Data Engineering

1. Data Introduction

We get our data from [Dr. Julian McAuley](http://cseweb.ucsd.edu/~jmcauley/)’s [Amazon dataset](http://jmcauley.ucsd.edu/data/amazon/).

The dataset contains product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996 - July 2014.

The dataset includes reviews (ratings, text, helpfulness votes), product metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also bought graphs).

For our project, we choose the data only for electronics, which contains 1689188 reviews. Since our project focus on text classification. We only keep the review texts and ratings in the data. Here is an example of the data.

|  |  |  |
| --- | --- | --- |
| Index | Rating | Review |
| 1 | 5 | I bought this for my husband who plays the piano. He is having a wonderful time playing these old hymns. …… |

*Table: one example of the data*

2. Data cleaning

Before we start fitting the model, some data cleaning processes are necessary. Because, if we do not clean the raw texts and use them directly, the dimension of the feature matrix will be ultra-high, which will not only slow down the modeling fitting, but also reduces the accuracy.

Our data cleaning processes contain the following step:

* Remove punctuations except ! and ?
* Transfer into lower case
* Remove stopwords
* Normalize the verbs and nouns

In the first step, we keep “!” and “?” because these two punctuations contain strong emotions. In the second step, there may be some all-capital words in the reviews to show the reviews’ angry or surprise. But it is hard to recognize, so we ignore them, which may cause a very small loss in accuracy. In the third step, a stopwords list is basically a collection of most common used words. But here we edited this list to keep the words such as “never” that we think contain strong emotions. In the fourth step, we do this because we want to make our feature matrix smaller.

3. Feature matrix

In this part, we will transfer the cleaned review texts into different kinds of feature matrix that could be put into all kinds of models. In this project, we will mainly use two kinds of methods to do this step.

The first one is TF/IDF method. TF/IDF method basically gives one score for every word to vectorize the texts. In detail, we remove all words with occurrences under 3 in the whole text as well as frequency above 0.9 in the whole text. These two thresholds are learned from our previous project on text classification. What’s more, we consider both single words and the combination of two words.

The second one is bag-of-word method. In this method, we just keep top 50000 frequent words and count their occurrence in every review. Again, the threshold 50000 is choose from the previous experience.

Here are examples to show how final feature matrixes look like.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | word1 | word2 | …… | word50000 |
| 1 | 0.2312 | 0.1983 | …… | 0.7823 |
| 2 | 0.6547 | 0.0023 | …… | 0.8731 |
| …… | | | | |
| 1689188 | 0.1231 | 0.6542 | …… | 0.0025 |

*Table: one example of the TF/IDF feature matrix*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Index | word1 | word2 | …… | word50000 |
| 1 | 1 | 0 | …… | 2 |
| 2 | 0 | 1 | …… | 0 |
| …… | | | | |
| 1689188 | 0 | 0 | …… | 1 |

*Table: one example of the bag-of-word feature matrix*

LSTM model

In this part, we will explore how deep neural network model with a long short time memory(LSTM) layer works for our data.

Firstly, we learn the architecture of the model. It is very hard to learn the structure of the model, especially for deep learning model. So, in this project, we just use the simplest structure: one word embedding layer, one LSTM layer and one full connection layer to output the prediction.

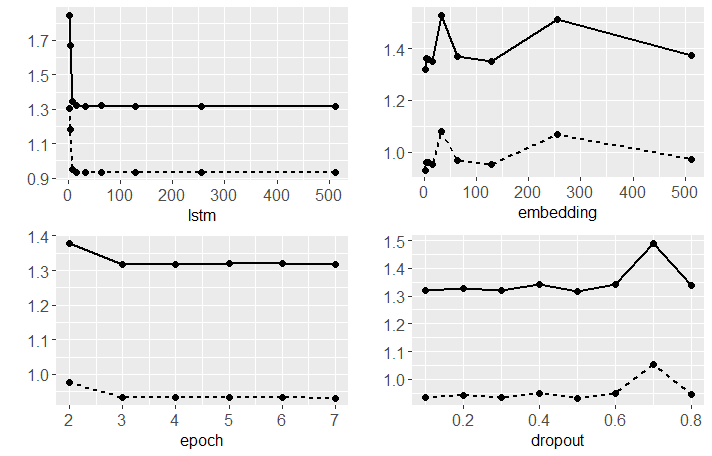
Parameter tuning part could be very crucial to this method. But we do not have enough time to do parameter tuning for all sample sizes. So, we only do parameter tuning for size = 100000 and use this setup for all sizes. This may affect the accuracy of our models.

For our LSTM model. We care about four parameters most. They are:

* Number of kernels in word embedding layer
* Number of kernels in LSTM layer
* Dropout rate
* Number of epochs

We did four experiments for the four parameters. Here is the setup for the experiments.

* Train on 10000 data
* Test on 5000 data
* Record the RMSE of train data and test data



*Figure: Parameter tuning for LSTM model, dash line for RMSE of train set, solid line for test set*

Based on the plots, here is our final model setup:

* Dropout rate: 0.2
* Number of epochs: 4
* Number of kernels in embedding: 128
* Number of kernels in LSTM network: 128

For the activation function and loss function, we look up for literature and use sigmoid function for activation function and mean square root function for loss function.

By now we have our model ready. We will use this model to do evaluation in the next part.