Autonomous Detection of Regular Languages

Joel Sommer, Lingjing Huang

# Objective:

The objective of this project is to determine whether a neural network trained on strings that are in a regular language is able to determine if a string is in said language. Ideally, the neural network would be capable of analyzing an arbitrarily long string.

# Performance Requirements:

In general, performance will be evaluated by using the Receiver Operating Characteristics ROC curve of the neural network’s after being exposed to varying quantities and types of training data and test data. Key metrics to collect will be

# Background:

While, by definition, a regular language does not require any sort of memory construct such as a stack or queue to process, it is believed that a Recurrent Neural Network (RNN) will be required to adequately determine if a string is in the language. This is because the network will be required to learn the states and transition criteria of the regular language, that is, the network will have to have some awareness of the states that came before.

# Source Software:

1. Libraries: TensorFlow plays main role to develop the project. TensorFlow is an open source software library for numerical computation using data flow graphs.
2. Programming Language: Python. Python was the first client language supported by TensorFlow and currently supports the most features.

# Experimental Plan:

1. The project plan is to have at least 2 parts, train a language model on input dataset and assess performance of the model against a test dataset.
2. Use LSTM Networks Model. Long Short-Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies.
3. Data input: At least 1000 valid strings and 6000 invalid strings for the training.

# Language Model

The LSTM model is composed of several cells that are interlinked as depicted in Figure 1. The current plan is to mimic the cell depicted in Figure 2.

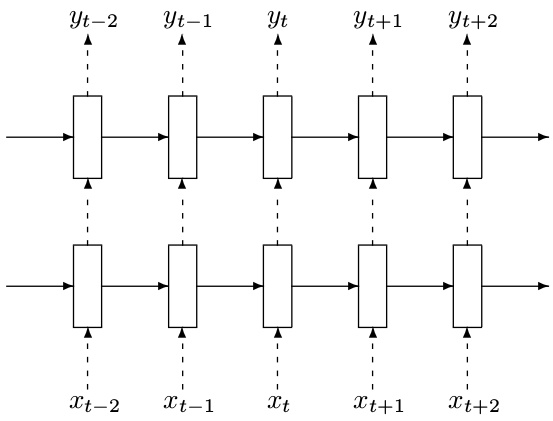


Figure 1: The language models – whole view of the LSTM model [1]

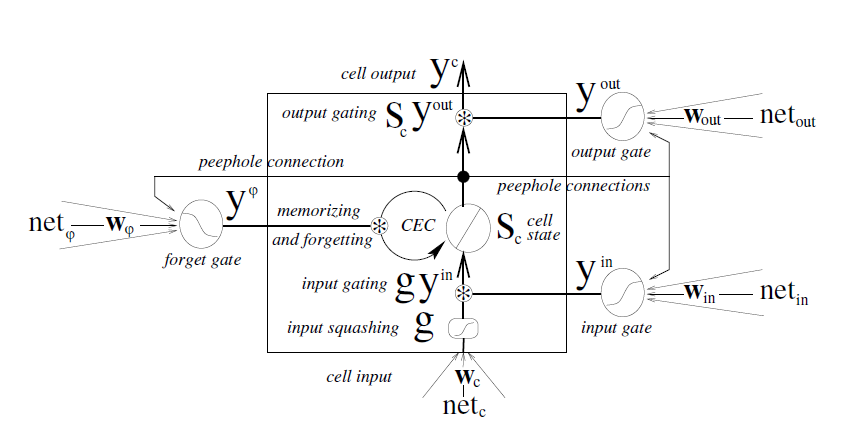


Figure 2: The language models – one cell of the LSTM model [2]

The design of the LSTM cell is borrowed from the work done in [2]. The authors of [2] learned that it was important to have “forget gates” or order to provide a mechanism to rest a given cell’s state and improve overall performance of the network. We are using LSTM with forget gate and recently introduced peephole connections.

Going back to the cell in Figure 2, the basic unit of the LSTM network is the memory block containing one or more memory cells and three adaptive, multiplicative gating units shared by all cells in the block.

For the input

1. Input gate activation
2. Forget gate activation
3. Cell input and cell state

Output gate activation and cell out

This approach differs from the approach described in [3] and [4] in that it allows for the ability to perform something akin to a reset. This is important because the approaches described in [3] and [4] are, more or less, to sample all of the data in given data set and predict the next word that may be from the context of all that has been before. The concern is that with this probabilistic style may make sense from a signal processing perspective, but does not necessarily apply from a state machine perspective. The rules for the two are not necessarily congruous.

# Datasets:

Multiple regular languages will be assessed during this experiment. For all of the languages, the same training and testing data sets will be used. This will allow for more reuse of the data sets.

The training sets have been developed based on DFAs that have been selected for the project. For each DFA, 1000 strings have been constructed that are a minimum of 1000 characters in length. The all generated strings (7000 in total) were evaluated against all 7 DFAs to ensure that they are only accepted by the DFA for which they were built.

The data set is stored in a CSV file that where the first column is the name of the DFA for which the second column is in the language.

图片包含 剪贴画

已生成高可信度的说明

Figure 3: DFA 0

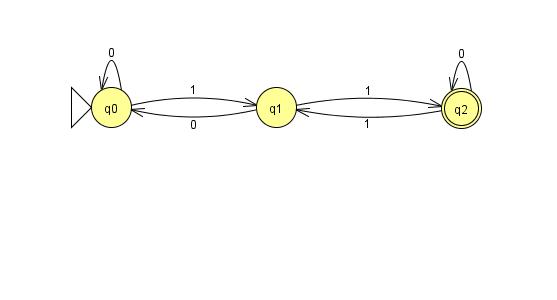


Figure 4: DFA 1

图片包含 剪贴画

已生成高可信度的说明

Figure 5: DFA 2

图片包含 剪贴画

已生成高可信度的说明

Figure 6: DFA 3

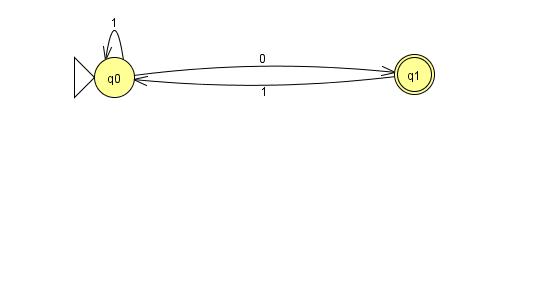


Figure 7: DFA 4

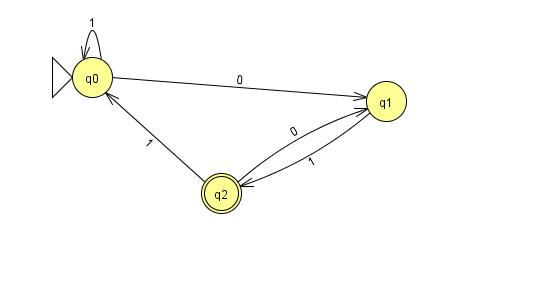


Figure 8: DFA 5

图片包含 物体, 时钟, 手表

已生成高可信度的说明

Figure 9: DFA 6

# Proof of Concept:

The initial proof of concept will be to develop on form of RNN that can be trained to operate on the data set. If time permits, additional RNNs may be developed to evaluate the performance of different RNN types against the data set. After performing an initial proof of concept, the overall project plan may be altered to improve the direction of the project.

# Final Deliverables:

The final deliverables are:

1. The final report – a document, which will summarize our project’s idea, objective, progress, accomplishments and results.
2. Raw data – all samples (images) we generated and/or utilized in the project.
3. Presentation slides ­– brief overview of our project for the final in class presentation.
4. Webpages – contain all information about the project.

# References

|  |  |
| --- | --- |
| [1] | "language model - extension of RNN," 8 Sep 2017. [Online]. Available: http://www.cnblogs.com/xinbaby829/p/7127451.html. |
| [2] | F. A. Gers and E. Schmidhuber, "LSTM recurrent networks learn simple context-free and context-sensitive languages," *IEEE Transactions on Neural Networks,* vol. 12, no. 6, pp. 1333-1340, Nov 2001. |
| [3] | R. Atienza, "LSTM by Example using TensorFlow," 17 Mar 2017. [Online]. Available: https://towardsdatascience.com/lstm-by-example-using-tensorflow-feb0c1968537. |
| [4] | "Recurrent Neural Networks," 27 Jan 2018. [Online]. Available: https://www.tensorflow.org/tutorials/recurrent. |
| [5] | C. Olah, "Understanding LSTM Networks," 27 Aug 2015. [Online]. Available: http://colah.github.io/posts/2015-08-Understanding-LSTMs/. |