Explaining Recurrent Neural Networks by Leveraging Interactive Visualizations

Ihor Markevych and **Clay Yoo**
Carnegie Mellon University
imarkevy, hyungony@andrew.cmu.edu

1 Introduction

While the performance of Neural Networks is becoming better and better nowadays, it comes at a cost. Researchers introduce more and more complex architectures which leads to almost uninterpretable models. Model interpretability is a process of explaining why some inputs result in a specific output and is critical for most domains where models are involved in decision-making, especially when their decisions might have an impact on humans, such as banking, application auto-processing, etc. By being able to interpret the model researcher can investigate for potential biases and mistakes in priors. While there is a lot of work on visualizing and interpreting Convolutional Neural Networks (CNN) and Dense Neural Networks, there is little to no work on increasing the interpretability of Recurrent Neural Networks. In this work, we introduce a general framework that is capable of explaining which sequence elements had an impact on the final output and how the feature space was transformed while being processed by network layers.

2 Related Work

There have been active works on visualizing Convolutional Neural Networks, and Wang et al. (2020) proposed an excellent interactive visualization tool to show the contributions of components that constitute CNNs. While the tool does not provide the flexibility of letting users choose which models to inspect, it captures how each node at each step contributes to the weight and activation in the granularity of pixels in the images.

Not as much as CNNs, however, for RNNs, Karpathy et al. (2015) showed saturation plots, by defining left/right-saturation according to activation value, to visualize how LSTM weights change during the training. They also used text highlighting to reveal the existence of interpretable cells in tracking long-range dependencies between word tokens. Figure 6 is directly inspired by this work.

Strobelt et al. (2018) proposed an interactive tool that allows the user to identify and explore patterns in RNNs trained on a large dataset. While the LSTMVis provides users ways to visualize hidden states at different timestamps, it requires the users to play around using pre-trained models with specifications required by the program.

We attempt to tackle the weaknesses of both papers related to RNNs, by providing users with an interactive tool to understand how general RNNs work, not necessarily limited to particular types of networks, and gain deeper insight on what each step is doing on the model.

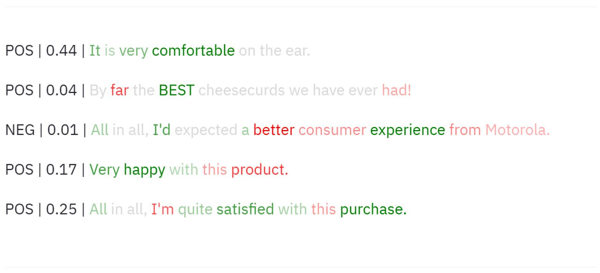
3 Methods

We illustrate the proposed framework with a specific and considerably simple model. We build a Recurrent Neural Network that contains Embeddings layer, LSTM layer with a latent dimension of 64, Dropout layer, Dense layer with 64 neurons and ReLu activation, another Dropout layer and output Dense layer with single neuron and sigmoid activation function. We train this model over binary sentiment classification problem on IMDB reviews (Maas et al., 2011). It provides a set of 25,000 highly polar movie reviews for training, and 25,000 for testing.

Model is trained over 10 epochs with binary cross-entropy loss and achieves 80% accuracy on the test set. To achieve interpretability we aim at giving a user an option to explore interactively, hoping that a broader picture can be extrapolated by observing partial examples.

Specifically, we introduce two types of visualizations, one for visually describing word impact in a sentence and another for showing transformations that are performed inside the model layers. We allow user to input their own sentence to allow more flexible interactive exploration or sample reviews from our prepared dataset to make it easier for user to explore.

To build the first visualization, we repeatedly evaluate sub-sequences of the sentence (e.g. “I” - “I like” - “I like Machine” - “I like Machine Learning”) by obtaining predictions from our model and measuring the relative impact of adding a word to the sequence. We use color encoding to denote positive (green) and negative (red) impacts. We additionally use grey color encoding to denote neutral or close to neutral impact (with an empirical threshold of (0.1 * max impact for that sequence) to decide whether to consider impact as being neutral). We use an opacity encoding channel to denote the strength of the impact, with higher opacity denoting higher impact. This effectively allows the user to observe patterns in how the model interprets specific sentences.


POS | 0.44 | It is very comfortable on the ear.

POS | 0.04 | By far the BEST cheesecurds we have ever had!

NEG | 0.01 | All in all, I'd expected a better consumer experience from Motorola.

POS | 0.17 | Very happy with this product.

POS | 0.25 | All in all, I'm quite satisfied with this purchase.

Figure 1: Proposed visualization of word impacts. First column denotes label (positive / negative), second - probability or confidence in predicted label and third is words impact visualization.

To visualize the progressive transformations of the feature space that are performed inside each layer we select a fixed subset of 500 samples from the unseen dataset and visualize the output of the next layer: Embeddings, LSTM, hidden Dense. Because those outputs are of high dimensions we perform ISOMAP (Balasubramanian and Schwartz, 2002) to obtain a two-dimensional representation for visualization. While we wanted to use more well-studied embeddings, such as t-SNE (Hinton and Roweis, 2002) and UMAP (Sainburg et al., 2020), we discovered that their processing time is considerably long and defeat the interactivity by making the application too slow and unresponsive.

User is also able to obtain neighboring (in terms of those feature spaces) reviews by hovering their cursor over the point. We also allowed to select a subset of points with a tooltip to observe how a specific subset was transformed:

We additionally visualize the training process by providing described visualizations for check-

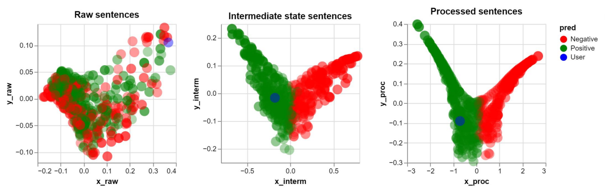


Figure 2: Proposed visualization of progressive transformations of the feature space . Left chart shows word embeddings, middle - output of LSTM layer, right - output of hidden Dense layer. Color denotes predicted label. Additionally we visualize user input with blue point.

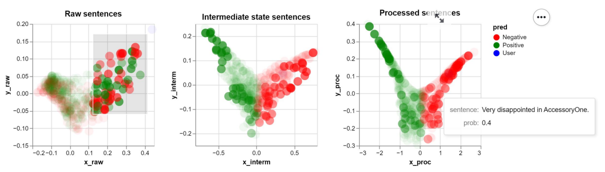


Figure 3: Created chart interactions. Note that we can see some dependency between how dense was feature subspace before and after transformations.

pointed model states after each two epoch for word impact visualization and every five epochs for feature space transformations to allow users to observe the learning process itself and potentially obtain additional insights into the model functionality.

4 Results

We have created an interactive web application as an illustration of the proposed interpretability framework. By using its functionality, the user can interpret both models at inference time and the learning process.

An example of insight that can be obtained from word impact visualization would be as follows. Consider the sentence “This movie is boring”:

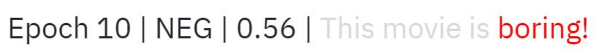

Epoch 10 | NEG | 0.56 | This movie is boring!

Figure 4: Visualization of words impact of specific sentence.

We can see that model ignores all words except “boring”. To confirm this we can type a close sentence “That film was boring” - we see that model predicts same probability of example being negative review.

However, this can also raise a concern - what about the sentences like “This movie is not boring”? We can see that the model again ignores all

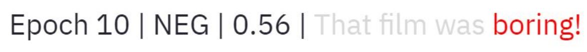

Epoch 10 | NEG | 0.56 | That film was boring!

Figure 5: Visualization of words impact of specific sentence. Note that probability is exactly same as for previous example.

words except for “boring” and outputs the same probability. We can interpret this in a way that model simply remembered this word as a negative indicator and is not properly learning sequential dependencies.

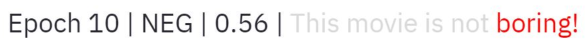
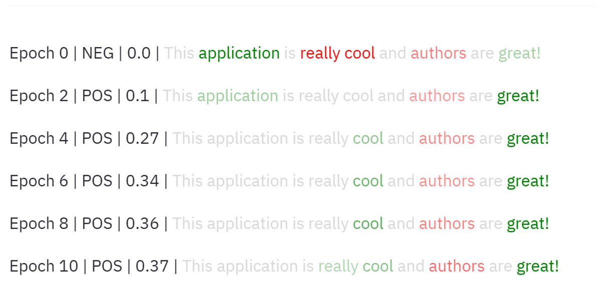

Epoch 10 | NEG | 0.56 | This movie is not boring!

Figure 6: Visualization of words impact of specific sentence. Note that while it has word "not", predicted probability is still the same.

Another example can be "This application is really cool and authors are great!". We can see that a supposedly neutral word “authors” is considered to have a slightly negative impact. It is possible that either model got this bias from the distribution of the train data, or that the word “authors” is unknown to the model and by default the model assigns a default token (e.g. `UNK` token) with a negative probability.


Epoch 0 | NEG | 0.0 | This application is really cool and authors are great!

Epoch 2 | POS | 0.1 | This application is really cool and authors are great!

Epoch 4 | POS | 0.27 | This application is really cool and authors are great!

Epoch 6 | POS | 0.34 | This application is really cool and authors are great!

Epoch 8 | POS | 0.36 | This application is really cool and authors are great!

Epoch 10 | POS | 0.37 | This application is really cool and authors are great!

Figure 7: Visualization of training process using feature space charts and model’s state at specific points during training.

By observing the progressive transformations of the feature space, we can see that while sentence embeddings are considerably inseparable, LSTM layer transforms them into linearly separable structure, where the further the point is from decision boundary, the higher probability model assigns. We also see that hidden layer almost does not change the feature space, which raises a hypothesis that this layer does not have a significant impact on model performance, and the model without it can achieve a comparable performance.

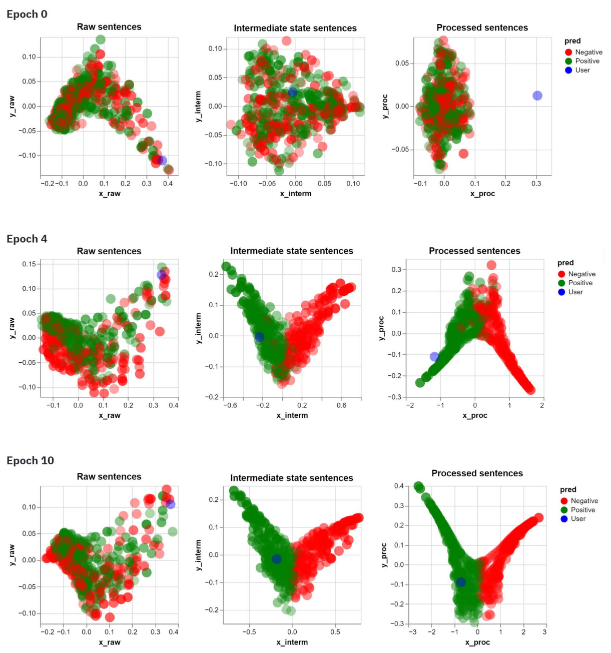


Figure 8: Visualization of training process using feature space charts and model’s state at specific points during training.

It is possible to interpret separate examples and observe feature space transformations for a subset of data. This enables both extrapolating over how the model works in general and interpreting some specific important examples by feeding user input into the application.

5 Discussion

The proposed framework for visualizing Recurrent Neural Networks is general and can trivially be extended to more complex models. In the multi-class classification, for instance, it is possible to encode a specific class with color, while still using opacity to encode the impact of a specific element. It is also possible to perform the same visualization for non-NLP problems. The visualization of progressive transformations of the feature space can be extended to models with a higher number of layers by adding additional charts (for instance, since our model had three hidden layers, we had three charts). Using such a framework it is possible to obtain some insights into how RNNs work and why specific examples were processed in one or another way. It should also be noted that this framework could be applied to any dataset or model as long as the user provides both dataset and model to the application.

6 Future Work

It is possible to improve the proposed framework by additionally visualizing hidden states of LSTM/GRU units and trying to find patterns there. Additionally we can evaluate the model on n-grams instead of separate words and create a visualization of context that had an impact on specific elements, for instance, neighboring words.

Attention mechanism (Vaswani et al., 2017) is a widely-studied and commonly-applicable method that usually yields significant performance increase in context-dependent problems. While it's more interpretable, we also consider adding it to our application since by having it together with other visualizations user will have broader perspective on different aspects of the model.

References

Mukund Balasubramanian and Eric L. Schwartz. 2002. The isomap algorithm and topological stability. *Science*, 295(5552):7–7.

Geoffrey E Hinton and Sam Roweis. 2002. Stochastic neighbor embedding. *Advances in neural information processing systems*, 15:857–864.

Andrej Karpathy, Justin Johnson, and Li Fei-Fei. 2015. Visualizing and understanding recurrent networks.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Tim Sainburg, Leland McInnes, and Timothy Q. Gentner. 2020. Parametric umap: learning embeddings with deep neural networks for representation and semi-supervised learning. *ArXiv e-prints*.

H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. 2018. Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):667–676.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Zijie J. Wang, Robert Turko, Omar Shaikh, Haekyu Park, Nilaksh Das, Fred Hohman, Minsuk Kahng, and Duen Horng Chau. 2020. CNN Explainer: Learning Convolutional Neural Networks with Interactive Visualization. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*.