Unsupervised Sentiment Analysis Using Small Recurrent Language Models

Vivekananda Journal of Research
July- Dec 2019, Vol. 8, Issue 2, 171-181
ISSN 2319-8702(Print)
ISSN 2456-7574(Online)
© Vivekananda Institute of Professional Studies
http://www.vips.edu/vjr.php

Tanseer Saji*
Pawan Whig**

Abstract

We explore the possibility of unsupervised byte-level sentiment learning a sentence in the English language using small recurrent language models. Long Short-Term Memory (LSTM) network is a simple and effective network to use while working with sequential data like text or audio. As LSTM processes the data, it learns all the information regarding the given input in the context of all the inputs before that. A. Radford et al [1] provided the evidence that a multiplicative LSTM (mLSTM) [9] is able to learn the concept of sentiment in a manipulable way, but they were able to achieve this result due to huge amount of data samples used for training. This paper tries to investigate the neuron or neurons responsible for sentiment analysis inside a Long Short-Term Memory (LSTM) network when there is a limited amount of training samples available.

Introduction

Sentiment Analysis or Opinion Mining [2] have been a major development of machine learning and artificial intelligence to quantify and extract the subjective information in a given text. Traditionally, sentiment analysis was done using supervised algorithms like support vector machines (SVM), bag-of-words techniques, naive bayes, etc.

In the traditional method of sentiment analysis, there is a huge list of words and their sentiment usually formed from -1 to 1 where -1 represents negative words such as "abort", "fraud", etc. and 1 represent is positive words such as "amaze", "rise", etc and the sentiment value of 0 is assigned to neutral words which include noun and adjective

* Research Engineer: Floss Creatives Ltd, Landon.
E-mail : tanseer.saji@flosscreatives.com
** Dean Research , Vivekananda Institute of Professional Studies , New Delhi.

in a sentence. Stop words which are commonly used words in a language such as “the”, “and”, “a”, etc are removed from the dataset in the training stage and ignored during inference stage. After preprocessing the input text, it is converted to a vector according to a dictionary such as GloVe [4] and is fed to some probabilistic algorithms such as Naive Bayes [5].

Deep learning for sentiment classification [3] helped in learning the complex internal sentiment representations between different words in a sentence. This was also the base motivation for this paper, Using deep learning, sentiment analysis can be done on text dataset that is not pre-annotated with their sentiment saving a lot of time in text preprocessing stage.

Recurrent Neural Networks (RNN) [6] which is a class of artificial neural network where the connections between nodes form a directed graph along a temporal sequence. Unlike Feed-Forward neural network [7], RNNs can use their internal state (memory) to process a sequence of inputs such as a sentence which is a sequence of characters arranged in a specific order.

A standard RNN suffers from problems such as vanishing and exploding gradients.

- The vanishing gradient problem is a difficulty found in neural networks that used gradient-based optimisation techniques such a gradient descent and back propagation. In such methods, each of the neural network’s weights receives an update proportional to the partial derivative of the error function with respect to the current weight in each iteration of training. The problem is that in some cases, the gradient will be vanishingly small, effectively preventing the weight from changing its value. In the worst case, this may completely stop the neural network from further training.
- The exploding gradient problem occurs when error gradients accumulate during an update and result in very large gradients which in turn result in very large updates in network weights and making it difficult for the network to minimize the error. At extreme cases, the error can overflow the memory allocated to floating-point numbers and become NaN (Not a Number) making it impossible for the network to learn.

To prevent these difficulties in RNN, Hochreiter and Schmidhuber [8] introduced LSTM, Long Short Term Memory network. LSTM networks are a type of RNN which include a memory cell that can maintain information in memory for longer periods of time.

A set of gates is used to control when information enters the memory, when it leaves and when it is forgotten.

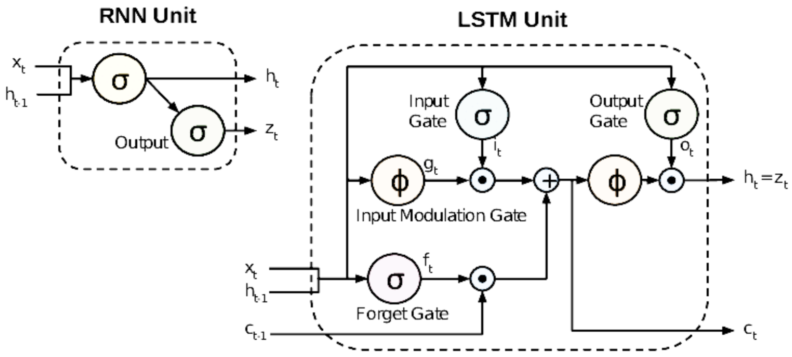

Figure 1:An illustration showing the difference between standard RNN and LSTM, the LSTM network has additional units that prevent it from vanishing and exploding gradient problem.

This paper focuses on a small 2 layer LSTM with 32 neurons each to investigate whether it is able to learn information about the sentiment of a byte-level sequence of a sentence given as input. A. Radford et al [1] provided the evidence that a multiplicative LSTM (mLSTM) [9] is able to learn the concept of sentiment in a manipulable way when trained on 82 million Amazon product reviews. The question is, will a small LSTM model with very few training data, approximately 150000 IMDB Movie reviews can recover such information?

Dataset

The model was first trained on a corpus of text which is not annotated with its sentimental value and then the trained model was applied to a dataset of sentences with its corresponding sentiments.

For the first part publicly available corpus of English to French translation was used and for the second part, a dataset of 150k IMDB movie reviews divided into 15 equal parts of 10k data samples.

Methodology

The training process is split into two parts: part 1 - make a text translator to train the model and make it familiarize to the language. Part 2 - Analyse the hidden state of the translator model by passing IMDB Movie reviews with known sentiment values.

Text Preprocessing

The very first step will be to create a character vocabulary where each unique character is associated with a unique integer id. This vocabulary will be used to represent the text in a three-dimensional one hot vectors [10]. One hot encoding is a process by which categorical variables are converted into a form that could be provided to ML algorithms to do a better job in prediction, the end product of one-hot encoding is called one-hot vector.

```
LET one_hot_vector BE A VECTOR OF ZEROS WITH SHAPE [len_of_samples,
                                                   len_of_line,
                                                   len_of_vocab]

FOR  i  IN  text_samples
   FOR  j  IN  text_samples[i].length
      one_hot_vector[i][j][vocab[text_samples[i][j]] = 1
```

Figure 2: An algorithm to one-hot vectorise the text.

The above algorithm will create a 3D one-hot vector of all the text in the dataset. For simplicity, the dataset only contains lower case alphabets and numbers 0 to 9. This makes the vocabulary consistent for part one and part two of the training process.

Text: \tthe quick fox\n

Vocabulary: {'\t': 0, '\n': 1, ' ': 2, 'c': 3, 'e': 4, 'f': 5, 'h': 6, 'i': 7, 'k': 8, 'o': 9, 'q': 10, 't': 11, 'u': 12, 'x': 13}

One Hot Vector Representation:
```
[[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
  [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
  [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
  [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
  [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]]
```

Figure 3: Each sentence in the dataset is concatenated with a TAB (\t) at the beginning and a NEWLINE (\n) at the end. This is to ensure that the model knows where to start and end the sentence while decoding.

Training Translator

After preprocessing the dataset, the next step in the process is to train then English to French translation model. This step can be replaced by any other technique using which the model's weights can be optimized to work with the English language or any other language depending on the dataset.

For the translator, the LSTMs and arranged in a sequence to sequence encoder-decoder architecture [11]. The encoder model takes the English text data as input and pass that information through an LSTM model and outputs the Hidden State (h) and Hidden Cell Memory (c) [13] which is concatenated and passed along with the French equivalent of the English text as the inputs to the decoder LSTM which then outputs a one-hot representation. The goal is to optimize this output and achieve a fairly accurate translation of English to French.

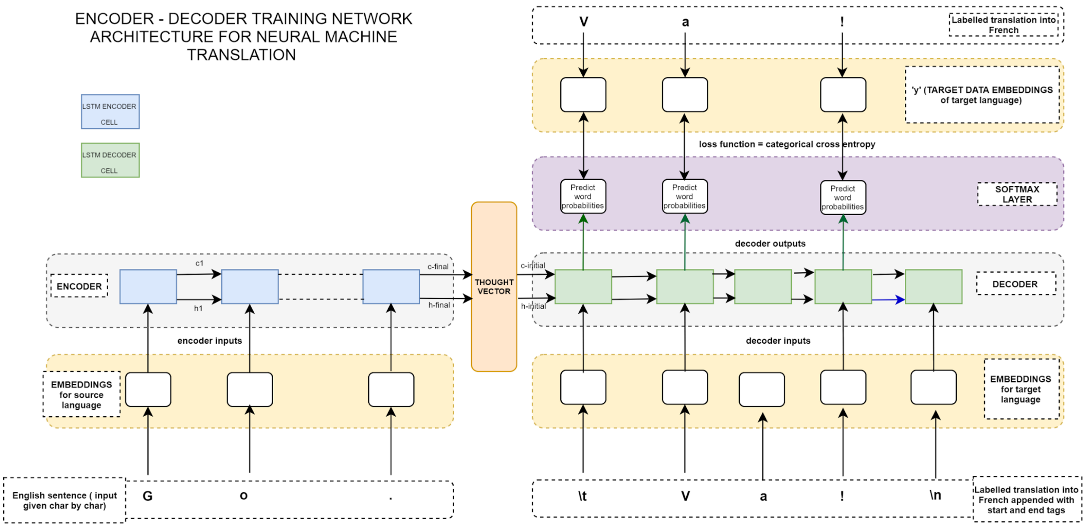
ENCODER - DECODER TRAINING NETWORK ARCHITECTURE FOR NEURAL MACHINE TRANSLATION
LSTM ENCODER CELL
LSTM DECODER CELL
V
a
!
Labelled translation into French
'y' (TARGET DATA EMBEDDINGS of target language)
loss function = categorical cross entropy
Predict word probabilities
SOFTMAX LAYER
decoder outputs
c1
h1
c-final
h-final
THOUGHT VECTOR
c-initial
h-initial
ENCODER
DECODER
encoder inputs
decoder inputs
EMBEDDINGS for source language
EMBEDDINGS for target language
English sentence (input given char by char)
G
o
.
\t
V
a
!
\n
Labelled translation into French appended with start and end tags

Figure 4: Encoder-Decoder training architecture for NMT [12]

Encoder-Decoder Sequence to Sequence network using in this paper takes three input vectors, Encoder Data: which is a character vector of the English text, Decoder Data: which is a character vector of the same English text shifted by one character and Decoder

Target Data: which is the character vector of the corresponding French text. All the inputs to the encoder-decoder network are three-dimensional one-hot representations of the text as explained in the previous section.

Encoder

The encoder network (used in this paper) consists of an LSTM network with 32 hidden units each of these hidden units has a hidden state which is a floating-point number.

iteration the LSTM units currently at called time steps.

Then at any time step , the hidden state of each hidden unit can be calculated as a function of the previous time step and current input vector given by this equation.

Where is some non-linear function such as

and are the hidden parameters that will be learned to fit the data in the most accurate way.

Decoder

Encoder's hidden state after the final time step is called an Encoder Vector. This vector aims to encapsulate the information for all the input elements in order to help the decoder to make accurate predictions. This vector also acts as initial hidden state for the decoder LSTM.

The decoder network LSTM will function exactly as the encoder LSTM, the only difference is after the final time step the hidden unit vector is passed through an activation function which will convert the elements in the hidden unit elements to desirable output that can be decoded using the vocabulary created during text preprocessing. In most cases, or function is used for this task.

In simple terms, argmax function returns the element in the domain where the corresponding range is maxima in a function. Or in case of lists and vectors, it returns the position of the maximum element.

In simple terms, the SoftMax function takes any vector, and converts it into a vector of probabilities range and .

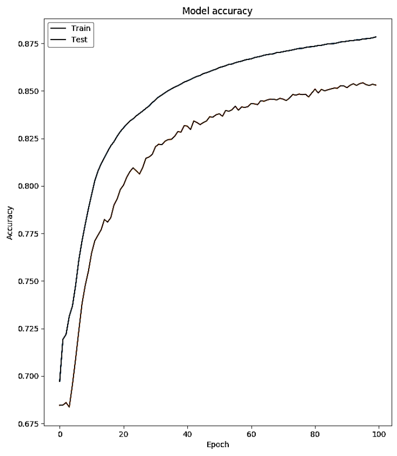

Figure 5: Training and Validation accuracy of the model over each iteration.

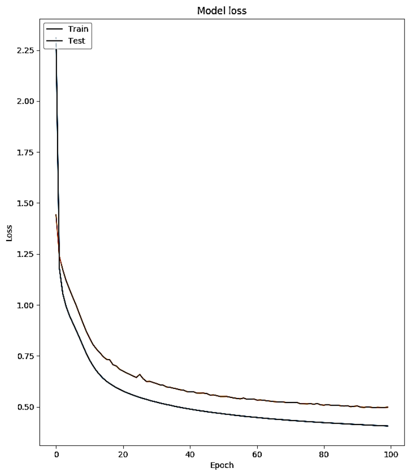

Figure 6: Training and Validation Loss of the model over each iteration.

After training for around 100 iterations, the model error started to converge towards 0.35 and a validation accuracy of 86%.

Testing with IMDB Movie Review

For the second part of the process, IMDB Movie Review dataset is used as a base sentiment of the text. Then each data sample is preprocessed using the same algorithms as translation data was preprocessed, this step is important to make a common vocabulary for both English translation data and IMDB movie review. If there is not a common vocabulary for both datasets then that could cause some runtime exceptions while running the translation model, i.e., the model will not work properly. After the preprocessing step, the dataset is divided into two sub-datasets. One dataset contains all the positive reviews or data samples and the other one contains all the negative data sample.

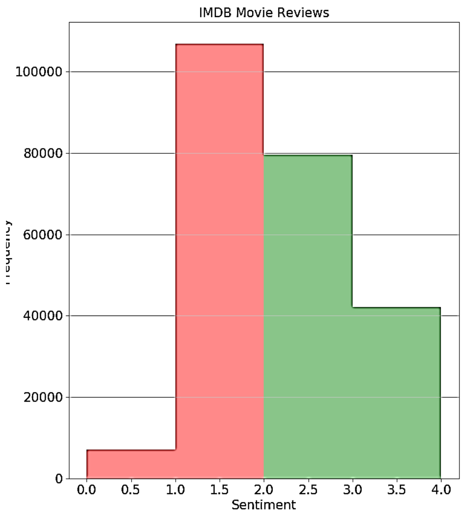

Figure 7: Distribution of positive and negative texts in IMDB Movie review dataset. A text is positive (shown in green) if its sentiment value is greater than or equal to 2 otherwise the text is negative (shown in red).

For this section, only the encoder network is loaded and the decoder network is discarded. This is done because of the assumption that the hidden state or the hidden vector of the LSTM learns information about the sentiment of a text, so there is no need for a decoder network to be loaded.

One data sample is taken from each positive and negative dataset, preprocessed to one-hot vector representation and passed to the encoder network to get the encoder vectors.

Results

It is found that unlike the results published A. Radford et al [1] our model which is a standard layer of LSTM cell trained on 50k data samples as compared to his mLSTM trained on 83M data samples, was representing emotions by an amplified hidden vector. Mathematically, the absolute median of all elements in the hidden vector of positive text is on an average larger than absolute median of hidden vector of a negative text.

After repeating this process for all 150k data samples available in IMDB Movie reviews dataset, this hypothesis holds for around 89% of texts.

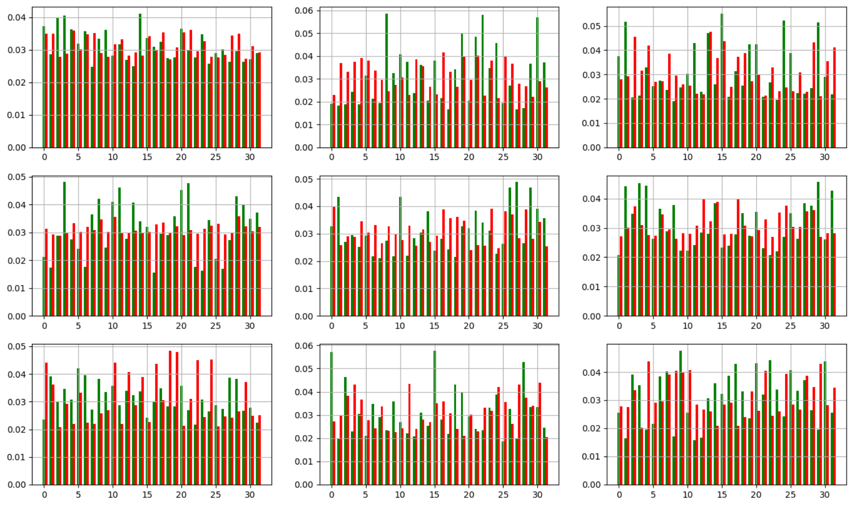

Figure 8: Hidden Vector values of 9 randomly selected data samples. The red bar represents the elements of negative hidden vector and green bar represents the elements of positive hidden vectors.

As shown in fig. 8, most of the elements of hidden vector of texts in positive dataset larger than the elements of hidden vector of texts in negative dataset.

Conclusion and Future Scope

The paper provides an evidence that small recurrent networks such a single layer LSTM can learn some information about sentiment of a given text even if there is a limited amount of data samples.

Using this information, it is possible to make a sentiment classification algorithm without the need of Bag of words and other lexicon-based methods. An application of such information is that a text generation system such as a chatbot can adjust the sentiment of the generated text based on the sentiment of its conversation with a human.

References

Radford, Alec, Rafal Jozefowicz, and Ilya Sutskever. “Learning to generate reviews and discovering sentiment”. arXiv preprint arXiv:1704.01444 (2017).

Pang, Bo, and Lillian Lee. “Opinion mining and sentiment analysis”. Foundations and Trends® in Information Retrieval 2.1–2 (2008): 1-135.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio”.Domain adaptation for large-scale sentiment classification: A deep learning approach.” Proceedings of the 28th international conference on machine learning (ICML-11). 2011.

Pennington, Jeffrey, Richard Socher, and Christopher Manning. “Glove: Global vectors for word representation”. Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). 2014.

McCallum, Andrew, and Kamal Nigam. “A comparison of event models for naive Bayes text classification”. AAAI-98 Workshop on learning for text categorization. Vol. 752. No. 1. 1998.

Mikolov, Tomáš, et al “.Recurrent neural network-based language model.” The eleventh annual conference of the international speech communication association. 2010.

Svozil, Daniel, Vladimir Kvasnicka, and Jiri Pospichal. “Introduction to multi-layer feed-forward neural networks”. Chemometrics and intelligent laboratory systems 39.1 (1997): 43-62.

Hochreiter, Sepp, and Jürgen Schmidhuber. “LSTM can solve hard long time lag problems”. Advances in neural information processing systems. 1997.

Krause, Ben, et al. “Multiplicative LSTM for sequence modelling”. arXiv preprint arXiv:1609.07959 (2016).

Wikipedia contributors. “One-hot”. Wikipedia, The Free Encyclopedia. Wikipedia, The Free Encyclopedia, 9 Sep. 2019.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio “.Neural machine translation by jointly learning to align and translate.” arXiv preprint arXiv:1409.0473 (2014).

Kompella, Ravindra. “Neural Machine Translation - Using seq2seq with Keras”. Medium, Towards Data Science, 14 Sept. 2018, https://towardsdatascience.com/neural-machine-translation-using-seq2seq-with-keras-c23540453c74.

Christopher. “Understanding LSTM Networks”. Understanding LSTM Networks -- Colah’s Blog, 27 Aug. 2015, https://colah.github.io/posts/2015-08-Understanding-LSTMs/.