values. W_f, W_i, W_o and their respective intercepts are the LSTM cell parameters that will be trained using backpropagation.

Equation (A) shows that we will update the memory based on previous memory (all that has happened so far) and on the new candidate values $\tilde{C}_t$ that takes into account the new contributions brought by new value x_t

This described how a forward LSTM works. At each position, we have at disposal an hidden representation h_t that takes previous positions of the sequence into account... but not take future positions. To get a sequence representation that takes both past and future into account at each timestep, we use bidirectional LSTM. Instead of taking the output of one LSTM layer, we concatenate the ouputs of one LSTM layer taking as input $[x_1, \dots x_{L_{max}}]$ (Forward LSTM) and another LSTM whose input is $[x_{L_{max}}, x_{L_{max}-1} \dots x_1]$ (Backward LSTM).

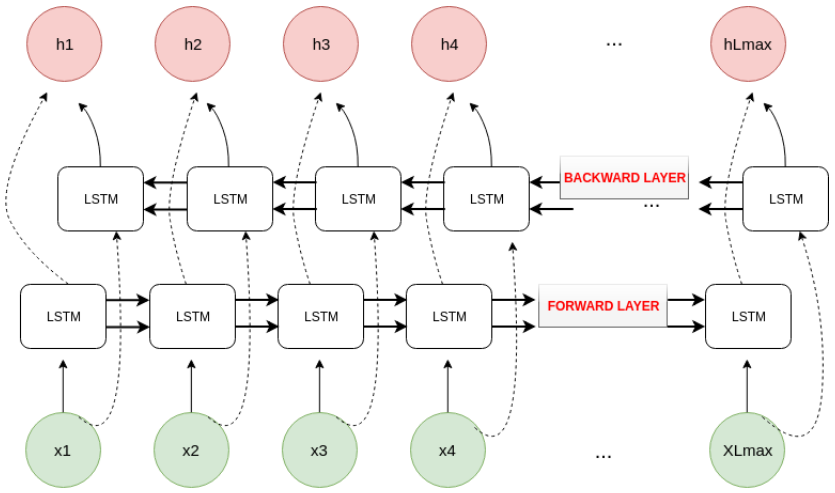

Figure 4: Bidirectional LSTM architecture

Single Bidirectional LSTM were not the only architectures that we tried. Unidirectional LSTM, deeper architectures and the use of other RNN cells, namely GRU cells were also considered. However, their use resulted in poorer results.

Attention layer At this point, the output of the last layers is an hidden representation $h \in \mathbb{R}^{L_{max} \times n_{hidden}}$ where $n_{hidden} = 2 \times n_{cells}$. This hidden representation cannot be directly used for classification, because of its potentially very high dimension, and because it is still positional-dependant. Attention networks[16] are a solution to this problem. To put into a nutshell, they help predict "where to look". Knowing which positions in the allergens sequences had influence in predicting it is an allergen is very valuable for us. We can imagine it could help biologists diagnose a wrongfully labeled protein. Better, it could help them in their work to find epitopes areas in allergens sequences.

Attention mecanism

To know where to look, we learn a function f_{att} that attributes positive, sum-to-one weights to each amino-acid position based on their hidden representation [16]:

$$f_{att} : h_i \mapsto e_i, e_i \in \mathbb{R}_+, \sum_{j=1}^{L_{max}} e_j = 1$$

In practice, f_{att} is a 1-unit dense layer followed by a softmax operation over all positions..

Now that we have these weights at disposal, the hidden representation of the protein we will use to predict its class will be:

$$o = \sum_{i=1}^{L_{max}} e_i h_i \in \mathbb{R}^{L_{max}}$$

Thus e_i can be interpreted as the *a priori* influence of position i to predict if the associated protein is an allergen or not.

Alternative attention mecanism:

Discussions with biologists underlined the fact that they are more interested in finding out if a position is allergy-inducing or not. To get such positions during inference, one solution can be to set a threshold $v \in]0, 1[$ s.t. $e_j > v \implies$ amino-acid i is allergy inducing. However, setting up v was laborious and results (when comparing candidates to ground-truth epitopes locations in allergens) were not very convincing.

For that reason came up the idea to change the attention mecanism in order to make the model make the decision. In other words, explicitely set up $e_i = 0$ To that purpose, f_{att} still is a 1-unit dense layer, this time followed by a tanh activation. Then, we take the maximum between 0 and this output, and make $\{e_i, 1 \leq i \leq L_{max}\}$ sum to one.

While it may look a bit far-fetched, this attention does not decrease performance, and it outputs attention weights more easily usable by biologists. This model also has the nice property to be trainable using backpropagation (compared to *hard attention* [16] for instance)

Performing classification A dense layer applied to o allows us to obtain logits. We use cross-entropy loss to train the network.

5 Meeting with biologists

In the last part of my internship, I had the chance to meet the biologists team to discuss the project advancement. This meeting was extremely helpful for several reasons. Many modelisation choices were made (regarding the data properties, and to define performance metrics for instance). Experts point of view was needed to validate our assumptions. As they will be the ones using the algorithm, it was also important for us to understand their needs to adapt our models and our protocols in consequence.

Their remarks also gave us new ideas. In proteomics, many high-quality and big datasets are at disposal. For that reason, we had tried to apply transfer learning techniques to our RNN model without success: we trained proteins' superfamily RNN classifiers on such datasets, and used obtained weights to initialize our allergens RNN classifier, without significant improvements. The experts made us realize that this task was not very pertinent, as these 2 tasks do not have much in common. Instead, we learned that 3d structure prediction was a much better task as it is one of the main cause for allergies.

6 Results

6.1 Algorithms performance

For the models we described previously, we applied the following protocol. We proceeded to a 10-Fold Species-Specific Cross Validation as described before. For each split, the different models hyperparameters were tuned in the following way:

- For Allerdictor and Word2Vec-based models, hyperparameters were selected by cross-validation using the training data (Nested Cross-Validation)
- As our RNN model is longer to train, we could not afford to perform nested cross-validation. For that reason, a validation set, regrouping all proteins with specific classes, was extracted from each training split, and hyperparameters we selected were the ones that maximized the validation's ROC AUC.

The results below show the obtained results of described protocol:

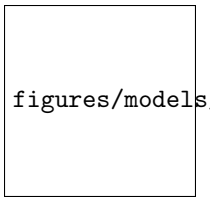

Figure 5: Models performance comparison

6.2 RNN attention interpretability

In this section, we focus on determining if protein areas categorized as important by the attention mechanism of our RNN model conveys meaning and can be explained. [16] shows an impressive computer vision application in which parts of images whose activation weights are high correspond to pertinent image areas. Do our attention weights have similar properties?

To answer this question, biologists gave us information on epitopes location in several allergens we had in our dataset. This data shows that no assumptions can be made regarding how long, and how many subsequences are responsible for allergy in allergens. For that reason, it is not possible to expect allergy-inducing areas to have a specific shape. This observation conforts us that Allerdictor extracted k-mers have limited use for our task.

Based on this data and to evaluate our attention mecanisms, it is difficult to create a metric to judge our attention pertinence for 2 reasons:

- Epitope data is partial. In other words, in annotated allergens could exist other epitopes areas that have not been discovered yet.
- We currently have a small number of proteins for which we have epitope information

Nevertheless, we display some of the attention weights we have obtained using the new attention. In the figure below and in annex are plotted side by side attention weights and ground-truth epitopes' positions (in white).

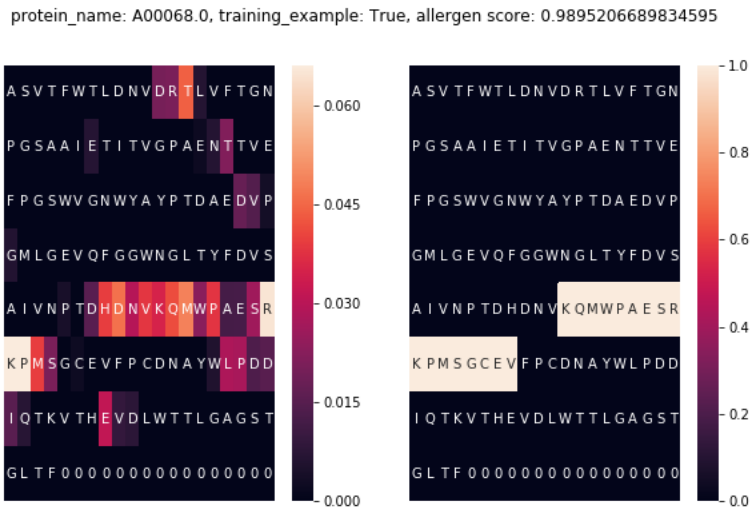

Figure 6: Attention weights (left) vs real epitopes locations

These examples seem to indicate that even if our model had no information on what subsequences are responsible for allergy, regions it focuses are not random and can correspond to epitope locations. That being said, further study of results by biologists will be needed to confirm this statement.

References

[1] Ha X Dang and Christopher B Lawrence. Allerdictor: fast allergen prediction using text classification techniques. *Bioinformatics*, 30(8):1120–1128, 2014.

[2] Ehsaneddin Asgari and Mohammad RK Mofrad. Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11):e0141287, 2015.

[3] Patrick Ng. dna2vec: Consistent vector representations of variable-length k-mers. *arXiv preprint arXiv:1701.06279*, 2017.

[4] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[5] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. http://is.muni.cz/publication/884893/en.

[6] Rolf Apweiler, Amos Bairoch, Cathy H Wu, Winona C Barker, Brigitte Boeckmann, Serenella Ferro, Elisabeth Gasteiger, Hongzhan Huang, Rodrigo Lopez, Michele Magrane, et al. Uniprot: the universal protein knowledgebase. *Nucleic acids research*, 32(suppl_1):D115–D119, 2004.

[7] Rhys Heffernan, Yuedong Yang, Kuldip Paliwal, and Yaoqi Zhou. Capturing non-local interactions by long short-term memory bidirectional recurrent neural networks for improving prediction of protein secondary structure, backbone angles, contact numbers and solvent accessibility. *Bioinformatics*, 33(18):2842–2849, 2017.

[8] Søren Kaae Sønderby, Casper Kaae Sønderby, Henrik Nielsen, and Ole Winther. Convolutional lstm networks for subcellular localization of proteins. In *International Conference on Algorithms for Computational Biology*, pages 68–80. Springer, 2015.

[9] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[10] JEAN-LUC FAUCHÈRE, Marvin Charton, Lemont B Kier, Arie Verloop, and Vladimir Pliska. Amino acid side chain parameters for correlation studies in biology and pharmacology. *International journal of peptide and protein research*, 32(4):269–278, 1988.

[11] Jack Kyte and Russell F Doolittle. A simple method for displaying the hydropathic character of a protein. *Journal of molecular biology*, 157(1):105–132, 1982.

[12] Akinori Kidera, Yasuo Konishi, Masahito Oka, Tatsuo Ooi, and Harold A Scheraga. Statistical analysis of the physical properties of the 20 naturally occurring amino acids. *Journal of Protein Chemistry*, 4(1):23–55, 1985.

[13] Stephen F Altschul, Thomas L Madden, Alejandro A Schäffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402, 1997.

[14] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[15] Christopher Olah. Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: 2018-09-12.

[16] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, pages 2048–2057, 2015.

Annex

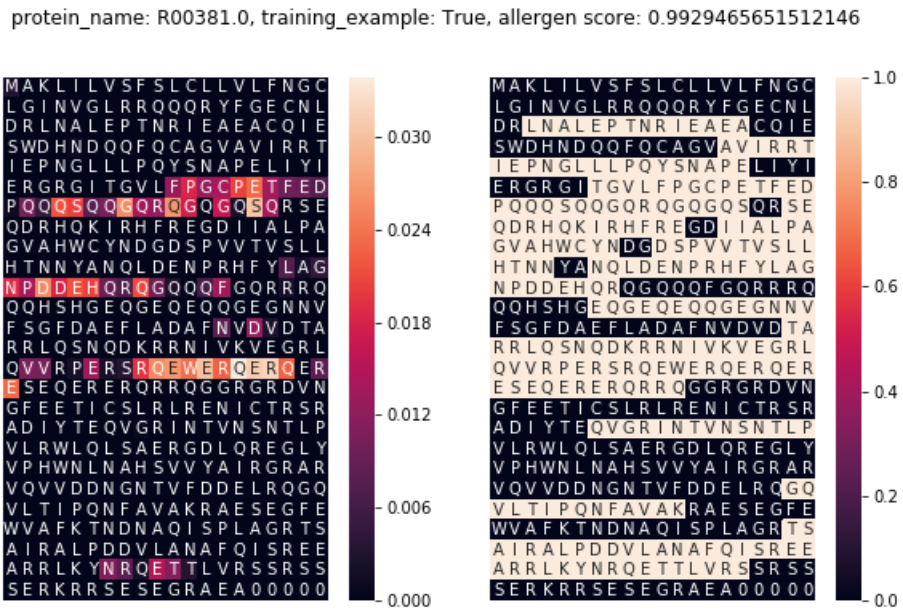

Figure 7: Attention weights (left) vs real epitopes locations

Figure 8: Attention weights (left) vs real epitopes locations