Predicting post-transcriptional ADAR activity using machine learning

Chang M. Yun (SUNet ID: cmyun) and Xiaowei Zhang (SUNet ID: zhangxw)

1. Introduction

RNA-based therapeutics has become a rapidly emerging field following the successful development of mRNA vaccines. Aside from use in vaccination, RNA-based therapeutics involving RNA editing is also promising, since post-transcriptional RNA editing may be able to correct genetic diseases in a non-invasive manner [1], or act as a switch to control where and when therapeutic effects take place [2]. Naturally in the body, RNA editing is mediated by a family of endogenous human proteins called adenosine-deaminases-acting-o- RNAs (ADARs). ADARs selectively modify the adenosine nucleotide (A) to inosine (I) in certain double-strand RNA (dsRNA), based on sequence and structure [3]. After modification, inosine is recognized as a guanine (G) during translation, allowing A-to-G base editing without modifying genomic DNA. However, despite this critical role, little is known about the exact guiding principles of ADAR activity. While there is some qualitatively understanding of ADAR editing efficiency, this study aims to quantitatively learn, and ultimately, better predict ADAR activity, using two families of machine learning models: (1) ensemble tree-based models, that can capture non-linear relationships, and (2) transformer-based language models, that take advantage of the language-like nature of RNA sequences, to learn underlying relationships between nucleotides (nt). Using these tools, we hope to better design enhanced editing at therapeutic targets, while repressing non-specific editing on off-target sites.

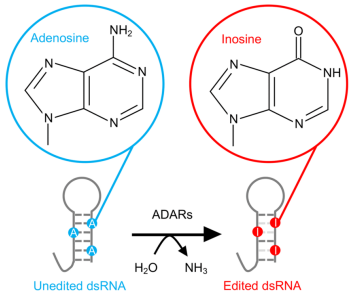


Figure 1. Schematic of adenosine-to-inosine (A-to-I) modification by ADARs [4]

2. Related work

Given the critical role of ADAR in human post-transcriptional control, there has been considerable effort to understand ADAR editing principles to date. However, they are insufficient in several aspects. Early work involved knock-out experiments to identify transcripts that are directly affected by ADAR deletion. However, such approaches are low-throughput and can only detect naturally occurring substrates [5]. Next, using high-throughput sequencing approaches, studies systematically sequenced the transcriptome of human tissues to identify specific sites on RNA preferred by ADAR [6]. However, given RNA's structural versatility, sequencing information alone was unable to capture formation of secondary structures from complementary binding with other RNA.

More recently, researchers have systematically induced structural perturbations on designated dsRNAs. Uzonyi et al. systematically induced structural perturbations on perfect dsRNA substrates to create a library of dsRNA sequences to study ADAR1 editing, although analysis was largely qualitative [3]. Liu et al., applied CRISPR-mediated random mutagenesis on 3 natural substrates and applied XGBoost algorithm to predict editing efficiency with regards to structural features extracted by bpRNA algorithm [7,8]. Although capturing known indicative structural features, including A:C mismatches and thermodynamic parameters on dsRNA, the model does not generalize well to different substrates.

3. Dataset and features

Given the high-quality dataset by Uzonyi et al., yet largely qualitative analysis, we believed that there was room for machine learning to provide additional quantitative understanding. Therefore, we selected the experimental data developed by Uzonyi et al. as our starting dataset. Below is a brief description of the data:

Starting with two known ADAR ds substrates (a mouse endogenous “B2” element, and part of the coding sequence for mNeonGreen “mNG”; 145 nt), a library of 1,822 artificial variants were generated, by introducing different structural perturbations to the lower strand (Figure 2). Eight types of perturbations were introduced: (1) random mismatch, (2) open from end or bulge, (3) extend complementarity, (4) running 1,2,3,4 nt mismatch, (5) T, TTC, TTCTT or TTCTTCT bulge, (6) 1, 2, 3 nt double mismatch, (7) T-to-C or T-to-G, and (8) mismatch base to other 3. This resulted in 1,239 variants of B2, and 583 variants of mNG. The variants were then synthesized, transfected into cultured human cells, endogenously edited, then extracted for RNA sequencing (RNAseq) to identify adenosines edited to inosine. The resultant dataset was provided as aggregated values, describing the percentage of editing at each A on the constant B2/mNG upper strand.

Based on this dataset, we took two approaches to process features and labels for our models. First, for ensemble tree-based models, we chose to model ADAR activity at the adenosine-level (vs. RNA-level), predicting the probability of editing at each adenosine, given features describing its surrounding environment. This allowed us to directly use the percentage editing at each adenosine as output labels. We then performed feature engineering, by selecting key features to describe each ‘A’, including structural perturbations introduced by Uzonyi et al. as well as additional factors known to influence ADAR-editing [3,9]. In total, 19 features were selected to describe each ‘A’ on each dsRNA variant, resulting in ~75,000 unique data points (Table 1).

Second, for transformer-based language models, we chose to model ADAR activity at the RNA-level, taking as input the full sequence of the initial RNA, and predicting as output the modified sequence. For input features, we reverse-engineered the sequence of the dsRNA variants (which were not explicitly provided), by analyzing the RNA synthesis methods, resulting in 1,822 input sequences (396 nt). For output labels, we chose to focus on only the constant upper strand (145 nt), as editing outside of the fixed region was unknown. Lastly, as the original dataset was pre-aggregated (without information about individual permutations), we selected a relative threshold for editing (>30%), and labeled all ‘A’ sites with probability higher than the threshold as edited to ‘I’.

Figure 2. Schematic of full length RNA sequence (396 nt), with a constant upper strand (145 nt), variable lower strand (with perturbations), and a barcode sequence for identification [3].

4. Methods

4.1 Ensemble tree-based models

Based on the features extracted, we expected the features to follow a non-linear relationship, with strong interdependencies between each other. As a consequence, we selected decision tree-based models as the first family of machine learning models, given their no-nlinearity, as well as easy prediction explainability. We applied two ensemble tree models: (1) random Forest Regressor, and (2) gradient boosting decision tree (using XGBoost regressor algorithm). A random forest algorithm employs a bagging strategy, and grows multiple trees independently by bootstrapping data and randomly selecting subsets of features at each node split. A gradient boosting algorithm builds trees sequentially, optimizing a specific objective function by adding trees that minimize the errors of previous ones. By comparing each tree-based model, we were also able to evaluate relative performance for predicting ADAR activity.

4.2 Transformer-based language models

Given the recent advances in language models (e.g., ChatGPT, DALL-E, ESMFold), and the language-like nature of RNA editing (sequence-to-sequence, with underlying relationship between ‘words’ = nucleotides), we selected a transformer-based language model as the second family of machine learning models.

First developed by Vaswani et al. [10], transformers are a type of neural network with several key features:(1) Self-attention: is a mechanism that captures the relative significance between two words (including itself) in a given context, using a query, key, and value to represent each word. A high attention score suggests a strong relationship between two words. The dot product allows for parallel operations, and multi-head (multiple attention mechanisms in parallel) helps capture more relationships. Weight matrices convert an input into a query, key, and

#	Features	Region	Description	Value
1	'distance_to_5'	Local	Distance to 5' end of dsRNA region	Discrete
2	'distance_to_3'	Local	Distance to loop at 3' end	Discrete
3	'global_GC_content'	Global	Total # of GC pair / Total length of RNA	[0,1]
4	'local_GC_content_21'	Local	Total # of GC pair in +/- 10 bp (total 21 bp) length of RNA	[0,1]
5	'bulge-T'	Global	Presence of T-bulge on RNA	Boolean
6	'bulge-TTC'	Global	Presence of TTC-bulge on RNA	Boolean
7	'bulge-TTCTT'	Global	Presence of TTCTT-bulge on RNA	Boolean
8	'bulge-TTCTTCT'	Global	Presence of TTCTTCT-bulge on RNA	Boolean
9	'bulgesite' [1]	Local	1 / Distance to bulge site	[0,1]
10	'mismatch_num'	Local	# of mismatch sites	{0,1,2}
11	'mismatch_site_num'	Local	Number of mismatches in each mismatched site	{0,1,2,3,4}
12	'mismatch_site_1'[1]	Local	1 / Distance to the first mismatch site (closer to 5' end)	[0,1]
13	'G_penalty_1'[2]	Local	Gibbs free energy penalty in the first mismatch	Continuous
14	'mismatch_site_2'[1]	Local	1 / Distance to second nearest mismatch (closer to 3' end)	[0,1]
15	'G_penalty_2'[2]	Local	Gibbs free energy penalty in the second mismatch	Continuous
16	'editing_site_mismatch'	Local	If self-mismatched	Boolean
17	'is_toC'	Local	If self-mismatch is A-C mismatch	Boolean
18	'is_toG'	Local	If self-mismatch is A-G mismatch	Boolean
19	'is_out_of_ds'	Local	If this adenosine is not in double stranded region	Boolean

Table 1. 19 engineered features selected to describe an adenosine (A) site.

value (learned during training). The output of attention is then fed into a standard feed-forward neural network. (2) Encoder-decoder: is an architecture that can handle inputs and outputs of varying length. An encoder takes an input, and encodes it into a certain state (here, through layers of attention and neural networks). The decoder then takes the encoded state and current output context, and decodes it (through additional layers of attention and neural networks) to predict the next word. (3) Masking: hides the appropriate training labels for self-supervised learning. In an autoregressive model (left-to-right), masking hides all attention with words to its right, mimicking left-to-right prediction. (4) Other features: Positional embedding inject positional information of each word by adding a sinusoidal wave, allowing positional information to be captured without additional tokens or dimensions. Addition & normalization: connects the input and output of attention, for more efficient weight matrix learning during training. Multi-layer: adds multiple layers of attention and neural networks, in series.

The first transformer model was used for language translation. Thinking of ADAR editing as a translation problem (original 'language': RNA sequence with A's; translated 'language': RNA sequence with some A's converted to I's), we decided to adapt this model (also provided by TensorFlow as a tutorial [11]).

During adaptation, the following features were newly added to the model: (1) RNA tokenizer: an 8-word vocabulary to tokenize each nucleotide (A, T, C, G, I, 5' *start*, 3' *end*, *empty*). (2) Teacher forcing: As the training outputs all have the same constant upper arm pattern, the model was prone to instability if the output veered away from the typical sequence. Therefore, for any non-A or I predictions, the model was forced to output the fixed upper arm sequence (T, C, G). (3) A vs. I: As ADAR activity is ultimately interested in whether an A is edited to I or not, for the final prediction layer, instead of selecting the nucleotide with the highest probability, we compared the probability between A vs. I, and selected the nucleotide with higher probability.

5. Results and discussion

5.1 Ensemble tree-based models

Training: To train the model, we split the dataset into training (70%), and test (30%), and used 5-fold cross validation on the training set to explore the best hyperparameter space for the model. RMSE was used as the

[1] Features were calculated as one over distance in order to take into account when the perturbation is not present. When not present, perturbation is infinitely far away, equaling zero.
[2] Gibbs free energy penalty when introducing a specific mismatch is calculated based on Vendeix and Agris, *RNA*, 2009 [9].

evaluation metric. In order to help increase model generalizability and keep the model simple, we chose the simplest model that could achieve $R^2 > 0.9$ through cross validation.

Performance and indicative features: For training on random forest regressor, we set the number of estimators to be fixed at 50, to reduce the computational cost, while maintaining complexity. We used GridSearchCV to explore the best hyperparameter combination on min_samples_leaf, min_samples_split and max_depth. The final parameters chosen were: min_samples_leaf =5, min_samples_split=5 and max_depth=15. The training and test accuracies (as shown by R^2) are reported in Table 2. To analyze the most indicative features, instead of using the built-in impurity-based feature importance in the SciKit-Learn package, which biases towards high cardinality features, we evaluated by permutation feature importance, which is defined as the decrease in a model score when a single feature value is randomly shuffled. The top 5 most indicative features identified in the model are listed in Table 2 (in descending order). Similar training and analysis pipelines were also applied in XGBoost regressor. By cross-validation, we chose the following parameters: learning_rate=0.02, n_estimators=800, max_depth=6. Training, testing accuracies, and top 5 most indicative features are listed in Table 2.

Model	Training MSE	Training accuracy	Test MSE	Test accuracy	Best indicative features
RandomForest Regressor	9.0310	0.9449	17.2526	0.8934	dist_to_3; dist_to_5; global_GC; local_GC_21; mismatch_site_1
XGBoost Regressor	12.80	0.9211	115.42	0.3009	dist_to_3; dist_to_5; global_GC; local_GC_21; bulgesite

Table 2. Model predictions and indicative features by different tree-based algorithms

From the result, random forest regressor outperformed XGBoost regressor in predicting an unseen test set, and XGBoost was observed to undergo severe overfitting. Nevertheless, both models gave similar results for the most indicative features: positional information (distances) and thermodynamic features (GC contents), which were also confirmed by what Liu et al. had observed [7].

Generalizability: Next, since random forest regressor worked well with test set adenosines that came from the same sequence context in the training set, we were further interested in testing if the model could generalize to different sequence contexts. This would be able to tell us if structural information alone could predict ADAR editing, without sequence-level information. We used the same model parameters to train a random forest regressor with only data from the B2 variants, and compared the model's performance between unseen data points from the same B2 origin, and data points from a sequentially different mNG origin. From the result, the model trained only on B2 sequence context performed well for predictions coming from the same sequence context, but had no predictive power for prediction in a different "mNG" sequence context (Table 3) This indicated that structural information alone is not sufficient in predicting ADAR activity, and that sequence-level information is critical, again, consistent with observations by Liu et al. [7].

Dataset	MSE	Accuracy
B2_training	5.284	0.9665
B2_test	7.867	0.9497
mNG	240.7	-0.3896

Table 3. Model predictions on model trained only with adenosines on B2 sequence context

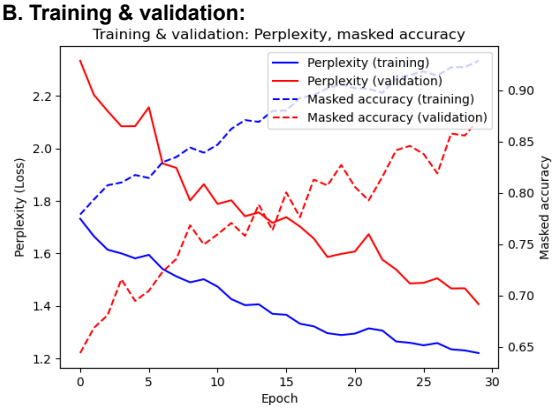
5.2 Transformer-based language models

Model training and validation: For the transformer model, the dataset for B2 and mNG was combined, then divided into 80% training, 10% validation, and 10% testing. During training and validation, the following parameters were selected, based on context length (396 nt), training time (~6-12 hours), and bias-variance analysis: model_dimensions=512, neural_network_dimensions=512, attention_heads=4, layers=2, epochs=30. Two metrics were used to analyze bias-variance: perplexity and masked accuracy. The model used categorical

cross entropy loss (cross entropy with softmax). As a linear proxy, perplexity, which is the exponential of the loss (= exponential of logarithm = linear; when loss = 0, perplexity = 1) was calculated. Over training epochs, perplexity of both training and validation decreased towards 1, reaching ~1.2 (suggesting low bias), and maintained a negative gradient (suggesting low variance, but with room still left to train) (Figure 3-B). Masked accuracy calculates the accuracy in predicting the masked labels, and was used to secondarily confirm the analysis.

Model testing: As the model was forcing all non-A/I predictions (Figure 3-A), the model was evaluated on A vs. I predictions. Four metrics were used to evaluate test performance: accuracy, precision, positive and negative recall. The model's mean test accuracy (= (true positive+true negative) / total predicted) across 139 test sequences was 96.44%, suggesting high predictive performance. We had some concern about the low levels of editing in the dataset (~10%), however, precision (= true pos / predicted pos) was 95.77%, and positive recall (= true pos / labeled pos) was 84.91%, suggesting positive prediction performance was still high. Finally, negative recall (= true neg / labeled neg) was 99.08%, confirming the model's expected strong negative prediction ability (Figure 3-B). The results showed that a transformed-based language model can be a highly effective way to predict the RNA editing activity of ADAR.

B. Training & validation:



A. Sample prediction *(A,I in bold)*:

Input (396 nt):
5ACCTTGGGTCATACGTTAACCGGACAGATGCGCCG…
…TCTTCAGATAGTTAGCCGCCATTGGCTGGCGCGCC3

Predicted (145 nt):
5**A**GCC**AA**TGGCGGCT**AA**CTATCTGA**AAA**CC**I**CCG**A**T…
…**I**TTT**A**CCG**A**TGTG**A**TGGGC**A**T**IA**CG**A**GCTGT**AC**I**IAA**3

True output (145 nt):
5**A**GCC**AA**TGGCGGCT**AA**CT**A**TCTG**AIAA**CC**I**CCG**A**T…
…**I**TTT**A**CCG**A**TGTG**A**TGGGC**A**T**IA**CG**A**GCTGT**AC**I**IAA**3

C. Test performance *(Test size: 139 data points)*:

Accuracy	Precision	Positive recall	Negative recall
0.964	0.958	0.849	0.991

Figure 3. (A) Sample input, predicted output, and true output. (B) Training and validation: Perplexity and masked accuracy, over 30 training epochs. (C) Test performance: Accuracy, precision, positive recall, and negative recall

6. Future work and conclusion

This study examined the use of two families of models to predict RNA editing activity of ADAR proteins: decision tree-based models, and transformer-based language models. In decision tree models, a random forest model performed well for predictions similar to the training set, however, were limited by generalizability of the engineered features describing each 'A' site. A universal feature set, that can describe any RNA sequence, would be required to explore the full potential of such models. In transformer-based language models, similarly, the model performed well during testing, however, the model experienced a similar data-limited, as the dataset was highly homogenous, originating from just two starting sequences (B2, mNG) that covered a small permutation space. As the test data also originated from the same homogenous dataset, performance metrics were likely inflated, as predictions were highly localized, making relatively 'easy' predictions. We were unable to show whether the model's training and prediction can be generalized to a broader set of RNA sequences, although we expect better generalizability than tree-based models as RNA sequences can be directly used as input . Additionally, the transformer used autoregressive masking, however, as RNA interaction is bidirectional, a random masking transformer model (e.g., BERT, BART) is likely more suitable. Lastly, a probability threshold during data processing likely lost a significant amount of information. Using an un-preprocessed RNAseq data (with individual sequences) or one-hot encoding (to capture percentages between 0 and 1) would preserve this information.

Overall, we believe transformer-based language models are an appropriate method to predict ADAR activity, and will be a critical tool towards the development of ADAR-based RNA therapeutics in the near future.

7. Contributions

Chang M. Yun: Contributed to the development of the transformer-based language model, including: reverse-engineering RNA sequence, data processing for transformer, 8-word vocabulary tokenizer, transformer-based model, parameter selection, model training and validation, and performance testing.

Xiaowei Zhang: Contributed to feature engineering portion of the original dataset and developing decision tree-based models. Both of the contributors worked on preprocessing and analyzing the original dataset.

Github link to code: https://github.com/Xiaowei0402/CS229_final_project.git

8. References

1. Booth, B.J., Nourreddine, S., Katrekar, D., Savva, Y., Bose, D., Long, T.J., Huss, D.J., Mali, P., RNA editing: Expanding the potential of RNA therapeutics., *Molecular Therapy*, 31 (6) (2023), 1533–1549.
2. Kaseniit, K.E., Katz, N., Kolber, N.S., Call, C.C., Wengier, D.L., Cody, W.B., Sattely, E.S., Gao, X.J., Modular, programmable RNA sensing using ADAR editing in living cells., *Nature Biotechnology*, 41 (4) (2023), 482–487.
3. Uzonyi, A., Nir, R., Shliefer, O., Stern-Ginossar, N., Antebi, Y., Stelzer, Y., Levanon, E.Y., Schwartz, S., Deciphering the principles of the RNA editing code via large-scale systematic probing., *Molecular Cell*, 81 (11) (2021).
4. Nakahama, T. & Kawahara, Y. Adenosine-to-inosine RNA editing in the immune system: friend or foe? *Cell. Mol. Life Sci*. 77, 2931–2948 (2020).
5. D. P. Morse, Identification of substrates for adenosine deaminases that act on RNA. *Methods Mol. Biol.* **265**, 199–218 (2004).
6. O. Gabay, Y. Shoshan, E. Kopel, U. Ben-Zvi, T. D. Mann, N. Bressler, R. Cohen-Fultheim, A. A. Schaffer, S. H. Roth, Z. Tzur, E. Y. Levanon, E. Eisenberg, Landscape of adenosine-to-inosine RNA recoding across human tissues. *Nat. Commun.* **13**, 1184 (2022).
7. X. Liu, T. Sun, A. Shcherbina, Q. Li, I. Jarmoskaite, K. Kappel, G. Ramaswami, R. Das, A. Kundaje, J. B. Li, Learning cis-regulatory principles of ADAR-based RNA editing from CRISPR-mediated mutagenesis. *Nat. Commun.* **12**, 2165 (2021).
8. P. Danaee, M. Rouches, M. Wiley, D. Deng, L. Huang, D. Hendrix, bpRNA: large-scale automated annotation and analysis of RNA secondary structure. *Nucleic Acids Res.* **46**, 5381–5394 (2018).
9. F. A. P. Vendeix, A. M. Munoz, P. F. Agris, Free energy calculation of modified base-pair formation in explicit solvent: A predictive model. *RNA* **15**, 2278–2287 (2009).
10. Vaswani, A. *et al.* Attention is All you Need. in *Advances in Neural Information Processing Systems* vol. 30 (Curran Associates, Inc., 2017).
11. Neural machine translation with a Transformer and Keras | Text | TensorFlow. https://www.tensorflow.org/text/tutorials/transformer.