Natural Language Processing

Chennai Mathematical Institute

Building model from Scratch for Sentiment Analysis

Data

We have textual data. These are some reviews or feed-backs. Every review is made up of several sentences. Some sentences may exhibit positive sentiments while some may exhibit negative sentiments. It is also possible that some sentences may not have any polarity information associated with them.

Generally we have 3 sentiment categories -

- Positive : 1
- Negative : -1
- Neutral : 0

We can also consider Five different categories of sentiment. These are in the following -

1. Strongly Positive 2.Moderately Positive 3. Strongly Negative 4. Moderately Negative 5. Neutral
Here 3 labels of sentiment is considered.

Task

Here our task is to do sentiment analysis which is one of interesting problems of Natural Language Processing. As each review is made up of several sentences, we also want to identify the bias in a review. So we want to find consolidated polarity for a review considering the polarities of all the sentences of the review.

Pre-processing

This is a very important step when we have a textual Data. Before feeding the data into the model we do several pre-processing steps. We transform the data to some numerical features which can be fed to the neural network model. This pre-processing step is domain specific and it changes based on the data we have and task we need to do. For this sentiment analysis task we do the following pre-processing steps -

- Here we want to find the polarity sentence wise. So first we need to split the reviews in sentences. Then we do the pre-processing tasks for each of the sentences of the data.
- We convert the text into lower case. It helps us to reduce the size of vocabulary and get rid of some unnecessary repetition or different forms of same word in the vocabulary. In some cases we may need the Casing information. In that case we can adapt different techniques to store the casing information.
- We remove the numbers as it doesn't effect much in determining the sentiment. We can also convert it to words in case we need to store this information also.
- We remove all the punctuations and extra white spaces from the texts. Removing punctuations again help to avoid different form of the same word in the vocabulary.
- We need to convert the text into tokens.
- We then remove the stop words for the language from the vocabulary list as it doesn't contribute to the meaning of a sentence.

- Then we do stemming which helps us to get the root form of a word. By removing prefix or suffix we get the stem of a word. It doesn't always give us an actual word.

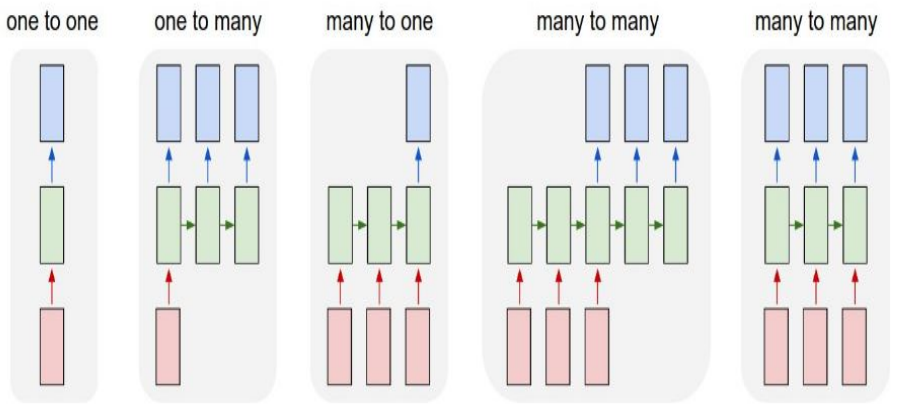
  Different types of stemmers are available. We can use the Porter Stammer for the task.

- Lemmatization also gives us the root word like stemming does but it gives a valid root word which belongs to the language we can working on.
- In this way we create our vocabulary.
- Now we need to convert the text(words of a sentence) to some numeric features so that we can feed it to our model.
- We got all the tokens corresponding to a text after pre-processing.
- Now we can use several approaches to convert the tokens to numeric features. Here are some of them -
  - We can use word2vec to convert the token to some numeric vectors.
  - We can use GloVE.
  - We can also use a simple way, one hot encoding.
    As the vocabulary can be huge and all words are not important in identifying sentiments, we can do the following.
    * We compute the TF-IDF values of the words present in all the reviews in the training data.
    * Based on the decreasing order of the TF-IDF values we sort our vocabulary.
    * Then based on total no of unique words and using some intuition we choose the size of the vocabulary and keep as many words from the beginning of the sorted vocabulary.
- So we finally get the numeric representation of the tokens which we will feed to the RNN model.

This is how we pre-process the text and make out input ready to feed our model.

Model

We use different RNN models for different NLP tasks. These tasks can be divided in some categories shown in the following picture -



Our task of sentiment analysis is "Many to One". As we feed the tokens of the text sequentially and after the last token of the text is fed to the network we get our output.

Input:

Text corresponding to the sentence of a review. So we have a text as an input and this text contains the tokens, the numeric feature corresponding to the tokens are fed sequentially.

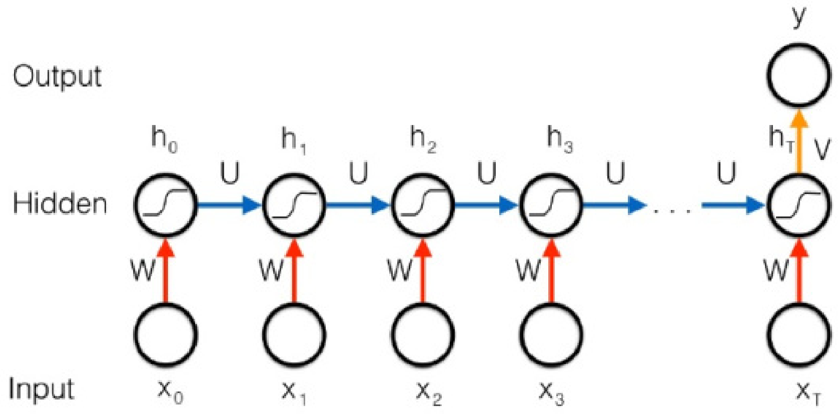
- If we use one hot encoding representation of words then the input size depends on the vocabulary size.
- If we use word2vec or GloVE to get the numeric representation of words then it depends on the no of dimension we take.

Output:

The output corresponding to a sentence of a review will be one of 3 categories of sentiment. Output layer will have nodes equal to the number of categories of sentiment. Here we have chosen our review to fall in one of 3 categories. So output layer will have 3 nodes.

Design of the network

So we have a "Many to One" model. Here is the network if we unfold that diagram -



- x_t is the input at the t_{th} time step.
- h_t is the hidden layer output at t_{th} time step.
- y is our final output after the final T_{th} step.

Here in this sentiment analysis task, We get our output after the final time step. We are not getting output at each time step as it is a "Many to One" RNN model.

We can see from the figure that RNN uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

The same weight matrix "W" is used at each time step when we feed the current input from input layer to hidden layer.

The same weight matrix "U" is used at each time step when we feed the previous hidden layer output to current hidden layer.

For the final hidden layer and output layer, a weight matrix "V" is defined.

Activation Function:

We can choose our activation function. Sigmoid, softmax, tanh, Relu are some activation functions. Here,
In hidden layer: tanh
In output layer: Softmax

Cost Function: Categorical Cross-entropy

$$L(y, p) = -\sum_{i \in C} y_i \log(p_i)$$

This loss function calculates the loss for one input and C categories. Here p is the predicted values for given input and y is the actual output.

Why this Model Architecture:

For this sentiment analysis task we have chosen this ”Many to One” Recurrent Neural Network. In simple machine learning algorithms and also in the traditional neural network the inputs and outputs are totally independent. But in these type of tasks the meaning of sentence is important to identify the review. So in these tasks the sequence of words is really important, not only the presence or absence of words. RNN remembers each and every information through time. At each time step encoded words are provided as input sequentially. Due to the shortcomings and the memory of RNN, this is preferable in these type of tasks over simple machine learning algorithms and traditional neural network. RNN uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks. As we have 3 categories we have used categorical cross-entropy and in the output layer softmax activation is used to compute the probabilities. The output from tanh can be positive or negative, allowing for increases and decreases in the state. That's why tanh is used to determine candidate values to get added to the internal state. That is why Tanh is used in the hidden layers. We can also play with the learning rate involved while updating the parameters of the model.

Consolidated Polarity and Bias Identification:

Here we want to compute the polarity as well as bias in the reviews. If there are more sentences with negative sentiments for a review and if the label was positive, then there is a bias in the review and vice-versa. For this purpose, here we are computing polarity for each of the sentence of the review and combining all these polarity information we are computing consolidated polarity. Now we want to find a way to compute this consolidated polarity.

Now say we have a review which contains m no of sentences. Implementing RNN we find the polarity for each of these m sentences. The polarity can be Positive(1), Negative(-1) or Neutral(0). No we just do a simple count. So we had m sentences in the review. Say the labels of these sentences are l_1, l_2,...,l_m. Now we can count the no of positive, negative sentiment sentences. We find that there are ”P” positive sentiment sentences and ”Q” negative sentiment sentences. Now say the true label of the review is T. T can be 1,0 or -1. Now we say that there is a bias in the review if one of the following holds-

- T = 1 and P <Q
- T = -1 and P > Q

We can do the above in a similar way. We can just take the sum of the labels(here integers - either 1, 0 , -1) i.e. we compute $\sum_{i=1}^{m} l_i$. We can name it the consolidated polarity as it is obtained by considering the sentiments of all the sentences of a review. So consolidated polarity, CP is given by -

$$CP = \sum_{i=1}^{m} l_i \quad or \quad CP = \frac{1}{m}\sum_{i=1}^{m} l_i$$

Now the rule of determining bias is in the following if one of the following holds -

- T = 1 and CP < 0
- T = -1 and CP > 0

Procedure

RNN is a special type of Artificial neural network where the output from previous step is also considered with the input f the current step. Here we are working with ”Many to One” RNN. So we feed the numeric features corresponding to tokens of a text sequentially to the network and we get the output after the final token of the text is fed to the network.
In our ”Many to One” network along with the current input we use the output of hidden layer of the previous time step as the input.
RNN have a “memory” which remembers all information about what has been calculated. It uses the same parameters for each input as it performs the same task on all the inputs or hidden layers to produce the output. This reduces the complexity of parameters, unlike other neural networks.

Training through RNN:

- First we need to initialize the parameters based on the input, output and hidden layer dimension.
- Then the current input and previous state output of the hidden layer is provided as input in the current step and we calculate h_t.
- Then this h_t from the last step becomes input for the next step along with the input at that time step.
- After all the tokens of the text are fed i.e all the time steps are completed we calculate the output in the final step.
- We get the probabilities corresponding to each category of sentiment after using softmax activation on the final output.
- We compare the output with the actual output and calculate the loss using categorical cross-entropy function.
- We then calculate the gradients of the loss w.r.t the parameters of the model using chain rule.
- We then use the gradient descent algorithm to update our parameters. We can use -
  - Batch Gradient Descent
  - Stochastic Gradient Descent
  - Mini-batch Gradient Descent
- The update rule involves a hyper-parameter,Learning rate.
- So using this Backpropagation through time(BPTT) and Gradient Descent algorithm, we update our parameters.
- We choose no epochs and run the above steps for those many epochs.

Testing

- From the training we learn all the parameters of the model. For a given text, based on these learned parameters we calculate our output by the same algorithm we used during the forward propagation while training.
- For all the test data we have, we compute our output for each of the sentence.
- We compute the consolidated polarity for a review from the polarities of sentences.
- Then we compute the total loss and no of correct predictions,hence the accuracy.

Mathematical Results and Formulas

Cost/Loss Function:

$$L(y, p) = -\sum_{i \in C} y_i \log(p_i)$$

This loss function calculates the loss for one input and C categories. Here p is the predicted values for given input and y is the actual output.

Forward Propagation:

Using the Following formulas all the values like hidden units, output, probabilities and loss is calculated.

$$
\begin{aligned}
Z_t &= W_{xh}x_t + W_{hh}h_{t-1} + b_h \\
h_t &= \tanh(Z_t) \\
y &= W_{hy}h_T + b_y \\
p &= \text{softmax}(y)
\end{aligned}
$$

h_t is the output of the hidden layer at time step t after applying tanh activation to Z_t. y is the output in the output layer and p is the vector of probabilities corresponding to each category obtained by taking softmax of y.

Backward Propagation:

Here are some calculations(chain rule) and results of the gradients of loss w.r.t model parameters -

$$\frac{\partial L}{\partial y_i} = \begin{cases} (p_i - 1)\,, i = c \\ p_i, i \neq c \end{cases}$$

$$\frac{\partial L}{\partial W_{hy}} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial W_{hy}} \quad and \quad \frac{\partial y}{\partial W_{hy}} = h_T$$

$$\frac{\partial L}{\partial b_y} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial b_y} \quad and \quad \frac{\partial y}{\partial b_y} = 1$$

$$\frac{\partial L}{\partial W_{xh}} = \frac{\partial L}{\partial y} \sum_t \frac{\partial y}{\partial h_t} \cdot \frac{\partial h_t}{\partial W_{xh}} \quad and \quad \frac{\partial h_t}{\partial W_{xh}} = (1 - h_t^2) \cdot x_t$$

$$\frac{\partial L}{\partial W_{hh}} = \frac{\partial L}{\partial y} \sum_t \frac{\partial y}{\partial h_t} \cdot \frac{\partial h_t}{\partial W_{hh}} \quad and \quad \frac{\partial h_t}{\partial W_{hh}} = (1 - h_t^2) \cdot h_{t-1}$$

$$\frac{\partial L}{\partial b_h} = \frac{\partial L}{\partial y} \sum_t \frac{\partial y}{\partial h_t} \cdot \frac{\partial h_t}{\partial b_h} \quad and \quad \frac{\partial h_t}{\partial b_h} = (1 - h_t^2)$$

So we have got the equations to compute the gradient of loss w.r.t all the parameters. We use the following two equations to get our required result -

$$\frac{\partial y}{\partial h_t} = \frac{\partial y}{\partial h_{t+1}} \cdot \frac{\partial h_{t+1}}{\partial h_t} = \frac{\partial y}{\partial h_{t+1}} \cdot (1 - h_t^2) \cdot W_{hh} \quad and \quad \frac{\partial y}{\partial h_T} = W_{hy}$$

Softmax Activation

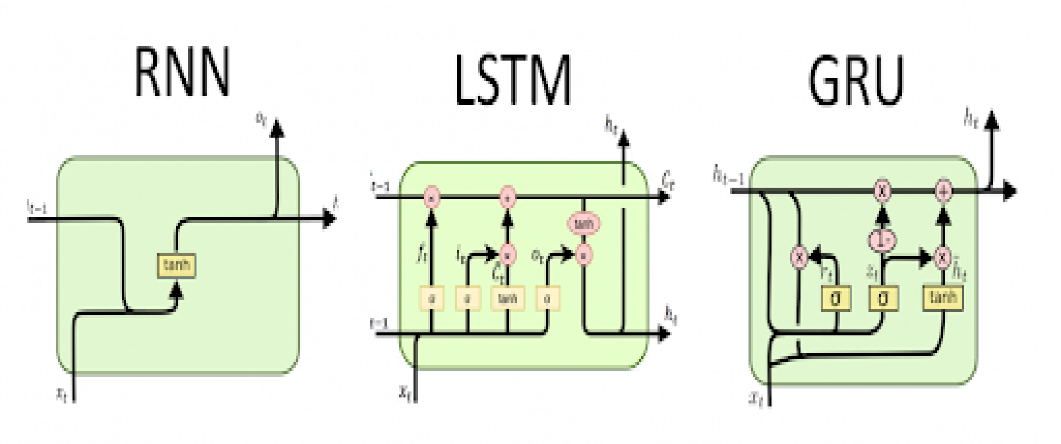
For the i th class,

$$softmax(x)_i = \frac{exp(x_i)}{\sum_j exp(x_j))}$$

It gives us the probabilities for each of the classes, here for each of the sentiment labels.

Comments:

Implementing RNN we get the polarity for each of the sentence of the review. Then using the consolidated polarity index we determine whether there is bias in the review or not. We have used RNN which has several advantages over traditional neural network. We can also use Long Short Term Memory which has a 'memory cell' that can maintain information in memory for long periods of time and used extensively in these tasks. We can also use Gated Recurrent units which also shows promising results in these type of NLP tasks. Following is the figure of these networks -


RNN
LSTM
GRU
tanh
σ
tanh
x_t
h_t
h_{t-1}
C_t
f_t
i_t
o_t
r_t
z_t
$\tilde{h}_t$