Predicting RNA-Binding Preferences for Proteins

Millie Dwyer mmd378 **Julie Helmers** jch609 **Shasha Lin** sl4964 **Lihan Yao** ly880

Abstract

In this paper, we explore different methods for representing the RNA Recognition Motifs (RRMs) of certain proteins to aid biologists with the discovery of protein-RNA matches. We aim to learn representations that preserve RNA-binding preference similarity. We use deep learning encoder-decoder methods to generate low dimensional representations of aligned and unaligned RRM sequences. We ultimately find that our methods perform better than non-machine learning methods for predicting sequence similarity and also produce similar results for both aligned and unaligned sequences. This paper is Team Viserion's final report for the Fall 2017 Capstone course, DSGA-1006, at NYU's Center for Data Science. All code for this project may be found in our GitHub repository at https://github.com/NYU-CDS-Capstone-Project/Viserion.

1 Introduction

RNA Recognition Motifs (RRMs) are substructures of some RNA-binding proteins that specify which RNA sequences the protein will bind. Because it is easier to identify the RRM specific substructure of a protein than it is to determine binding affinity, for most identified RRMs it is unknown which RNA sequences they will bind, out of the tens of thousands of possibilities. However, due to similarities in biological structure and evolution, there are many RRMs that are homologous, meaning that they share sequence patterns as well as binding behaviors. If it is possible to generate a representation of these similarities from the RRM sequences themselves, and relate this representation to RNA binding preferences, it may be easier to predict how a newly identified RRM will behave. This approach would be faster and less expensive than experimentally determining behaviors of individual RRMs. From a broader perspective, an informative RRM representation will lead to more accurate predictions of RNA binding preferences, and may help researchers to better understand RRMs' role in gene expression. Through this capstone project, our goal is to learn reliable and predictive low dimensional representations of RRMs through deep learning techniques, especially those inspired by natural language processing, in order to predict RRM RNA-binding preferences.

2 Objective

The goal of our project is learning a meaningful mathematical representation of RRM sequences of amino acids. The quality of the representations are evaluated qualitatively, through tSNE visualizations, and quantitatively, through regression analyses. We also rely on the domain expertise of our advisor, Professor Quaid Morris, for insights into our representations.

3 Data

Due to the small number of known binding matches between RRMs and RNA, our RRM sequence data is broken into labeled and unlabeled datasets, pulled from different sources. Throughout this paper, "unlabeled" will refer to RRMs that do not have known associated RNA binding preferences, while "labeled" will refer to RRMs that do. In general, the RRM sequences are combinations of 20 letters, each of which represents a different amino acid, and are at maximum around 90 letters long.

3.1 Unlabeled Sequences

Identifying RRM sequences is easier for biologists than identifying which RRMs will bind with which RNAs. The RRM domain can be identified within a protein sequence based on its structure and location. Because of this imbalance in difficulty between the identification of RRMs and RNA preferences, we have a largely unlabeled dataset. We specifically have approximately 99,000 sequences with unknown binding preferences. This unlabeled data is pulled from the online database PFAM [9], which provides aligned RRM sequences of length 551. Each datum consists of an ID/Name and Sequence. The ID/Name generally matches the name of the protein associated with the RRM. We provide more details about aligned sequences below.

3.2 Labeled Sequences

For a small number of RRM sequences, the distribution of binding affinity over RNA sequences is known for the proteins associated with the RRMs. These matches are generally identified through manual testing, and are known for approximately 200 RRM-containing proteins. Biologists have discovered that RRM sequences with at least 70% sequence similarity have nearly identical binding preferences, which expands this "labeled" dataset. Our labeled dataset contains 407 proteins with known affinity expressions. This data was provided by our advisor's lab at the University of Toronto and includes protein sequences and z-score distributions over RNA sequences. These related affinities were generated using the RNAcompete method [12] defined by Ray et al.

Because this dataset was obtained from a different source than the unlabeled data, the alignments differed and did not directly translate between datasets. We discuss how we accounted for this issue in the sequence alignment section below.

While we did use our labeled sequences to supplement our unlabeled dataset for autoencoder training, we also used this dataset for judging the success of our learned representations at capturing structural similarities in RRMs.

3.3 Sequence Alignment

Because RRM sequences are not consistently of the same length, it is possible that learning a representation without aligning the sequences could incorrectly capture illusory local structure across RRMs. Sequence alignment provides global structure across RRM sequences of different lengths by aligning similar groups of amino acids into similar positions within the sequences. For example, two RRMs may differ only in that one has an extra "loop" of amino acids in the middle of the sequence. These sequences would seem quite different without alignment, but quite similar once the matching sets of amino acids are aligned between the two RRMs. The alignment process allows for gaps to be placed in the sequences when certain RRMs are missing a given amino acid, ensuring that important positional information about the other amino acids within the sequence will not be lost.

The most common method of sequence alignment in biological applications uses hidden Markov models (HMM) to fit a sequence of amino acids to a profile. However, for any given set of sequences, the generated alignment may differ. This depends on how similar the sequences are, and how sparse their representations are. For our data, the alignments differed between our unlabeled and labeled sequences, so we had to make some adjustments in order to combine both data sources.

We used an additional alignment software program, Clustal Omega [17], to generate a dataset with a unified alignment for both our labeled and unlabeled sets. This program expanded the length of our sequences to 605 from the original sizes of 551 and 324 for the unlabeled and labeled datasets, respectively.

We now provide an example of the same RRM before and after the alignment process. T117424||RNCMPT00259_RRM__0 is an identifier for the RRM. Everything that follows it is the sequence itself.

- Sample Aligned RRM Sequence:
  $T117424||RNCMPT00259\_RRM\_\_0-----TPSTNVFINY--IP---------P--RF--------$
  $----------T----------E----QD----------L-R---N--------$
  $----------L-------CS-Q---Y--------------G---------$
  $-------E--I-IS-----------------------------S----K--------$
  $---$
  $----------IM-----$
  $---$
  $----------I-NL---E---------TG----QSKCFG---F----------V-$
  $----------K----------F---------R-------E---------L---------$
  $--S------Q---------A--H--------A--------A--------I-QA--------$
  $---I----D---G----M------SIGN-------K-----R--LLAKYAESQE----$

- Sample Unaligned RRM Sequence:
  $T117424||RNCMPT00259\_RRM\_\_0$
  $TPSTNVFINYIPPRFTEQDLRNLCSQYGEIISSKIMINLETGQSKCFGFVKFRELS$
  $QAHAAIQAIDGMSIGNKRLLAKYAESQE$

3.4 Data Processing

We processed our aligned data further by first filtering to exclude any positions that contained gaps in all of the sequences, then filtering to exclude sequences which still contained a significant percentage of gaps (more than 99%)in the remaining positions. A gap is denoted by the dash symbol "-", noting the absence of an amino acid. After the first step, our sequences of length 605 were shortened to length 81. This drastic decrease in length is due to the nature of alignment data: A majority of positions in the center of all sequences contain no meaningful symbols due to the HMM alignment process. See Appendix A for the distribution of amino acids versus gaps by position.

For the second step, we removed sequences in which more than 30% of the remaining 81 symbols were gaps. We believe learning a representation while including sequences with significant numbers of gaps would not only introduce marginal information into the representation, but deteriorate overall representation quality. By setting the gap percentage threshold at 30%, we filtered out 41.5% of our dataset, with 58,500 sequences remaining.

4 Methodology

Because we had such a large dataset of unlabeled data and so few labeled data, we approached this project with unsupervised methods. The problem of discovering a meaningful latent representation of RRM sequences resembles some Natural Language Processing (NLP) tasks, in which neural network models learn a low dimensional embedding of words, where distance represents different semantic properties. Therefore, all of the approaches we tried were heavily inspired by the NLP literature.

Although aligned sequence data are typically used in biology because of the challenges discussed in the sequence alignment section, we applied our methods to both the aligned and unaligned versions of our RRMs, in the hope of achieving similar results that would support the generalizability of our models to both types of data.

4.1 Seq2Vec

Seq2Vec [11] is an application of the Doc2Vec embedding method to biological sequences. Both Seq2Vec and Doc2Vec generate a distributed representation of a sequence of "words," where each sequence is embedded according to its document summary vector in a context prediction task. A

natural challenge, and where the analogy between our biological domain and NLP weakens, is the choice of "words" in an RRM sequence, which contains no inherent segmentation between its elements (amino acids) and no clear segmented structure. However, it is common practice in the literature to use a sliding window of width three, where at a given position $[w_i, w_{i+1}, w_{i+2}]$, the window contents form a three-symbol "word" which is then embedded. Our implementation follows this practice, and feeds the resulting three-mers into a Doc2Vec model via the Gensim Python package.

4.2 CNN + LSTM

To emulate state-of-the-art NLP methods, we experimented with two different model architectures that combine a Convolutional Neural Network (CNN) encoder with an LSTM (Long Short Term Memory) decoder to learn latent space representations of the RRM sequences. In the original proposal and early conversations with our advisor, we considered using a principal components analysis (PCA) module in addition to the CNN and LSTM. The goal of the PCA step would have been to capture global structure in the sequences. However, after further discussion, we decided to omit the PCA, as layers in the CNN or LSTM can achieve the same global results as needed. Below we describe in depth our two CNN + LSTM architectures: a ResNet implementation and a character-level autoencoder.

4.2.1 ResNet + LSTM

We used an architecture similar to a ResNet encoder and LSTM decoder previously applied in image captioning [6]. Introduced in [13], Residual Network (ResNet) is a family of CNN architectures with skip connections between layers, thereby replacing the task of learning the latent representation $\mathcal{H}(X)$ with the task of learning the residual representation: $\mathcal{F}(X) = \mathcal{H}(X) - x$ (x is the original data), which is easier to train and optimize. As a type of CNN, ResNet is capable of capturing complex high-level features that are invariant to their location in the data. This is in contrast to our decoder, which uses LSTM [14] to capture sequential information in the data.

In an image captioning setting, the ResNet is used to extract high-level features from images. Those features are then used as the context vector for a downstream language modeling LSTM, which generates target words (the caption) through time, conditioned on both the image context vector and the caption word prior to the current word. We have taken the same type of architecture and applied it to our RRM modeling: RRMs are first run through a 41-layer ResNet CNN, the output of which is fed into a downstream LSTM as a context vector. At each time step, the LSTM unit learns to predict the next amino acid in the RRM sequence, conditioned on the context vector and the previous amino acids. Figure 1 shows the end-to-end structure of this model.

As an aside, the architecture here is exactly the same as the 50-layer model proposed in [13], except that we did not have the last three bottleneck blocks. We did not use all 50 layers, because we think the complexity of our task is significantly less than that of the ImageNet task.

The primary difference between our task and the image captioning task is that in our case, the inputs for the CNN and LSTM come from the same data source, a single RRM sequence, whereas the inputs come from the image and from a human-generated caption, respectively, in the original task. The justification for this adapted architecture, however, is that the LSTM should learn to model RRM sequences in general (i.e. learning the statistical mapping that generates RRMs), whereas the CNN should learn local structure that is specific to each particular RRM sequence, thereby helping the LSTM predict the next amino acid based on the particular case of the current RRM. Another difference is that in the image captioning task, the ResNet is pretrained on the ImageNet dataset [18], whereas in our task, the ResNet and LSTM are trained end-to-end at the same time.

Once this model has been trained, there are two possible representations of the RRMs that we can extract: The hidden vector generated at the last time step of the LSTM and the feature context vector generated by the CNN. We evaluated both of these representations (see Evaluation section). Using

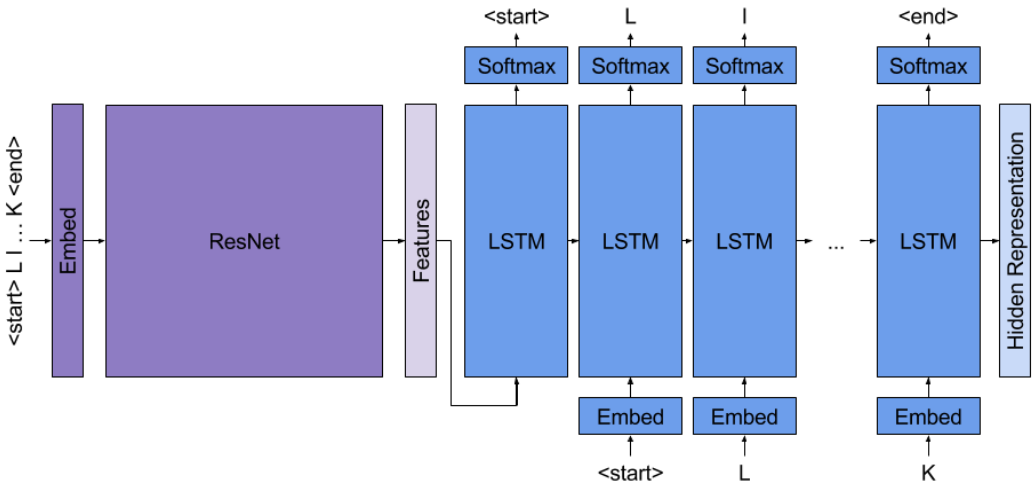

Figure 1: Diagram of the architecture of our version of this model.

the former representation, which we have dubbed the "decoder-side," intuitively makes sense, as this representation should not only take RRM-specific features into consideration (as the CNN encoder does), but also should have learned general sequential information (as the LSTM does). However, we also looked at the "encoder-side" representation from the CNN, because it represents a more compact version of the input sequence, and likely should also contain some sequential dependency information because it has been trained in tandem with the decoder.

4.2.2 Character-Level Neural Machine Translator

For our second CNN + LSTM architecture, we implemented a character-level Neural Machine Translation (NMT) model as proposed in [1]. This NMT method, originally intended for translating natural language, can be generalized to encode and decode symbol sequences with character-level morphemes with no fixed vocabulary, such as RRMs. After training our implementation of the NMT, shown in Figure 2, the feature maps of the model are activated by biological motifs. These motifs are typically local features of a protein sequence that are 3 to 8 amino acids in length; see Filter Motifs under Results. The Bidirectional Gated Recurrent Unit (BiGRU) layer encodes the entire sequence according to its motif activation pattern. The long-term dependencies learned by the BiGRU layer may encode higher-dimensional structures of the protein sequence, such as alpha helices and beta coils, that provide the protein's three-dimensional shape. Structures like these appear in the sequence with semi-periodicity and over long distances, for which the BiGRU is well suited.

5 Training Details

For each of the models above, we sought to find an optimal fit for our unlabeled training data. Each model had its own training framework based on literature and heuristics surrounding the model type.

5.1 Seq2Vec

Instead of producing a representation after an encoding process, Seq2Vec generates a distributed representation of sequences. Following a context window prediction task, Seq2Vec is given a context three-mer (three consecutive amino acid letters) and utilizes a sequence vector v_s to predict surrounding three-mers within the context window via maximizing its log likelihood distribution over the training set. v_s is randomly initialized and then concatenated to the three-mer context vector, and it is trained simultaneously with the three-mer embedding (i.e. words in our NLP analogy) during the prediction task. After training, v_s is a numerical representation of the sequence.

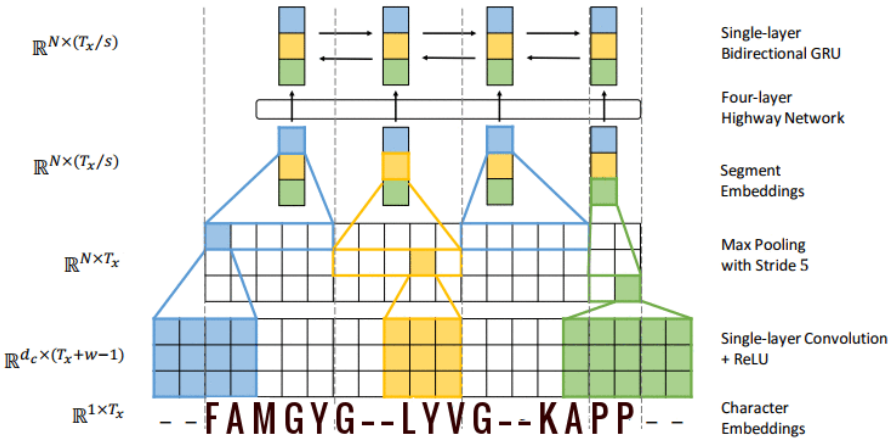

Figure 2: Diagram of character-level NMT taken from [1], with biological sequence input

5.2 ResNet + LSTM

The ResNet + LSTM model is trained by stochastic gradient descent with a batch size of 64. We used cross-entropy between the sequence predicted by the LSTM and the original sequence as the loss function:

$$L(y, \hat{y}) = \frac{1}{N}\sum_{n=1}^{N}[y[n]\log(\hat{y}[n] + (1 - y[n])\log((1 - \hat{y}[n])]$$

We trained with the Adam optimizer [15], which is a method of stochastic optimization that is efficient and robust to noise in data. We experimented with a decaying learning rate versus a constant learning rate throughout training. The decaying learning rate provided better results, as evaluated from our tSNE visualizations (more details provided under TSNE Evaluation). Our final model had a constant learning rate for the first five epochs, and decayed by a factor of a quarter for each consequent epoch, for a total of 14 epochs. We used early stopping, i.e. we stopped training when the loss on a validation set did not improve for 10 batch iterations, to make sure that our model did not overfit on the training set. We also used teacher forcing during training for the LSTM decoder. Teacher forcing, a heuristic from NLP models, involves feeding the true symbol into the network at each time step during training, instead of the predicted letter.

5.3 Character-Level NMT

Similar to the ResNet + LSTM model, the character-level NMT model uses a batched training process and gradient descent to learn its weights. As proposed in [1], the character-level NMT model uses negative log likelihood as its loss function. Our implementation calculates cross-entropy loss for computational purposes (the cross-entropy objective is equivalent to minimizing the negative log likelihood of softmax), with variations that utilize teacher forcing to various degrees in the decoding process.

6 Model Design Choices

After creating the initial structure of the above models using PyTorch, we sought to further adapt them to our biological application through hyperparameter tuning and architecture modifications.

6.1 Seq2Vec

Two significant design decisions for Seq2Vec are choosing whether to segment the RRM sequence into overlapping or non-overlapping three-mers and choosing between two context prediction tasks,

paragraph vector-distributed memory model (PV-DM) and paragraph vector-distributed bag-of-words (PV-DBOW). PV-DM and PV-DBOW are closely related tasks presented in [2], where a paragraph vector is used in the prediction of words within a context window. In PV-DM, paragraph vectors are concatenated with word vectors as input to predict the next word; in PV-DBOW, the paragraph vector is the only input to predict a random word chosen from a context window. (In our case, a context window is a three-mer of amino acids.)

We chose to use non-overlapping three-mers over overlapping three-mers due to the superior performance as demonstrated in [11]. We trained both PV-DM and PV-DBOW models, but selected PV-DBOW for our final model, as it out-performed PV-DM. Other hyperparameters, such as context window size, learning rate and decay, and negative sampling were chosen according to performance on the similarity regression task.

6.2 Resnet + LSTM

The size of the feature vector from the ResNet and the size of the hidden vector from the LSTM are both tunable hyperparameters: We used 64 and 128, respectively, which gave us good performance. Given more time, we would experiment with other embedding sizes.

6.3 Character-Level NMT

We explored adding noise to the input of the Character-Level NMT. The corruption procedure we tried chooses five random symbols throughout the sequence and then swaps those symbols with an adjacent symbol. Therefore, in addition to the reconstruction task the NMT is already performing, the model must also recover the original sequence from an encoded representation of the corrupted version. From examining the resulting representation's t-SNE visualizations, we observed tighter clusters when using this noise, with fewer points which did not belong to a defined cluster. When considering the impact of this additional denoising task on our similarity regression results (see Similarity Regression under Evaluation Methods), we observed a decrease in the variance of the loss across trials. Note that the RNA samples for which similarity regression must predict binding affinity vary between trials. This suggests that a representation learned via denoising may be more robust against data variation within the downstream similarity task.

Teacher forcing was explored but not used in the final model. While teacher forcing speeds up convergence, the resulting representations performed considerably worse when used in similarity regression.

7 Evaluation Methods

As discussed above, our high-level goal was to learn reasonable latent representations of RRMs. We employed both qualitative (tSNE) and quantitative (downstream regression) approaches to evaluating the reasonableness of our representations.

7.1 tSNE Evaluation

We used t-Distributed Stochastic Neighbor Embedding (tSNE) [16] as a way to evaluate the quality of our learned RRM representations. RRMs of the same group should cluster together in high dimensional space, and so should their low dimensional (two- or three-dimensional) tSNE embeddings. Contrary to other dimensionality reduction techniques, such as principal component analysis (PCA), tSNE focuses on keeping similar data points in high dimensional space together in the low dimensional mapping, instead of focusing on keeping dissimilar ones apart. This difference makes it more suitable for visualizing data that lie on high dimensional non-linear manifolds, as we expect our biological data to.

As the unlabeled data from PFAM [9] is noisy — many sequences included are suspected RRMs that have not been clearly identified by biologists — we chose to plot the learned representations of just a small, hopefully reliable subset of the data. After matching RRMs with their corresponding gene IDs using the UniProt protein database [10], we verified gene IDs using the GeneCards human gene database [4], and selected a subset of sequences from the dataset that are known to be true RRMs.

7.2 Affinity Regression

Affinity regression is a supervised learning method developed by Pelossof et. al. [5] to learn binding relationships between proteins and RNAs through a bilinear function. Because the learned association is a linear relationship between a protein and RNA sequence, applying affinity regression with a low dimensional representation instead of the original sequences reduces the number of weights included in this model, allowing better results with our very small labeled dataset. This is particularly true given that the original aligned sequences are of length 551 (or 605 after realignment), while we have a labeled dataset of only 407 RRMs. Training a model with the original representation would certainly overfit to our data, and likely would not generalize to any novel RRM sequences.

As RNA-binding proteins can contain one to four different RRM domains, before affinity regression, the low dimensional representations of RRMs belonging to the same protein are averaged to get the representation of the corresponding protein.

We generated our affinity regression results by sharing our representations with members of our advisor's lab, who have trained affinity regression for similar applications. However, one downside of using affinity regression is computational time – affinity regression requires a low dimensional representation of the "ground truth" RNA affinity matrix, which for a given protein will contain over 16,000 features, each of which is an RNA. Acquiring this low dimensional representation is generally done using singular value decomposition (SVD), a method that has a high computational cost when the dimension of data is large. At the time of this paper, we have not yet obtained affinity regression results for all of our models' representations. However, we present the results of an alternative downstream regression task, similarity regression, below.

7.3 Similarity Regression

In the course of this project and discussions with our advisor, we realized that affinity regression may not be the best way of judging the predictive power of our representations. Our alternative method, which we are calling similarity regression, more directly compares the similarities of the low dimensional representations from our models (for both labeled and unlabeled sequences) and the similarities of binding affinities (for labeled sequences).

Figure 3 explains the structure of the similarity regression task. The original RNA affinities in our labeled dataset are expressed over 16,382 potential RNA matches for each protein. Because this high dimensionality would likely lead to computational issues, we reduce the dimensionality by approximating the principal components of our affinities by taking

$$YY^T$$

where Y is the matrix of our known affinities with dimensions of dataset size $\times$ number of potential matches. The weights of a matrix W are then learned to translate the input H, which contains the learned representations, into a representation matching YY^T. This ultimately leads to learning the weights of the matrix W by minimizing:

$$||HWH^T - YY^T||_2$$

After training, we want the weight matrix W to be able to transform any given low dimensional representation $\hat{H}$ into its RNA binding affinities, given that the representation is of a sequence that

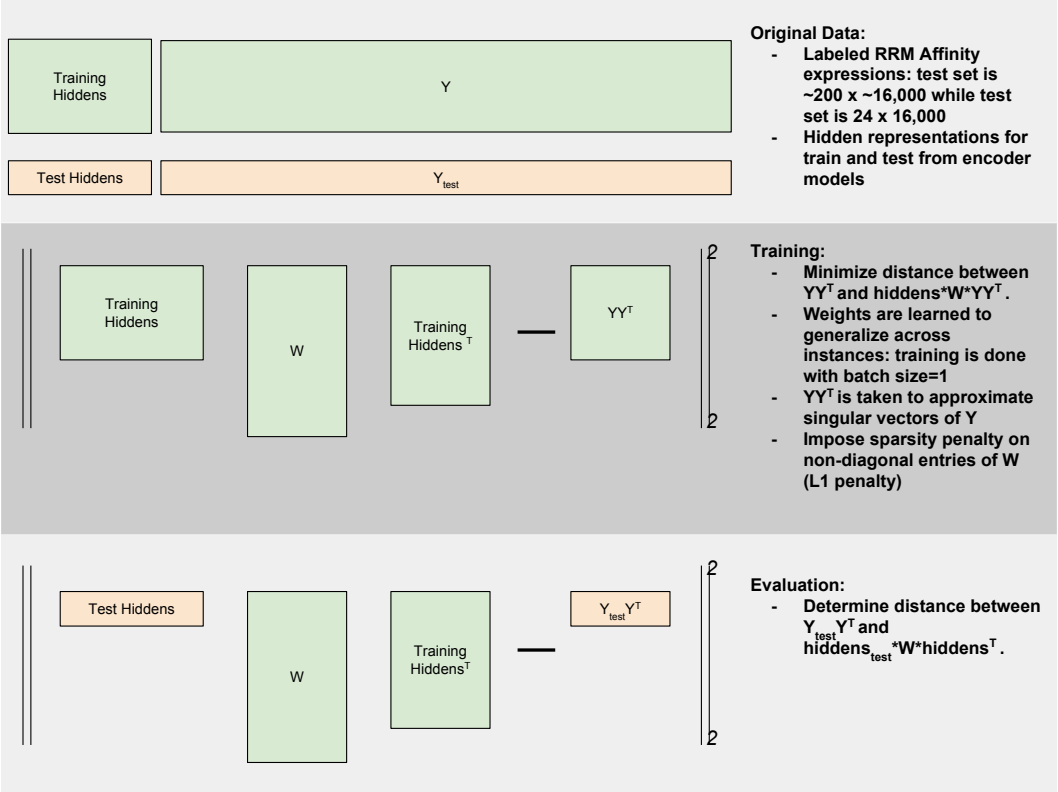

Figure 3: Diagram explaining similarity regression

is similar to ones in our labeled dataset. This could be a powerful tool for predicting the binding behavior of novel RRMs.

8 Results

8.1 tSNE Results

As described above, we obtained distributed representations of RRMs from Seq2Vec and latent representations from our two CNN + LSTM models. We then mapped RRMs to gene ID using the matching tool provided by UniProt[10], before selecting a verified RRM subset to plot based on GeneCards [4]. Proteins containing these verified RRMs are represented using the sum of their one to four constituting RRMs' representations.

Figure 4 shows the learned two-dimensional mappings of the protein representations from our model architectures. Each point represents a single protein, and the different colors indicate the gene ID corresponding to that protein, such as RBM17, RBM19, SRSF1, etc. We obtained two separate representations from our ResNet + LSTM model, one from the output of the ResNet, and the other from the last hidden vector of the LSTM. Those representations are titled "CNN Encoder Features" and "LSTM Hidden State," respectively, in Figure 4. For both the ResNet encoder and LSTM representations, Figure 4 includes tSNE projections for the representations learned from both the aligned and unaligned input.

We find clear clusters emerging along gene ID delineations, suggesting that our representations are appropriately dependent on the gene associated with a particular protein/RRM. All three of our models are capable of clearly separating different protein groups, with the ResNet encoder representation more clearly doing so than the LSTM decoder representation for the ResNet + LSTM network.

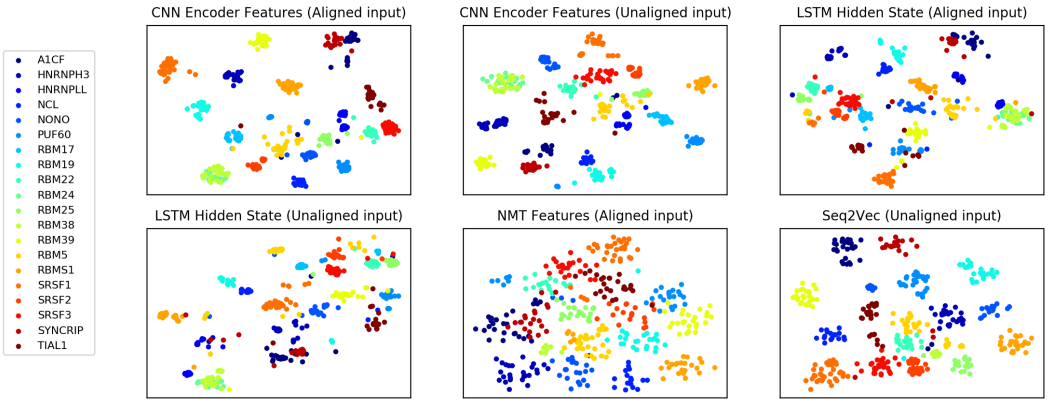

Figure 4: tSNE visualizations

Some RRMs that are can be difficult for biologists to distinguish, such as SYNCRIP and HRNPH3, are well-separated by the ResNet encoder for the unaligned input, as well as by Seq2Vec, while being less separated by other models. Overall, it's evident that the high dimensional representations learned by each model, including those trained on the unaligned input, capture essential information about different types of RRMs.

8.2 Filter Motifs

For our character-level NMT model, we further evaluated the learned motifs extracted from each convolutional filter. For each filter that had a substantial number of activations, Table 1 lists the three most frequent motifs that activate them, as evaluated on a holdout set of 1,000 samples.

n-mer 1	**Activation Count**	**n-mer 2**	**Activation Count**	**n-mer 3**	**Activation Count**
F	142	Y	56	L	15
GF	49	GY	23	AF	4
GFG	34	GYG	24	GYA	16
LYVG	20	IFIK	12	LYVK	10
—LYVGDL	13	—IFIKNL	10	—LYVGNL	9
——FVGQ	14	—-IFVGD	12	—-LFIGM	8

Table 1: Motif activations for active filters in the NMT

Specific n-mers shown here, such as "GFG," have appeared in mammalian protein sequence studies; with more domain expertise on interpreting biological n-mers, we may be able to explain our models' encoding behavior in more detail.

By examining activation statistics across filters, we find that groups of filters in the NMT model have been trained to detect identical motifs. This is evidence that fewer neurons at the convolutional layer could encode the necessary information as efficiently, since there seem to be filters serving redundant roles. (The current filter bank contains 40 filters per filter width, with widths varying from one to nine.)

8.3 Similarity Regression Results

Table 2 shows the results of applying similarity regression with 5-fold cross-validation to the labeled dataset. On each of the five folds, similarity regression was run against different validation sets of the same size. We computed both the mean and standard error of the losses across different runs. Here, the training and validation losses are root mean square error (RMSE) values for the distance

between our predicted RNA affinities and the actual affinities for each protein sequence:

$$RMSE(y, \hat{y}) = ||y - \hat{y}||_2$$
$$= \sqrt{(y[0] - \hat{y}[0])^2 + (y[1] - \hat{y}[1])^2 + \cdots + (y[n] - \hat{y}[n])^2}$$

Bold font in Table 2 indicates best performing models.

The naive k-mer representation serves as a untrained baseline for our trained models. k-mer representations, similar to n-grams in NLP, are an approach that biologists use to approximate the similarities of two sequences. We used four-mers, which is standard for running affinity regression. For a more direct comparison with our learned representations, we have projected all low dimensional embeddings and the k-mer representations into 64- and 128-dimensional spaces. The 64-dimensional space corresponds to the ResNet + LSTM encoder-side representations, while the 128-dimensional space corresponds to the ResNet + LSTM decoder-side representations.

As seen in Table 2, NMT and Seq2Vec models performed significantly better than our other learned representations. All our other learned representations performed better than k-mer features. This is consistent with our qualitative evaluation from the tSNE plots — all of our learned representations do a decent job at separating different RRM types.

An important observation from the similarity regression results is that representations learned from aligned and unaligned sequences do not vary significantly in performance. There are a few possible explanations for this finding. One possibility is that either the original HMM alignment of the unlabeled data or the additional step of aligning the labeled and unlabeled data with Clustal Omega program was not effective or accurate, as the alignment does not seem to add to the success of our models. Unfortunately, this step was crucial for similarity regression, as there were no RRMs that appeared in both our labeled and unlabeled datasets. Therefore, without a joint alignment for the labeled and unlabeled data, we could not have sent any of the labeled data through our models, which are learned mostly from unlabeled data, and there would have been no representations to evaluate for our labeled dataset.

Another possible explanation for the lack of performance difference between the aligned and unaligned sequence input is that our models were able to extract local motifs and long-term dependencies even without the alignment, which is meant to ensure that similar structures appear in similar locations within sequences. This explanation is most likely at play for our CNN+LSTM models, as a CNN by design learns location-invariant motifs, and so using aligned input should not be a requirement. At the very least, we can claim that the representations learned from unaligned sequences are not missing significant information versus the aligned sequences. This is an important result, since it would take away the need to align entire RRM or protein sequences prior to learning a viable representation, and would help avoid any alignment errors that may occur when combining two different sequence sets, as we did.

Model	Dimension	Alignment	Training Loss	Validation Loss
Character-Level NMT	128	aligned	3967.20 ± 47.99	4492.35 ± 73.89
Seq2Vec	128	unaligned	4057.98 ± 55.92	4534.29 ± 51.13
Seq2Vec	64	unaligned	4650.91 ± 23.25	4666.35 ± 69.70
ResNet+LSTM Encoder	64	unaligned	4545.28 ± 25.94	4678.62 ± 83.15
Character-Level NMT	64	aligned	4700.97 ± 22.32	4730.99 ± 51.49
ResNet+LSTM Encoder	64	aligned	4814.17 ± 28.86	4732.11 ± 97.45
ResNet+LSTM Decoder	128	unaligned	4709.95 ± 48.19	4886.82 ± 90.95
ResNet+LSTM Decoder	128	aligned	4691.39 ± 100.33	4866.51 ± 66.55
Naive k-mer (ngram)	64	N/A	4233.98 ± 26.09	8307.65 ± 1138.23
Naive k-mer (ngram)	128	N/A	3314.426 ± 25.13	10171.16 ± 1148.82

Table 2: Similarity regression results

9 Conclusions

All three of our models inspired by current NLP methods were able to learn meaningful representations of RRMs, either with unaligned sequences or aligned ones, as demonstrated by the clear clustering of different protein types in tSNE visualizations, as well as by the significantly lower RMSE values for similarity regression when compared with naive k-mer features.

In addition, using unaligned RRM sequences as the input for our models was at least as successful as using sequences that had been aligned by HMM. While a highly useful technique in a large range of biology problems, HMM based sequence alignment may not be a necessary preprocessing step for analyzing protein sequences.

Finally, Seq2Vec has already been a popular example of the successful application of an NLP framework in the biology domain. Our work demonstrates that other neural network NLP models also generalize well to the study of biological sequences, expanding the literature at the intersection of natural language processing and biological sequence processing.

10 Future Work

The next step of this work will be to train models that use entire protein sequences as input, as opposed to our current work, which uses RRM domains as input to the models and then gets protein representations through summing their constituting RRMs. While looking exclusively at the RRM sequences allows us to evaluate some sense of similarity between RRMs, feeding an entire protein sequence into a model instead allows for much greater flexibility when evaluating new examples. It would further remove some of the challenges and sources of error we encountered in matching the RRM domains with their associated gene ID.

11 Acknowledgements

We would like to thank Professor Quaid Morris and Alexander Sasse of the University of Toronto for their invaluable guidance and assistance. Thank you also to Professors Richard Bonneau and Claudio Silva for their suggestions and discussions during the NYU Capstone poster session.

References

[1] Jason Lee, Kyunghyun Cho, and Thomas Hofmann. "Fully character-level neural machine translation without explicit segmentation." arXiv preprint (2016). http://arxiv.org/abs/1610.03017

[2] Quoc Le and Tomas Mikolov. "Distributed representations of sentences and documents." Proceedings of the 31st International Conference on Machine Learning (ICML-14), 1188–1196, 2014

[3] Uniprot Gene Mapping, Accessed 2017-10-30. http://www.uniprot.org/uploadlists/

[4] Human Gene Database, Accessed 2017-10-30. http://www.genecards.org/

[5] Raphael Pelossof, Irtisha Singh, Julie L Yang, Matthew T Weirauch, Timothy R Hughes, and Christina S Leslie. "Affinity regression predicts the recognition code of nucleic acid–binding proteins." Nature Biotechnology Journal (2015). https://www.nature.com/articles/nbt.3343#supplementary-information

[6] ImageCaptioning PyTorch tutorial by Yunjey on GitHub. Accessed: 2017-11-09. https://github.com/yunjey/pytorch-tutorial/tree/master/tutorials/03-advanced/image_captioning

[7] Pytorch Seq2Seq Tutorial, Accessed 2017. http://pytorch.org/tutorials/intermediate/seq2seq_translation_tutorial.html

[8] Xu Min, Wanwen Zeng, Ning Chen, Ting Chen, and Rui Jiang. "Chromatin accessibility prediction via convolutional long short-term memory networks with k-mer embedding." Bioinformatics (2017). http://dx.doi.org/10.1093/bioinformatics/btx234

[9] Robert D. Finn, Penelope Coggill, Ruth Y. Eberhardt, Sean R. Eddy, Jaina Mistry, Alex L. Mitchell, Simon C. Potter, Marco Punta, Matloob Qureshi, Amaia Sangrador-Vegas, Gustavo A. Salazar, John Tate, Alex Bateman. "The Pfam protein families database: towards a more sustainable future." Nucleic acids research 44, pp. D279–D285, 2016

[10] Apweiler R1, Bairoch A, Wu CH, et al. "UniProt: the universal protein knowledgebase." Nucleic acids research 32, pp. D115–D119, 2004 http://www.uniprot.org/help/uploadlists

[11] Dhananjay Kimothi, Akshay Soni, Pravesh Biyani and James M. Hogan. "Distributed representations for biological sequence analysis." arXiv preprint (2016). http://arxiv.org/abs/1608.05949

[12] Debashish Ray, Hilal Kazan, Kate B. Cook. "A compendium of RNA-binding motifs for decoding gene regulation." Nature vol. 499, 2013.

[13] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778, 2016

[14] Hochreiter, Sepp and Schmidhuber, Jürgen. "Long short-term memory." 9, pp.1735–1780, 1997

[15] Kingma, Diederik and Ba, Jimmy. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980, 2014

[16] Maaten, Laurens van der and Hinton, Geoffrey. "Visualizing data using t-SNE." Journal of Machine Learning Research, 9, pp.2579–2605, 2008

[17] Sievers F, Wilm A, Dineen DG, Gibson TJ, Karplus K, Li W, Lopez R, McWilliam H, Remmert M, Söding J, Thompson JD, Higgins D. "Fast, scalable generation of high-quality protein multiple sequence alignments using Clustal Omega." Molecular Systems Biology 7 Article number: 539 doi:10.1038/msb.2011.75

[18] Deng, J. and Dong, W. and Socher, R. and Li, L.-J. and Li, K. and Fei-Fei, L. "ImageNet: A Large-Scale Hierarchical Image Database." CVPR09, 2009 http://www.image-net.org/

Appendix A: Data preprocessing details

While preprocessing the aligned sequences, we eliminated positions that were populated in less than 1% of the combined dataset of labeled and unlabeled sequences. Figure 5 shows the population frequency for each position in the sequence, where population means the presence of an amino acid. As we had far more unlabeled than labeled sequences, using this 1% hard threshold could have resulted in the loss of useful positions in the labeled data, as indicated by the high population frequencies in the labeled data of many of the positions we discarded (plot in the lower left). However, we obtained comparable similarity regression performance when using the aligned sequences, which were preprocessed in this way, and the unaligned sequences, in which no positions were thrown away. Thus, we think this preprocessing step did not cause us to lose predictive information in the labeled aligned sequences.

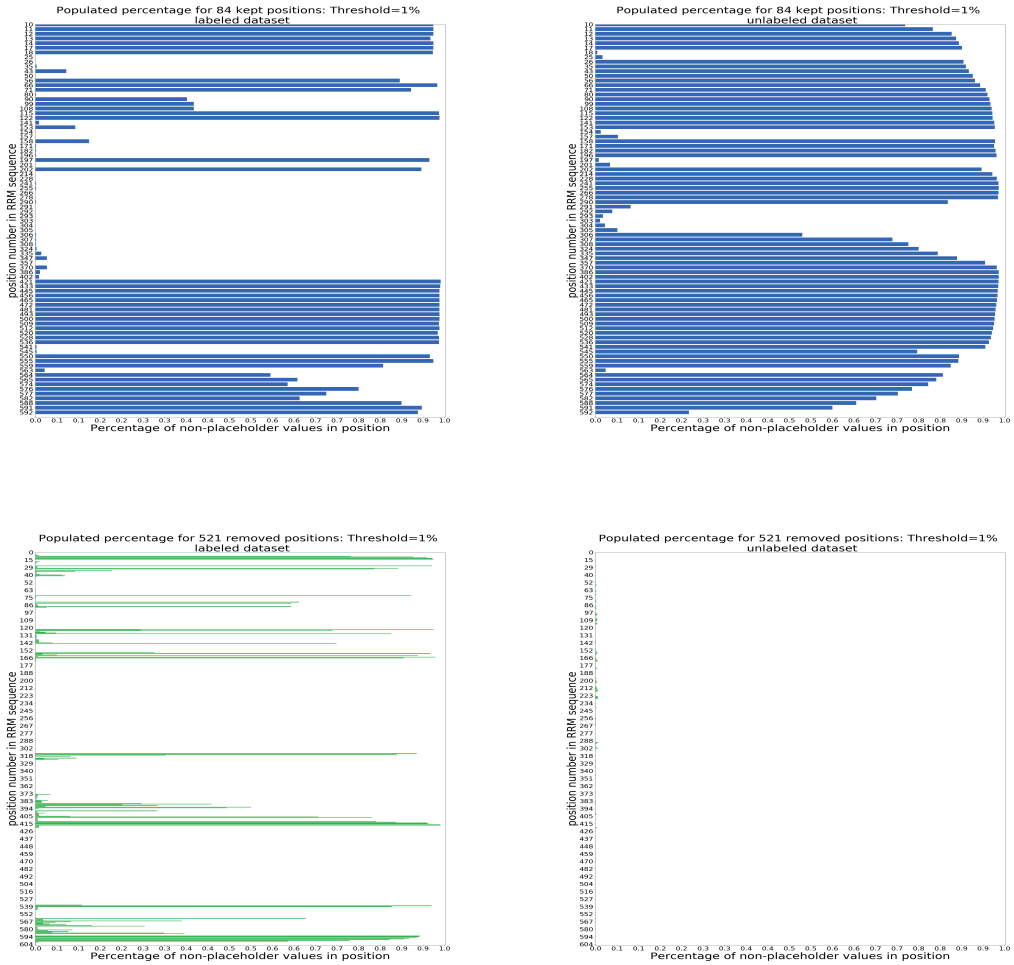

Figure 5: Population frequency by sequence position

Appendix B: Glossary of important terms

Biology:

- Amino acid: One of about 500 organic compounds containing amine (-NH2) and carboxyl (-COOH) functional groups, 20 of which appear in the genetic code.
- Domain: In the context of a protein sequence, a specific substructure that has known biological implications, such as an RRM.
- RNA: Ribonucleic acid, involved in gene expression, regulation, and coding/decoding.
- RRM: A subsequence of an RNA-binding protein that specifies which RNA the protein will bind to.
- Protein: Very large molecule made up of amino acids and involved in innumerable processes within cells in living organisms.
- k-mer: similar to an n-gram in the machine learning nomenclature, k-mer is a group of consecutive amino-acids with length k, within an RRM sequence.

Machine Learning:

- Autoencoder: A model that consists of both an encoder and a decoder and that takes the same sequence as the input and the target.
- Decoder: A model (such as a CNN or RNN) that takes a representation created by an encoder and translates it back into the original sequence or a target sequence.
- Encoder: A model (such as a CNN or RNN) that takes an input sequence and translates it to a lower dimensional representation, to be fed into a decoder.
- CNN: Convolutional Neural Network.
- LSTM: Long Short Term Memory.
- RNN: Recurrent Neural Network.