Visualizing RNNs with Automatic Rule Extraction

ZEWEI CHU,

ACM Reference format:
Zewei Chu. 2017. Visualizing RNNs with Automatic Rule Extraction. 1, 1, Article 1 (June 2017), 4 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 ABSTRACT

We apply the automatic rule extraction method as proposed in [1] to analyze the contribution of each word to the final prediction on sentiment classification tasks, and generate heat maps to rationalize LSTM model predictions.

2 INTRODUCTION

Recurrent neural networks such as Long short-term memory and gated recurrent unit works as a black box in many experiments. In this project, we study the contribution of each sequence token to the final prediction probabilities in classification tasks using the method proposed in [1].

We use the methods proposed in [1]. More specifically, when we use LSTM for classification tasks, we typically use the final hidden state multiplied by a matrix, and do a SoftMax on the output to get a distribution. [1] gives a simple method to analyze the contribution of each input token to the final probability distribution.

We visualize the predictions by heat maps of varying sizes, where words of larger fonts sizes make more contribution to the final predictions.

3 RELATED WORK AND METHODS

3.1 LSTM

Long short-term memory is defined as follows.

$$f_t = \sigma(W_f x_t + V_f h_{t-1} + b_f) \tag{1}$$

$$i_t = \sigma(W_i x_t + V_i h_{t-1} + b_i) \tag{2}$$

$$o_t = \sigma(W_o x_t + V_o h_{t-1} + b_o) \tag{3}$$

$$\tilde{c_t} = \tanh(W_c x_t + V_c h_{t-1} + b_c) \tag{4}$$

$$c_t = f_t c_{t-1} + i_t \tilde{c_t} \tag{5}$$

$$h_t = o_t \odot \tanh(c_t) \tag{6}$$

Manuscript submitted to ACM

Long short-term memory has cell states c_t and hidden states h_t. We define $c_0 = h_0 = 0$ as initial values. When applied to classification tasks, the output of the LSTM h_t is typically followed by an output transformation matrix W and a SoftMax layer to be converted to a probability distribution over the output categories.

$$p_i = SoftMax(Wh_T) = \frac{e^{W_i h_T}}{\sum_{j=1}^{C} e^{W_j h_t}} \tag{7}$$

3.2 Decomposing the output of LSTM (β decomposition)

We can decompose the numerator of p_i into a product of factors[1], and interpret the factors as the contribution of individual word to the final predicted probability.

$$\beta_{i,j} = \exp(W_i(o_T \odot (\tanh(c_j) - \tanh(c_{j-1})))) \tag{8}$$

so that

$$\begin{aligned}\exp(W_i h_T) &= \exp(\sum_{j=1}^{T} W_i(o_T \odot (\tanh(c_j) - \tanh(c_{j-1})))) \\ &= \prod_{j=1}^{T} \beta_{i,j}\end{aligned}$$

We can treat $\beta_{i,j}$ as the contribution of token j to the predicted probability of class i.

3.3 Additive decomposition (γ decomposition)

The previous method of decomposition does not account for the information affected by forget gates. The following approach as introduced in [1] better captures the contribution each word with forget gates.

$$c_T = \sum_{i=1}^{T}(\prod_{j=i+1}^{T} f_j) i_i \tilde{c_i} = \sum_{i=1}^{T} e_{i,T} \tag{9}$$

$$\exp(W_i h_T) = \prod_{j=1}^{T} \exp(W_i(o_T \odot (\tanh(\sum_{k=1}^{j} e_{k,T}) - \tanh(\sum_{k=1}^{j-1} e_{k,T})))) \tag{10}$$

$$= \prod_{j=1}^{T} \exp(W_i(o_T \odot (\tanh((\prod_{k=j+1}^{T} f_k)c_j) - \tanh((\prod_{k=j}^{T} f_k)c_{j-1}))))) \tag{11}$$

$$= \prod_{j=1}^{T} \gamma_{i,j} \tag{12}$$

$\gamma_{i,j}$ can be viewed as the contribution of word j to the final predicted probability to class i.

4 EXPERIMENT

We experiment on Stanford sentiment treebank movie review dataset from Assignment 1 and Yelp dataset challenge. The Stanford sentiment treebank movie review dataset can be found at http://ttic.uchicago.edu/~kgimpel/teaching/31210/assignments.html.

As for the Yelp dataset challenge, it can be downloaded from https://www.yelp.com/dataset_challenge. We use the data from file yelp_academic_dataset_review.json. It contains a review to a merchant from a user, as well as the star rating given by the user. We treat a review of star 1-2 as negative and 3-5 as positive. As this dataset is too huge for training, we only randomly picked 50000 instances for training (TRAIN), 6000 for development set (DEV) and 3800 for testing (TEST). The yelp polarity dataset we constructed can be downloaded at http://people.cs.uchicago.edu/~zeweichu/lstm_rule_extract/yelp_polarity_data.tar.gz.

We use an LSTM model to train on both datasets, and use the previously defined rule extraction approach to study the contribution of each word to the final predictions. We write the code in PyTorch. We generate heat maps on TEST. Bigger font size means bigger contribution to the final prediction, i.e., if an instance is classified as positive, large font size means positive word and small font size means negative word, while if an instance is classified as negative, larger font size means more negative and smaller font size means positive. Word with RED color always represents the most positive word and word with GREEN color always represents the most negative word.

We use Adam as the optimizer, and we set the initial learning rate to be 0.001. We train for at most 10 epochs. The model with highest accuracy on DEV is saved for evaluation and heat map generation. We did not do any parameter tuning as the goal of the project is not to improve accuracy on sentiment classification.

The code can be found at https://github.com/ZeweiChu/rule_extract_lstm, and the heat maps of 1000 instances of each category (HW1 β, HW1 γ, Yelp β, Yelp γ) are hosted at http://people.cs.uchicago.edu/~zeweichu/lstm_rule_extract/.

The accuracy of our model is in Table 1 and the heat maps generated are shown at Fig 1, Fig 2, Fig 3, Fig 4, Fig 5, Fig 6, Fig 7, and Fig 8.

	Assignment 1	**Yelp Polarity**
accuracy	82.87%	87.43%

Table 1. Accuracy of LSTM on sentiment classification

Fig. 1. β Reviews classified as positive in Assignment 1

Fig. 2. β Reviews classified as negative in Assignment 1

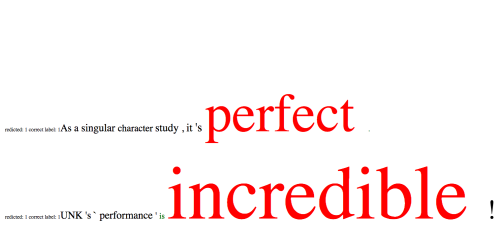

Fig. 3. γ Reviews classified as positive in Assignment 1

Fig. 4. γ Reviews classified as negative in Assignment 1

Fig. 5. β Reviews classified as positive in Yelp polarity dataset

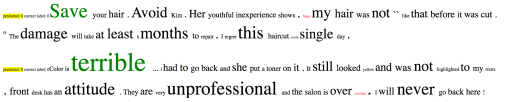

Fig. 6. β Reviews classified as negative in Yelp polarity dataset

Fig. 7. γ Reviews classified as positive in Yelp polarity dataset

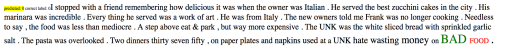

Fig. 8. γ Reviews classified as negative in Yelp polarity dataset

4.1 Observations

(1) The results of both β and γ decomposition look reasonable, i.e., words of larger font sizes tend to make more contribution to the final predictions.

(2) Our results show that β decomposition has more false positive, the same as what [1] discovered, while γ decomposition gives more accurate results.

(3) When applying this method on longer text, the contribution of each word to the final prediction tend to be closer while for shorter text, the contribution varies more significantly.

5 CONCLUSION AND FUTURE WORK

Our work shows that the method of automatic rule extraction as proposed in [1] can generate good heat maps to visualize the results of LSTM on classification tasks. This method can both explain what LSTM learns and visualize the underneath prediction logic.

REFERENCES

[1] W. James Murdoch and Arthur Szlam. 2017. Automatic Rule Extraction from Long Short Term Memory Networks. *arXiv preprint* (2017).