Autonomous Detection of Regular Languages

Joel Sommer, Lingjing Huang

# Objective:

The objective of this project is to determine whether or not 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 30 valid strings and 30 invalid strings for the training. And then around 100+ entries for the test set.

# 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 set will contain x valid and y invalid sample string. Each test set will contain t valid and u invalid strings.

# 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.

# References:

[1] Tutorial of the Recurrent Neural Networks. <https://www.tensorflow.org/tutorials/recurrent>

[2] Understanding LSTM Networks. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>