Decrypting Cipher text using the DFA extracted by RNN and Lstar Algorithm

Group Number 26:

Amruta Katke

Anirudha Nilegoankar

Austin Mathew

Jiangfeng Pian

Piyusha Jaisinghani

Rohit Nikam

Sai Vinherkar

Objective

- Using a Deterministic Finite Automata to decipher, message of a subset of English language cyphered by non-statistical models

- Train an Long-Short Term Memory model to comprehend English grammer

- Extract a DFA from this LSTM.

- To extract the DFA we are using L* Algorithm and RNN which is based on the paper “Extracting Automata from Recurrent Neural Networks using Queries and Counterexamples”.

State of the Art

1. The Cipher is attacked in the present using frequency analysis.

2. Extracting DFA from RNN by defining a finite partitioning of the real-valued RNN state space. Exploring the network transitions in the partitioned space to done by using techniques such as BFS exploration and other transition-sampling approaches.

3. State space partitioning is done to extract DFA, it divide each dimension into q equal intervals with q being the quantization level.

4. Using unsupervised classifier such as k-mean to a large sample set of reachable network states.

Limitation of current Practice

1. Frequency analysis relies on the fact that in any language, one letter in a language is more frequently used than the other. Example Z is less frequent than A.

2. To attack a ciphered text with frequency analysis a long message is required so that we can get accurate frequencies of letter. And hence would not work for short cipher texts.

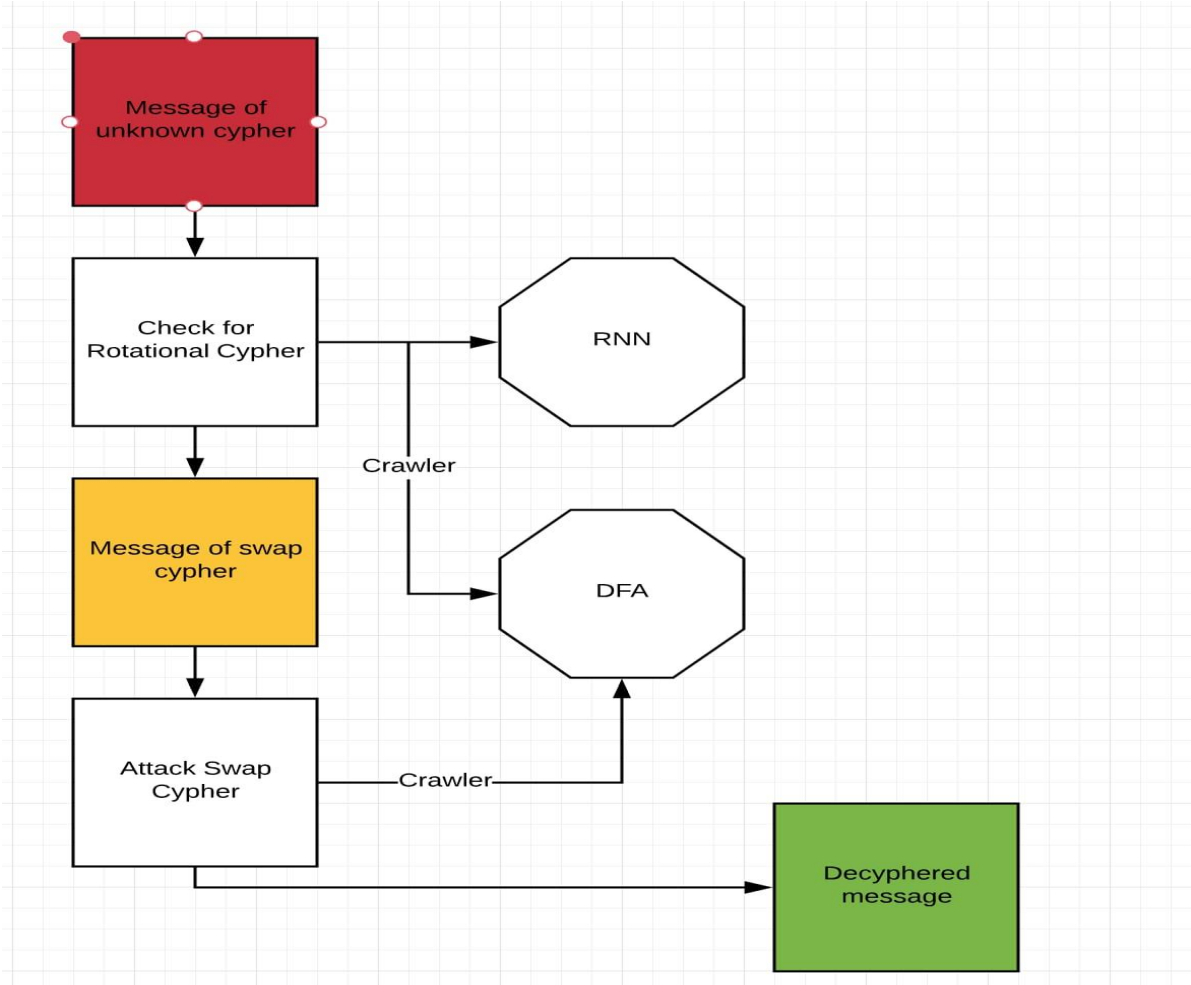
3. In Frequency analysis subject to matter of the text is not considered, for e.g. if some article is related to Wonder Woman then the letter W is more frequent in comparison to other letters.

4. The state space partitioning approach to extract the DFA suffers from inherent state space explosion.

5. Previous approaches face the challenge of choosing the best parameter value for extraction because the partitioning is set before the extraction begins .

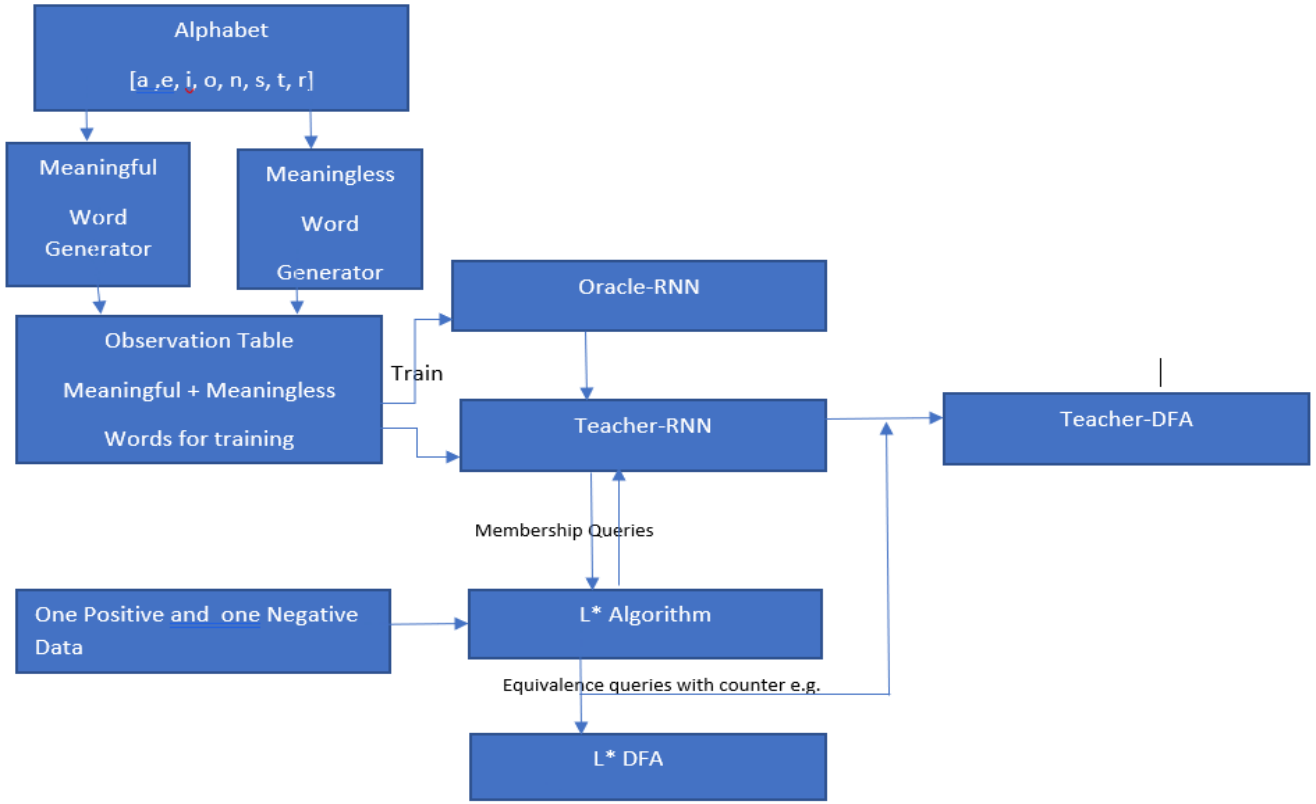
Our Approach

We are using DFA to decipher the cyphered text.



Our Approach

- The DFA used to decipher the texts is extracted using L* Algorithm and RNN which is based on the paper “Extracting. Automata from Recurrent Neural Networks using Queries and Counterexamples”.



Applications

1. Many variants of cyphers are used by gorilla warriors and insurgents of multiple 3rd world countries to communicate among each other.

2. Cyphers are used to encode the messages sent over http protocols.

3. Application software's using Apache, Tomcat or Microsoft iis as web servers always need a technique to discover weak ciphers.

Risks

1. RNN trained over English alphabet subset often accepts non English words. To make it a more robust teacher for L* we tuned the parameter of RNN to num layers = 3 and hidden layers =50. These hyper-parameters were found to allow the RNN to train quickly and to a high accuracy.

2. Due to the polynomial complexity of L* Algorithm extracting DFA for a complex network like words of English Alphabet set containing more than 8 unique letters takes longer and large DFAs are extracted. We are limiting the time of extraction to 60 seconds and hence the DFAs extracted for an alphabet set greater than 8 have missing edges. We would expect that large DFAs would be missing states as well, however this was not verifiable.

3. The RNN is trained over a word set of 1032 which is not enough data for the RNN to generalize over the language and our method tries to find adversarial inputs and in result builds a DFA that is large yet unrefined

Mid term outcome results

1. We trained RNN over eights letter alphabet.
2. For 200 nodes the size of the DFA dictionary is 16KB.

Training RNN for one minutes results approximately 200 nodes of DFA	
Accuracy of RNN	0.2833
Precision of RNN	1.0
Hit Rate of RNN	1.0
Runtime of RNN	14.060
Accuracy of DFA	0.0255
Precision of DFA	0,6320
Hit Rate of DFA	0.092
Runtime of DFA	0.1435
Accuracy of Brute Force	1.0
Hit Rate of Brute Force	1.0
Runtime of Brute Force	0.0727

Contd.

For 500 nodes the size of the DFA dictionary is 16KB.

Training RNN for three minutes results approximately 500 nodes of DFA	
Accuracy of RNN	0.2675
Precision of RNN	0.9965
Hit Rate of RNN	0.983
Runtime of RNN	26.147
Accuracy of DFA	0.0675
Precision of DFA	0.775
Hit Rate of DFA	0.265
Runtime of DFA	0.3124
Accuracy of Brute Force	1.0
Hit Rate of Brute Force	1.0
Runtime of Brute Force	0.0957

Contd.

For 100 nodes the size of the DFA dictionary is 5KB.

Training RNN for fifteen seconds results approximately 100 nodes of DFA	
Accuracy of RNN	0.2635
Precision of RNN	0.998
Hit Rate of RNN	0.99
Runtime of RNN	6.700
Accuracy of DFA	0.0556
Precision of DFA	0.0
Hit Rate of DFA	0.2
Runtime of DFA	0.0468
Accuracy of Brute Force	1.0
Hit Rate of Brute Force	1.0
Runtime of Brute Force	0.0957

Final outcomes

- As the data suggests, validation information with a DFA is magnitudes faster than when using an RNN. However we were not able to extract a reasonably accurate DFA

- While the DFA does offer quick validation time, the comparison method shown in the BruteForce section did often result in a time about half that of the DFA

- The DFA does however offer one advantage. The file size of the DFA is very small compared to that of the dictionary required for comparisons. We did calculate that a DFA of 100% accuracy would required about 800 kb, which is about half the size of the dictionary we used.

Proposed solution for swapped-cypher cracking in the event that a 100% accurate DFA could be extracted.