Text Classification and Sentence Generator

Dingting Huang, Yize Wang

Northeastern University
{huang.din,wang.yize1}@husky.neu.edu

Abstract. As our title, our project is divided into two parts. The first part for our project is to compare recurrent neural network and other algorithms to classify the comments whether it is a good comment or not to validate whether RNN is suitable for text area. The second part is to use recurrent neural network make a sentence generator. It will generate relevant comments based on the values entered. This part is tweaking RNN and applying it in a novel way. The algorithms we used in the project are RNN(recurrent neural network), logistic regression, SVM(support vector machine), Random Forest Classifier, XGBoost Classifier, LSTM(Long Short Term Memory), GRU(Gated Recurrent Unit), and word embedding. Through the comparison, we find the RNN has the highest accuracy in text classification. It has about 84.3% accuracy. So RNN does a better job in text classification than other algorithms. In the sentence generator part, the best model structure is one GRU layer and its parameter is 512 and its accuracy is 18.99%. When we use low creativity, the generated word will be the highest appear frequency in the model. However, when we use the higher creativity, the generated word will be diverse.

Keywords: RNN(Recurrent Neural Networks), LSTM(Long short-term memory), GRU(Gate Recurrent Unit), Word embedding

1 Introduction

Recurrent Neural Networks are the state of the art algorithm for sequential data and among others used by Apples Siri and Googles Voice Search. This is because it is the first algorithm that remembers its input, due to an internal memory, which makes it perfectly suited for Machine Learning problems that involve sequential data. It is one of the algorithms behind the scenes of the amazing achievements of Deep Learning in the past few years.

2 Database

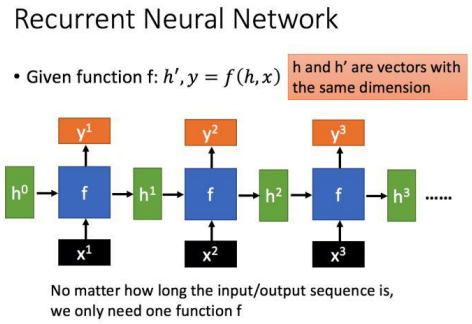
This dataset consists of reviews of fine foods from amazon. The data span a period of more than 10 years, including all 500,000 reviews up to October 2012. Reviews

include product and user information, ratings, and a plain text review. It also includes reviews from all other Amazon categories. The following is the link for the database: https://www.kaggle.com/snap/amazon-fine-food-reviews . What we use to classify is whether the comments are 5 star or not. In addition, what we use to sentence generator are the text of the database.

3 Methods

3.1 Recurrent Neural Networks：

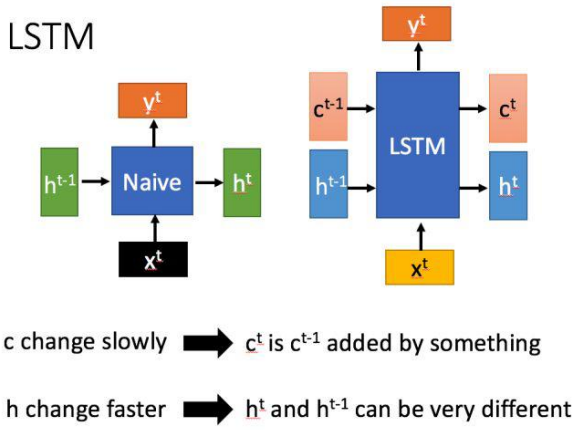
The Recurrent Neural Network (RNN) is a neural network for processing sequence data. He is able to process data from sequence changes compared to a typical neural network. For example, the meaning of a word will have different meanings because of the different content mentioned above, and RNN can solve such problems well.
Let's briefly introduce the general RNN.
x is the input of the data in the current state, and h is the input of the last node received.
y is the output in the current node state, and h' is the output passed to the next node.
As you can see from the formula in the figure above, the output h' is related to the values of x and h.
However, y often uses h' to invest in a linear layer (mainly for dimensional mapping) and then uses softmax to classify the required data.
How to use y to calculate how to use the specific model is often used.



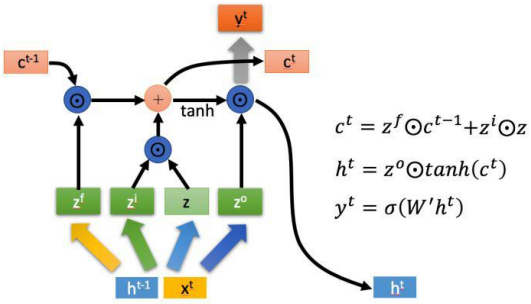
3.2 Long short-term memory

Long short-term memory (LSTM) is a special RNN, mainly to solve the gradient disappearance and gradient explosion problems in long sequence training. Simply put, LSTM can perform better in longer sequences than normal RNN.

The main input and output differences between the LSTM structure and the normal RNN are as follows.



Compared to RNN, which has only one transfer state h^t , LSTM has two transfer states, one c^t (cell state), and one h^t (hidden state). (Tips: h^t in RNN for c^t in LSTM)
The c^t that is passed down is changed very slowly. Usually, the output c^t is the c^{t-1} passed from the previous state plus some values.
However, h^t tends to be very different under different nodes.



There are three main phases within LSTM:
1. Forget the stage. This phase is mainly to selectively forget the input from the previous node. Simply put, it will be “forgetting not important, remembering important ones”.
Specifically, it is calculated by z^f (f is a forget) as a forgotten gate to control the previous state of c^{t-1} which needs to be left for forgetting.
2. Select the memory phase. This phase selectively "memorizes" the inputs to this phase. Mainly, the input x^t is selected and memorized. What are important are

recorded and those that are not important are less. The current input is represented by the z calculated previously. The selected gating signal is controlled by z^i (i stands for information).
Add the results obtained in the above two steps to get c^t transmitted to the next state. This is the first formula in the above picture.
3. Output stage. This phase will determine which outputs will be treated as current states. Mainly controlled by z^o. And the c^o obtained in the previous stage is also scaled (changed by a tanh activation function).
Similar to a normal RNN, the output y^t is often also obtained by h^t variation.

3.3 Gate Recurrent Unit

GRU (Gate Recurrent Unit) is a kind of Recurrent Neural Network (RNN). Like LSTM (Long-Short Term Memory), it is also proposed to solve problems such as gradients in long-term memory and back propagation.
GRU internal structure:
First, we first get the two gating states by the last transmitted state h^{t-1} and the current node's input x^t. As shown in Figure 2-2 below, where r controls the reset gate and z is the update gate.
Sigma is a sigmoid function, by which the data can be transformed into a value in the range of 0-1 to act as a gating signal.

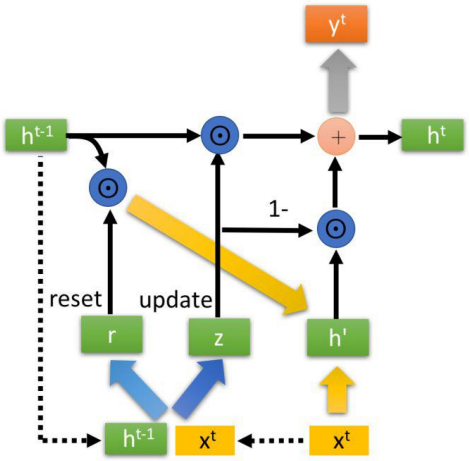
$$r = \sigma(W^r \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix})$$

$$z = \sigma(W^z \begin{bmatrix} x^t \\ h^{t-1} \end{bmatrix})$$

After getting the gating signal, first use the reset gating to get the data after the "reset" {h^{t-1}}' = h^{t-1} ⊙ r , and then {h^{t -1}}' is spliced with the input x^t, and then the data is scaled to a range of -1 to 1 by a tanh activation function. That is, h' is obtained as shown in the figure below.

$$h' = tanh(W \begin{bmatrix} x^t \\ h^{t-1'} \end{bmatrix})$$

Here h' mainly contains the x^t data currently input. Targeted addition of h' to the current hidden state is equivalent to "memorizing the state of the current moment." Similar to the selection memory phase of LSTM.



⊙ is the Hadamard Product, which is the multiplication of the corresponding elements in the operation matrix, so the two multiplication matrices are required to be homotype. ⊙ stands for matrix addition.

Finally, we introduce the most critical step of GRU, which we can call the "update memory" stage.

At this stage, we also carried out two steps of forgetting the memory. We used the previously obtained update gate z (update gate).

Update expression: h^t = z ⊙ h^{t-1} + (1 - z)⊙ h'

First of all, the gating signal (here z) ranges from 0 to 1. The closer the gating signal is to 1, the more data represents "memory"; the closer to 0, the more "forgotten".

One of the smart things about GRU is that we use the same gating z to forget and select memory at the same time (LSTM uses multiple gating).

z ⊙ h^{t-1} : Indicates the selective "forgetting" of the original hidden state. Here z can be imagined as a forget gate, forgetting some unimportant information in the h^{t-1} dimension.

(1-z) ⊙ h' : Indicates selective "memory" of h' containing the current node information. Similar to the above, here (1-z) will forget some unimportant information in the h ' dimension. Or, here we should look at it as a choice of some information in the h' dimension.

h^t = z ⊙ h^{t-1} + (1 - z)⊙ h' : In combination with the above, the operation of this step is to forget some of the dimensional information in the passed h^{t-1} And join some dimension information entered by the current node.

It can be seen that the forgetting z and the choice (1-z) are linked. That is to say, for the dimensional information passed in, we will selectively forget, and how many weights (z) are forgotten, we will make up (1-z) with the weight corresponding to h' containing the current input. To maintain a "constant" state.

3.4 Word embedding

Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where words or phrases from the vocabulary are mapped to vectors of real numbers. Conceptually it involves a mathematical embedding from a space with one dimension per word to a continuous vector space with a much lower dimension.

3.5 Logistic regression

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

3.6 Support Vector Machine

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space(N—the number of features) that distinctly classifies the data points.

3.7 Random Decision Forests

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

3.8 XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework.

4 Code

The code in part A is similar with assignment 1 (https://github.com/nikbearbrown/CSYE_7245/blob/master/Assignments/Assignment_1_Properly_Format_and_Explain.ipynb). Here I want to focus on the code of part B.

In part B, first we create a dictionary containing 130,000 words and convert them to lowercase. Then we fit our data in the dictionary and change the text to sequence, because word embedding only accept sequence.

```
tk = Tokenizer(num_words = 130000, lower = True)
gen_txt=train_text_five.str.lower()
tk.fit_on_texts(gen_txt)
generate = tk.texts_to_sequences(gen_txt)
```

The following is our base model which is very simple. First we use word embeddings pack more information into far fewer dimensions and then it is the only layer of the model GRU which is introduced above.

```
model = Sequential()
model.add(Embedding(130000,32))
model.add(layers.GRU(32,
                     dropout=0.1,
                     recurrent_dropout=0.5))
model.add(Dense(len(word_index), activation='softmax'))

model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy')
history = model.fit(X_gen, y,
                    epochs=3,
                    batch_size=512)
```

Training the language model and sampling from it

Given a trained model and a seed text snippet, we generate new text by repeatedly:

1) Drawing from the model a probability distribution over the next character given the text available so far

2) Reweighting the distribution to a certain "temperature"

3) Sampling the next character at random according to the reweighted distribution

4) Adding the new character at the end of the available text

This is the code we use to reweight the original probability distribution coming out of the model, and draw a character index from it (the "sampling function"):

```
def sample(preds, temperature=1.0):
    preds = np.asarray(preds).astype('float64')
    preds = np.log(preds) / temperature
    exp_preds = np.exp(preds)
    preds = exp_preds / np.sum(exp_preds)
    probas = np.random.multinomial(1, preds, 1)
    return np.argmax(probas)
```

Then we generate sentences by above model and function. In the part B, we focus on the model structure. We change the model structure to see how it will have an effect on the accuracy.

5 Results

Text Classfication:

Algorithm	Accuracy	Hyperparameters
Logistic Regression	79.92%	
XGBoost Classifier	81.33%	max_depth=8, n_estimators=350
Random Forest Classifier	83.89%	'max_depth': None, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 200
RNN(Recurrent Neural Network)	84.34%	optimizer='rmsprop', loss='binary_crossentropy', activation='sigmoid', one layer: GRU

We can see in the table that the accuracy of RNN is the highest in text classification.

Sentence Generator:

Process Data	Model structure	Accuracy
(408280, 20,128)	First Layer: GRU(128), Second Layer: convolution layer(128)	5.52%
(213550, 60,128)	First Layer: LSTM(128)	14.04%
(408280, 20,128)	First Layer: LSTM(128), Second Layer: LSTM(128)	15.64%
(408280, 20,128)	First Layer: GRU(128), Second Layer: LSTM(128)	16.02%

(408280, 20,128)	First Layer: GRU(128), Second Layer: GRU(128)	16.75%
(408280, 20,128)	First Layer: LSTM(128)	17.23%
(408280, 20,128)	First Layer: GRU(128)	17.53%
(408280, 20,128)	First Layer: GRU(128), Second Layer: GRU(256)	17.54%
(408280, 20,128)	First Layer: GRU(128), Second Layer: GRU(256), Third Layer: GRU(512)	17.61%
(408280, 20,128)	First Layer: GRU(256)	18.27%
(408280, 20,128)	First Layer: GRU(512)	18.99%

The best model structure is one GRU layer and its parameter is 512 and its accuracy is 18.99%.
Below is the sentence generated by our model:

Input: he can keep up his energy keep his weight up and recover more quickly from bumps from running the field
Output: he can keep up his energy keep his weight up and recover more quickly from bumps from running the field individually creamy 30 same coffee is sauces deal to quality excellent flavor thing extra least counter indulgence ever blends half snacks quality food the syrup food combination deep recipe deal medium banana compares size start suitcase without gluten calcium crashing lamb stuff paper after once absolutely red boy affect cup

Input: nutrition but also because they seem to be more filling this is now my favorite if you purchase the fiber
Output: nutrition but also because they seem to be more filling this is now my favorite if you purchase the fiber my coffees pepper artificial paying course dark wonderful hips at seems cookies avid disappointed blueberry stand condition locally offering snacks mainly take coats ended enjoy big work half myself reviews buy beginning when energy keeps gives part stomach acidic vitamin mouth same ingredients food fill tasted plate cracker product baking

Input: is on a meat free diet due to severe food allergies it seems to have very good quality ingredients and
Output: is on a meat free diet due to severe food allergies it seems to have very good quality ingredients and acne considered cheaper nutrition grew allergies for wake waffle nuts consider good simple squirt to air told for 20 hip for called great recommended ingredients hope diet reviews thing coffee it portions meal 'em chowder milk says let texture taste often fall which fairly aren't mahogany way medication maple bloating

Input: you add butter and an egg that's it you want to bake them until the bottom

of the cookie is
Output: you add butter and an egg that's it you want to bake them until the bottom of the cookie is breakfast price satisfying olive varieties trash drinks groups work quality useful capsule call dont maybe scoops plant health calories football not whipped price dog blender to bottom chips buying extra at deliciousness clear price vernor's food heaven use gravy biscuits step thins mg leave tend deal pamela's perfectly freeze skeptical

Input: to see this drink in the market although i would have liked a larger size container so i could drink
Output: to see this drink in the market although i would have liked a larger size container so i could drink p baking crisp bite or now grab reviewers choice buck cures watcher dogs packaging act oats short slow 24 blend prepared review he garlic for portion least delivery am flavor appealing have frontier particular for roll manganese ingredients aroma sandwiches live higher tasty health heat fruit supplement foods price sized

The result is not good. We do some researches on it, other sentences generated by text generator are also not good. So we should do more researches and experiments to create better sentences in the future.

6 Discussion

Here are several conclusions:

RNN(Recurrent Neural Network) really does a good job in text area.

It is wrong that the more layers, the better the model will be. We think it may be vanishing gradient problem in our model. When the model is three layers(128/256/512), its accuracy will be lower than one layer(512). Because the learning rate of first layer and second layer are low that will have a bad effect on the third layer.

The model running time is mainly related to the parameters of the layer with the largest parameter in the model.

When we use low creativity, the generated word will be the highest appear frequency in the model. However, when we use the higher creativity, the generated word will be diverse.

References

1. Diego Alejandro Salazar, Jorge Iván Vélez, Juan Carlos Salazar (2012) Comparison between SVM and Logistic Regression: Which One is Better to Discriminate? Revista Colombiana de Estadística volume 35, no. 2, pp. 223 a 237, http://www.kurims.kyoto-u.ac.jp/EMIS/journals/RCE/V35/v35n2a03.pdf
2. Tianqi Chen， Carlos Guestrin XGBoost: A Scalable Tree Boosting System, https://www.kdd.org/kdd2016/papers/files/rfp0697-chenAemb.pdf
3. Leo Breiman (2001) RANDOM FORESTS University of California, https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf
4. Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, Wang-chun Woo Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting, https://papers.nips.cc/paper/5955-convolutional-lstm-network-a-machine-learning-approach-for-precipitation-nowcasting.pdf
5. Alex Graves, Santiago Fernandez, Jurgen Schmidhuber (2013) Multi-Dimensional Recurrent Neural Networks IDSIA, https://arxiv.org/pdf/0705.2011v1.pdf
6. Ilya Sutskever (2013) TRAINING RECURRENT NEURAL NETWORKS University of Toronto, http://www.cs.utoronto.ca/~ilya/pubs/ilya_sutskever_phd_thesis.pdf
7. Xiang Li, Tao Qin, Jian Yang, Tie-Yan Liu (2016) LightRNN: Memory and Computation-Efficient Recurrent Neural Networks NIPS, https://papers.nips.cc/paper/6512-lightrnn-memory-and-computation-efficient-recurrent-neural-networks.pdf
8. Michael Nguyen (2018) Illustrated Guide to LSTM's and GRU's: A step by step explanation TowardsDataScience, https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44c9eb85bf21
9. Text generation with LSTM, https://nbviewer.jupyter.org/github/fchollet/deep-learning-with-python-notebooks/blob/master/8.1-text-generation-with-lstm.ipynb
10. Vanishing gradient problem, https://en.wikipedia.org/wiki/Vanishing_gradient_problem
11. Chen Chen (2018) LSTM that everyone can understand. Zhihu, https://zhuanlan.zhihu.com/p/32085405
12. Chen Chen (2018) GRU that everyone can understand. Zhihu, https://zhuanlan.zhihu.com/p/32481747