Ass 3 – Part 2

Dorin Keshales – 313298424
User name: keshald

Model's Parameters (for all tasks)

- Number of epochs: 13
- Activation function: ReLU
- Optimizer: Adam
- Learning rate: 1e-3
- Embedding dim: 30
- Lstm output dim: 30
- Size of the hidden layer: 25
- Train set size: 5000
- Test set size: 750

Palindrome language

The alphabet is ∑ = { 0-9, a-z, A-Z }

The language is **L = { $W\sigma W^R$ | W ∈ ∑*, |W| ≤ 50, σ ∈ ∑ or σ = ε }**
** Each sequence (in L or not) is up to 101 characters long.

- I've chosen the palindrome language because it's known as a non-regular language, which means that a Deterministic finite automaton cannot accept it. It's context-free language therefore it can be accepted by a pushdown automaton(stack). Since our LSTM is unidirectional (as opposed to a bi-LSTM) it lacks the ability to consider both past and future for each token in the sequence, which could help it compare between the start and the end of the sequence.

- I trained the model for 13 epochs when after each epoch I checked the accuracy on the test set. Along the epochs, the model improved it's accuracy on the training set when on the 13'th epoch the accuracy on the training set got to 94.42% accuracy. However, the model did not generalize

well to the test - the accuracy ranged from 48.4% to 50.4% when there was no consistent upward trend (ups and downs).

Duplicate Word language

The alphabet is ∑ = { 0-9, a-z, A-Z }

The language is L = **{ WW | W ∈ ∑*, |W| ≤ 50 }**
** Each sequence (in L or not) is up to 100 characters long.

- I've chosen that language because it's also a non-regular language. In a matter of fact, it's a Context-sensitive language therefore it can only be accepted by Turing machine. I thought that the language will be hard to distinguish because it will require the LSTM to pay attention when the first appearance of the word ends, and that the rest of the sequence is a second appearance of the same word it already saw at the first half of the sequence. It's a hard task to the LSTM because it doesn't know where or when the original word ends(first appearance).

- I trained the model for 13 epochs when after each epoch I checked the accuracy on the test set. Along the epochs, the model improved it's accuracy on the training set when on the 13'th epoch the accuracy on the training set got to 93.12% accuracy. However, the model did not generalize well to the test - the accuracy ranged from 48.13% to 53.6% when there was no consistent upward trend (ups and downs).

Cross language

The alphabet is ∑ = { a-z }

The language is **L = {** $\alpha^n \beta^m \gamma^n \delta^m$ **|** $\alpha, \beta, \gamma, \delta \in$ **∑ , n,m≤ 25 }**
** Each sequence (in L or not) is up to 100 characters long.

- I've chosen another non regular and Context-sensitive language which can only be accepted by Turing machine. I thought that the language will be hard to distinguish because it will require the LSTM to pay attention to the

length of each of the first two sub sequences - α,β , in order to check that the next two sub sequences of γ, δ are with compatible length [len(α) = len(γ) ∧ len(β) = len(δ)]. It's seems to be a hard task to the LSTM .

- I trained the model for 13 epochs when after each epoch I checked the accuracy on the test set. The model failed on both training set and test set. On the training set, the accuracy ranged from 50.78% to 56.02% and on the test set the accuracy ranged from 46.26% to 51.2%. On both, there were no consistent upward trend (ups and downs).

Following the results, The LSTM seems to have failed in this task.