Data Background and Preliminary Analysis

Data Description

There are 1546379 yelp data and it has 8 variables: stars, name, text, city, longitude, latitude and categories. Our goal is to predict stars rating through comment texts and other important information. For stars distribution, there are 5 types of stars. For star 1, there are 164676 data with proportion 10.6%; for star 2, there are 152401 data with proportion 9.9%; for star 3, there are 225710 data with proportion 14.6%; for star 4, there are 443599 data with proportion 28.7%; for star 5, there are 559993 data with proportion 36.2%. The five-star business accounts for the most proportion and low-star business are less proportion.
For the length of reviews distribution, it approximately follows a Poisson distribution. The reviews more than 300 words are pretty little, so we prefer to discard these data.

Data Cleaning

There are 392712 unique words and it's too much for us. For data cleaning, we mainly focus on two types of word: low frequency words and high frequency words. For the high frequency words, since we would like to use neural network, we discard some high frequency words to increase our model efficiency and accuracy. We drop these 8 meaningless words: "the", "to", "that", "this", "on", "at", "with", "of". For low frequency words, we would like to discard words whose frequency less than 50 and there are approximate 40,000 unique words remained. After we check some example we discarded, they are messy words, foreign language and spelling mistakes.

Word Embedding and LSTM Neural Network

Below we will first carry out preliminary clean-up of the text and then use the function (reference) provided in the R language keras and tensorflow package to construct the neural network. We convert the word into a word vector, and then train the LSTM (Long Short Term Memory) neural network to implement the classification of the text data.

Data Transformation

We removed all punctuation and converted all uppercase letters to lowercase give each word we have selected a separate integer label. For each text, we remove the words that are not in the first 40,000 in frequency and convert the remaining words into corresponding integer labels so that each text becomes an integer vector. Then for a vector whose length longer than 300, it is truncated to a 300-dimensional vector. For an integer vector less than 300 in length, add 0 at the end to make it a vector of length 300. In this way, each text is converted to a 300-integer vector with a total of 40,000 possible values for each component.

Word Embedding

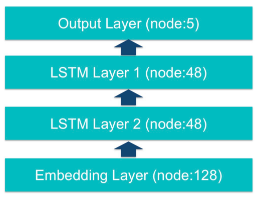
Since the component of the integer vector is a categorical variable, the actual dimension corresponding to each text is 300 * 40000, so we use the word embedding method to convert it into a word vector to reduce the dimension. We found that 500G memory is needed to directly convert text into a word vector, so we add a 128-node text layer to directly convert text into a word vector in the neural network after our input layer of 40,000 nodes. In this way, we translate the words that actually correspond to 40000-dimensional vectors into 128-dimensional word vectors.

LSTM Neural Network

Motivation

Long Short Term Memory networks - usually just called "LSTMs" - are a special kind of RNN, capable of learning long-term dependencies. The LSTM does have the ability to remove or add information to the cell state, a place where information is integrated.

How it works



We added two LSTM layers after the text layer and added an output layer with five nodes at the end corresponding to different stars. The 128-layer word vector obtained in the text layer will be input into the LSTM layer. After processing the two LSTM layers, each node in the output layer will be assigned a value between 0 and 1. The larger the node corresponding value, the more likely the neural network tends to classify text into this category. In order to obtain the parameters required by the neural network, we consider that each LSTM layer can select 32 to 64 nodes through an empirical formula (reference), and the dropout rate can be found to be 0.2 to 0.5 by consulting the reference. We use a sample size of 10000 to train the neural network multiple times with different parameter combinations. At the same time, we balance the relationship between accuracy and training time. Finally, we decided to use the dropout rate 0.2 and 64 nodes for each LSTM layer, and the epoch number to use 4 times to select the frequency. For words we have selected, we transform them into 128-dimensional word vectors. After the neural network is trained, we use it to classify each text in the test set, the RMSE of the result is found to be 0.68. However, we do not use the LSTM classification result directly. The five values of the output layer corresponding to each text are used as new input variables of the following regression tree.

Regression Tree

Motivation

Then, we want to include more information to improve our model. Since regression tree does not require data normalization or statistical assumptions, and it works great on large datasets (good accuracy in reasonable time), so we consider using a regression tree model.

Regression Tree Model and Results

We decide to use the 5 output values of the LSTM neural network as well as date, longitude, latitude and length of the sentence (in words) as the input of our regression tree model. Date is represented by UNIX Epoch time, that is, the number of seconds that have elapsed since 1970-1-1 00:00 (UTC). For example, the date 2018-03-01 could be represented by number 1519884000. Longitude and latitude are in the same form as the raw data. Length of the sentence (in words) is counted without removing any stopwords or low-frequency words. We use the eXtreme Gradient Boosting algorithm (a.k.a, xgboost) to train the regression tree. After trying different parameters combination on a small subset of the training data, we decide to use $learnrate = 0.1$, $eta = 0.5$, and 80 iterations, and the depth of the tree is 7. Finally, we train the regression tree on the whole training dataset and use the trained tree model to obtain our final predictions. The RMSE of the final predictions is about 0.566. Compare to the simple LSTM classification result, we could say the model performs much better.

Sentiment Analysis

Analysis by LSTM Output

For one text, LSTM model will give 5 output values with respect to five different stars. And， If neuron network "thinks" text from one specific star with high probability, the corresponding output value will be high. Therefore, if the corresponding value is the highest, it indicates that the neural network determines the score corresponding to the node. In the process of inputting texts, the neural network will update as long as a word is inputted. Therefore, the judgment of the neural network's score on the text will continue to change. We can analyze the emotional changes of text by tracking how the nodes change.

Find Hidden Sentiment in Word

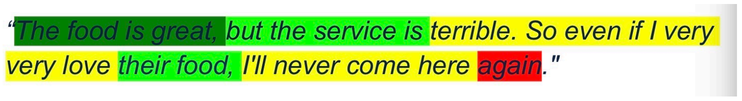
Since our input is a word, neural network will give output only based on the word. We can define the sentiment value of this word as the number of stars which corresponds to highest output value. For example, the output of text "delicious" is (0.002, 0.007, 0.062, 0.395, 0.620), so its sentiment value is 5, which means it is strongly positive.

delicious	good	good but	water	pour
5	4	3	1	4
3 star	steak	flavor	come again	silent
3	5	5	5	1

From this we can see that "delicious", "come again", "steak", "flavor" are strong positive words; "3 star" is neutral words; "pour" is positive words. Surprisingly, "good" is just a weak positive word. "Water", "silence" is a strong negative word. This shows that some words will show yelp emotions that are not in the usual context. "Good" and "but" will become neutral words when matched, which shows that but has the effect of changing emotional tendencies.

Find Hidden Sentiment in Text

In discussing the emotional orientation of text, we also need to combine the context of the text, for example, Very delicious" and "Not delicious" indicates different sentiment tendency. For one text, we can put first N words to neural network, N from 1 to length of this text, and record sentiment values to find how sentiment tendency changes in this text. For the next example, We will "visualizing the value of the sentiment cell as it processes texts. Red indicates negative sentiment while green indicates positive sentiment."


"The food is great, but the service is terrible. So even if I very very love their food, I'll never come here again."

In this artificially constructed example we can see how the neural network's judgment of textual sentiment changes. In the absence of any information, the neural network tends to assume that the text corresponds to a five-star rating because the five-star rating in the yelp data is the most. When "food is great" appears, the neural network thinks that emotion is strongly positive. When "but" appears, the neural network emotion judgment changes immediately. When the keyword "terrible" appears, the neural network assumes that the text may correspond to a neutral evaluation. When it comes "very very love their food", the evaluation picks up, but when it comes to "never coming back", the neural network determined that the text was negatively rated.

Conclusion and Model Evaluation

Advantages

1.The model is accurate. The RMSE of our final prediction is about 0.566, which is relatively small. 2.The model does not based on any statical assuptions, that is, our data do not need to follow certain distribution, or have specified pattern.

Disadvantages

1.The training process of the LSTM neural network costs plenty of time. We spend over 30 hours to train our LSTM model. 2.The model is difficult to interpret. The LSTM part is a black-box model, and the regression tree is complicated(since the depth is 7, there are more than 100 nodes in the tree). Therefore, this model is not simple to interpret.