Report: Eluvio DS Challenge NLP

Md Jonybul Islam
mdjonybulislam@gmail.com

1 INTRODUCTION

Natural language processing (NLP) is a branch of artificial intelligence that helps computers understand, interpret, and manipulate human language. NLP draws from many disciplines, including computer science and computational linguistics, in its pursuit to fill the gap between human communication and computer understanding. Every day and everywhere, we are generating tons of data. Using these data, we can find business and analytical insights to boost our personal and professional life. In this task, NLP is used to find insight from the news data. Here, I tried to find the news acceptance by the users depending on their upvote's opinion. Moreover, it can also give us perspicacity into current and future Title acceptance accuracy for news authors.

2 DATA DESCRIPTION

Given dataset is the different news title by different authors of world news from the time of 2008 to 2016. There are upvotes and downvotes count for all news by readers. Overall, the dataset has around 500,000 data rows with 8 columns.

3 IMPLEMENTATION

I have used Google colab as my IDE and PyTorch library for solving the task. I do not have enough processing power device available to handle the given size data. Thereafter, I used Google colab's GPU to train the model. PyTorch is an open-source machine learning library used primarily for applications such as computer vision and natural language processing. It has advantages like easy to learn, higher developer productivity, easy debugging, data parallelism, etc.

3.1 Pre-processing

For preprocessing, basic functionalities of Pandas, NumPy and NLTK have been used. Started with removing the English stopwords from the NLTK library. As I am using the only title and up votes column, I separated these two columns to create separated dataset. Then removed the punctuation, some special characters from the dataset and dropping all the missing value rows. Afterward, using 're' library to remove some other punctuation, new lines, and blank spaces. I have used the average up votes to create the class of acceptance by the users or not. We can even create different classes using different techniques like quantiles to create more classes and train the model depending on our need and use cases. For testing reason, I have also created 80% training data and 20% testing data beforehand.

3.2 Lemmatization

Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. I used lemmatization on titles to make information more precise.

3.3 Field Object

An object of class type 'Field' is defined that will store information about the way of preprocessing the data. There are two kinds of columns are defined: Field and Label.

- Field is used to specify how to preprocess each data column in our dataset.
- LabelField is used only to define the label in classification tasks (title/ news accepted or not).

A text classification model is trained on fixed vocabulary size, all the words that were not included in the vocabulary are known as "Out of Vocabulary" (OOV) words. To handle the OOV, PyTorch supports a feature that replaces the rare words in training data with unknown token. Then, I have connected the created Field objects to the columns in the dataset. Afterward, the both the training and testing data are loaded into Tabular dataset model.

3.4 Build The Vocabulary

In the next step, the vocabulary is created by mapping all the unique words in the training data to an index. By using word embedding, it will map the index to the corresponding word embedding right after. TorchText creates a dictionary of all the unique words and arranges them in a decreasing order in accordance to their frequency. Next, TorchText assigns a unique integer to each word and keeps this mapping in Text.vocab.stoi (string to index) and a reverse mapping in Text.vocab.itos (index to string).

3.5 Iterator

TorchText provides BucketIterator that groups sequences together. It returns a Batch object that contains the data of one batch while the text and labels can be accessed via column names. Beforehand, I have created 20

4 LSTM MODEL

Long Short-Term Memory (LSTM) network is a kind of RNN model that deals with the vanishing gradient problem. It learns to keep the relevant content of the sentence and forget the non-relevant ones based on training. This model preserves gradients over time using dynamic gates that are called memory cells. At each input state, a gate can erase, write and read information from the memory cell. Gate values are computed based on linear combinations of the current input and the previous state. The hidden state acts as the neural network's memory. It holds information on previous data the network has seen before. The operation on the information is controlled by three corresponding gates: Forget gate: Controls which content to keep and which should be forgotten from prior steps. Input Gate: Controls which information from the current step is relevant to add to the next steps. Output Gate: Controls what should be the next hidden state, i.e., the output of the current step.

- **Forget gate:** Controls which content to keep and which should be forgotten from prior steps.
- **Input Gate:** Controls which information from the current step is relevant to add to the next steps.
- **Output Gate:** Controls what should be the next hidden state, i.e., the output of the current step.

4.1 Pack padded sequence or Padpacked sequence Functions

The pack padded sequence is a format that enables the model to ignore the padded elements. LSTM model does not distinguish between padded elements and regular elements but using this function it will not perform gradients calculation for backpropagation step for the padded values. When we feed the model with packed input it becomes dynamic and save unnecessary calculations. The pad packed sequence function is a reversed operation for pack padded sequence and will bring the output back to the familiar format [batch size, sentence length, hidden features].

5 RESULT

After training the model with 5 epochs with 0.00001 learning rate, using Adam as optimizer and Cross Entropy loss as loss function, I have got 90% accuracy on both training and testing dataset. You can find the code here [1]. Codes are commented and naming convention is self-explanatory.

6 POSSIBLE INSIGHTS FROM THE DATASET

- Using time and dates, we can do time series analysis for looking at the trend of the readers depending on their up and down votes.
- Reader and writer analysis are also possible using users' votes and authors' writing.
- The author's performance analysis is possible by users up votes and writing analysis.

7 CONCLUSION

There are a lot of possibilities with this dataset. I feel that, I have created a model for prediction which can perform with good success. Other possible and additional insight can be found here which totally depends on more time and in my case more processing power.

REFERENCES

[1] Md Jonybul Islam. [n.d.]. *Eluvio DS Challenge NLP.* https://github.com/jonybul2153/Eluvio_DS_Challenge