2017

Formal Languages and Computability CMPT440

Extracting Learned States From An LSTM

PROJECT WRITEUP

MICHAEL GUARINO

Abstract:

We purpose a method for extracting states from a Recurrent Neural Network architecture, Long Short Term Memory, to assemble rules that govern transitions between states in a Deterministic Finite Automata. The Long Short Term Memory (LSTM) deep learning architecture performs a language modeling task for word-level prediction. The LSTM is regularized using dropout between the layers not between the recurrent connections and experimentation has been done to determine the optimal number of layers and recurrent connections.[4] Methods to extract state transitions from the LSTM Recurrent Neural Network were experimented with using the Recurrent Neural Network's learned weights to understand how the network encodes state transitions with network input.

Introduction:

The Recurrent Neural Network architecture is quite popular for language modeling tasks. The LSTM variant of the Recurrent Neural Network family of deep learning architectures is specifically used for capturing long term dependencies within sequences.[3] Learning formal grammars via the Recurrent Neural Network architecture for the purpose of extracting the learned states have been explored by Giles 1992[1], Firoiu 1996[2], and Vahed 2004[5]; however, the LSTM architecture is not known to be used for this task. Once states are extracted from the LSTM architecture a Deterministic Finite Automata can be constructed using the extracted states that describe the transitions between states in the Deterministic Finite Automata. Methods for LSTM state extraction has been experimented with using a state extraction process where LSTM states are clustered together using dimensionality reduction then states are clustered together based on a similarity metric and are then used to construct state transitions on the constructed Deterministic Finite Automata. These methods; however, are prohibitively computationally expensive and were not found to be effective.

System Description:

The LSTM Recurrent Neural Network will perform a language modeling task on a corpus dataset of text documents doing word-level prediction. The LSTM will learn states that allow it to properly perform word-level prediction task on the corpus.

The Recurrent Neural Network architecture that was used was heavily based on the LSTM architecture of Zaremba in "Recurrent Neural Network Regularization.[4] A two layer LSTM with 30 recurrent connections (steps) for every LSTM memory cell. The LSTM memory cell using a gating mechanism to update/forget 'memory' states as depicted in figure 1-1. The hidden states of the LSTM were initialized to zero. Several sizes were experimented with for the LSTM layers to determine the most effective architecture. We use a medium LSTM architecture with approximately 2250 units per layer were used which is the approximate size of the unique words in the text corpus. All parameters initialized uniformly between -0.5 and 0.5 with dropout applied to 20% of non-recurrent connections. A larger LSTM architecture was experimented with that had approximately 5000 units per layer all parameters initialized uniformly between -0.5 and 0.5 with dropout applied to 20% of non-recurrent connections; however, no performance benefit was seen for the extra time spent training the LSTM model. With both of these architectures the gradients were normalized and clipped using Tensorflow's style of truncated backpropagation on the backpropagation through time algorithm, which has been found to an effective method of backpropagation error through timesteps in sequence based models [6].

Once the LSTM Recurrent Neural Network has successfully learned the dataset a process to extract network states will proceed. Several authors have briefly eluded to how this is done Giles 1992[1], Firoiu 1996[2], and Vahed 2004[5]; however a concrete method has not yet been established. The original hope to extract network states by performing a dimensionality reduction such as principle component analysis on the network state then clustering them using some unsupervised machine learning algorithm to group related states together based on how they respond to input. Due to the size of the LSTM Recurrent Neural Network architecture the original concept of determining state transitions to construct Deterministic Finite Automata using the learned weights from the trained LSTM Recurrent Neural Network architecture turned out to be prohibitively computationally expensive. Figure 1-2 shows the learned weights that a Recurrent Neural Network cell contains learning more from the sequence as the next time step is learned by the network. An alternative method that we are purposing is to pass every combination of sequence found in the text corpus into the network outputting a probability of the next word in the sequence therefore constructing a three dimensional probabilistic state transition table where for every sequence element found in the corpus dataset contains a predicted next word to transition to for every given time step based on a probability. As input is fed through the network the highest probability word is selected for every time step as seen in Figure 1-3.

In this way the learned states may then be extracted from the LSTM Recurrent Neural Network and be used to create state transitions between states in a Deterministic Finite Automata in a computationally effective manner. The constructed Deterministic Finite Automata can then accept a text input sequence to determine if the text input sequence was in the original corpus. The DFA essentially acts to validate that the text input sequence exists within the original dataset corpus which confirms that the LSTM state extraction process has been successful therefore valid input to the DFA is an text input sequence that exists in the dataset corpus.

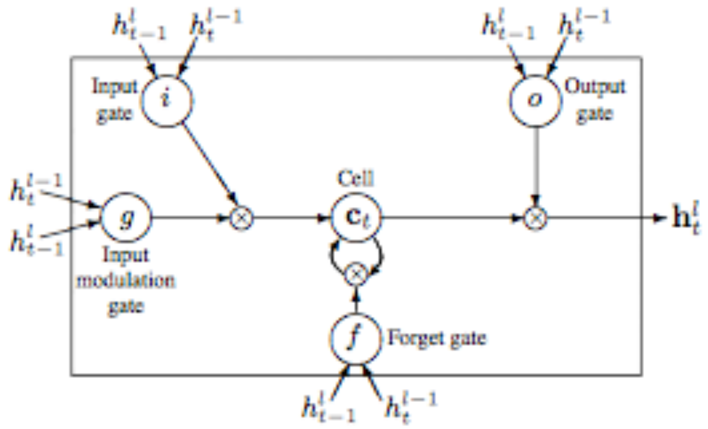

Figure 1-1 LSTM memory cell[4]

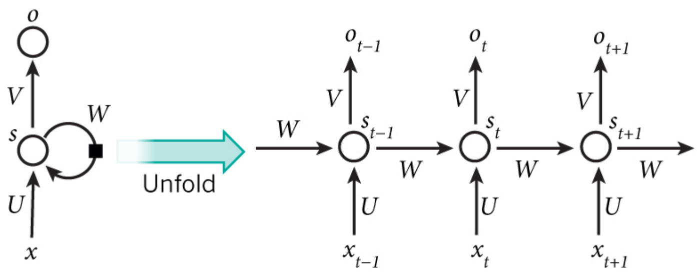

Figure 1-2 Recurrent Neural Network Cell unrolled[7]

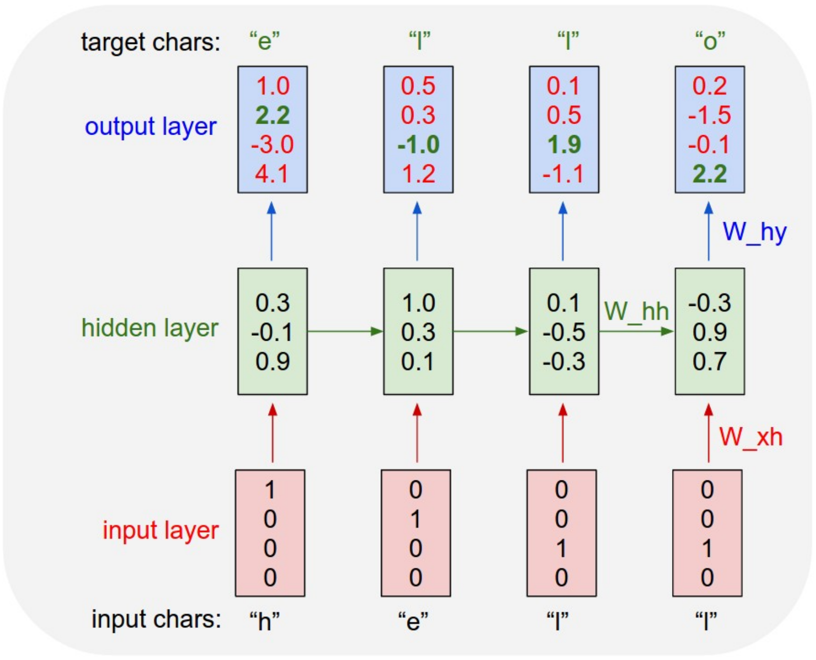

Figure 1-3 Recurrent Neural Network Cell unrolled[8]

Requirements:

In order to train the LSTM Recurrent Neural Network we used a small dataset of politician speeches from the Democratic and Republican national conventions of the 2016 presidential election, which we determined to be small enough to train quickly on our computing resources but large enough to serve as an actual test to the method purposed in this document. The LSTM model was trained on a machine containing Nvidia 1080TI GPU and Intel 6800K CPU. The model was built in Keras using a Tensorflow back end to speed up development time and computation.

Literature Survey:

There has been some existing interest in using Recurrent Neural Network architectures to construct Deterministic Finite Automata in the area of natural language processing. Giles and Chen constructed a Recurrent Neural Network to learning an unknown grammar and used a dynamic clustering algorithm to extract the states learned by the Recurrent Neural Network in order to construct a Deterministic Finite Automata.[1] Firoiu, Oates, and Cohen also performed a similar task using a Elman Recurrent Neural Network to perform a word-level prediction task on a text dataset and used a state extraction process to extract the learned states of the neural network to construct a Deterministic Finite Automata.[2] There has not been any research done in this area using a Long Short Term Memory Recurrent Neural Network architecture. The Long Short Term Memory architecture has become very popular for problems involving learning long term dependencies within sequences with variants showing significant improvements in the area of speech recognition, polyphonic music modeling, and acoustic modeling.[3] These models area extremely powerful in their ability to learn time series/sequential data.

User Manual:

This system will validate that text input exists in the dataset corpus and given an input string in the text corpus will predict the next word in the sequence based on a probability. The user can validate text input by calling the main.py file and passing it the command line argument 'testing'. The user will then be prompted to enter a text string and the system will predict the next word in the string based on the learned states of the LSTM Recurrent Neural Network. The system can also be used to learn sequences in a new dataset by dropping new text data into the data directory and calling the main.py file passing the command line argument 'training' to the system.

Conclusion:

This project explored extracting learned state transitions from a LSTM Recurrent Neural Network based on the work of Giles[1] and Firoiu[2]. The methods proposed in these works; however, were found to be prohibitively computationally expensive as state extractions were performed based extraction of states encoded in the learned variables of a Recurrent Neural Network architecture at every time step in the sequence for every Recurrent cell in the Recurrent Neural Network. Another method considered in this document is an alternative method that we are purposing is to pass every combination of sequence found in the text corpus into the network therefore outputting a probability of the next word in the sequence. The resulting output could then be used to construct a three dimensional probabilistic state transition table where for every sequence element found in the corpus dataset contains a predicted next word to transition to for every given time step based on a probability.

References/Bibliography:

1. Giles, C. I., C. B. Miller, D. Chen, G. Z. Sun, H. H. Chen, and Y. C. Lee. "Extracting and learning an unknown grammar with recurrent neural networks." *Neural Information Processing Systems* (1992): n. pag. Web.
2. Firoiu, Laura, Tim Oates, and Paul R. Cohen. "Learning deterministic finite automata with a recurrent neural network." (1996): n. pag. Web.
3. Greff, Klaus, Rupesh K. Srivastava, Jan Koutnik, Bas R. Steunebrink, and Jurgen Schmidhuber. "LSTM: A Search Space Odyssey." *IEEE Transactions on Neural Networks and Learning Systems* (2016): 1-11. Web.
4. Zaremba, Wojciech, Ilya Sutskever, and Oriol Vinyals. "Recurrent Neural Network Regularization." *International Conference on Learning Representations* (2015): n. pag. Web.

5. Vahed, A., and C. W. Omlin. "A Machine Learning Method for Extracting Symbolic Knowledge from Recurrent Neural Networks." *Neural Computation* 16.1 (2004): 59-71. Web.
6. Sutskever, Ilya. *Training Recurrent Neural Networks*. Thesis. Thesis / Dissertation ETD, 2013. N.p.: Graduate Department of Computer Science U of Toronto, n.d. Print. http://www.cs.utoronto.ca/~ilya/pubs/ilya_sutskever_phd_thesis.pdf.
7. "Understanding LSTM Networks." *Understanding LSTM Networks -- colah's blog*. N.p., n.d. Web. 07 May 2017. http://colah.github.io/posts/2015-08-Understanding-LSTMs/
8. *The Unreasonable Effectiveness of Recurrent Neural Networks*. N.p., n.d. Web. 07 May 2017. http://karpathy.github.io/2015/05/21/rnn-effectiveness/.