Report 2:

Shilo Avital 206487407 Tamar Michelson 323805861

RNN Acceptor Challenge Report - Part 2

1. Overview

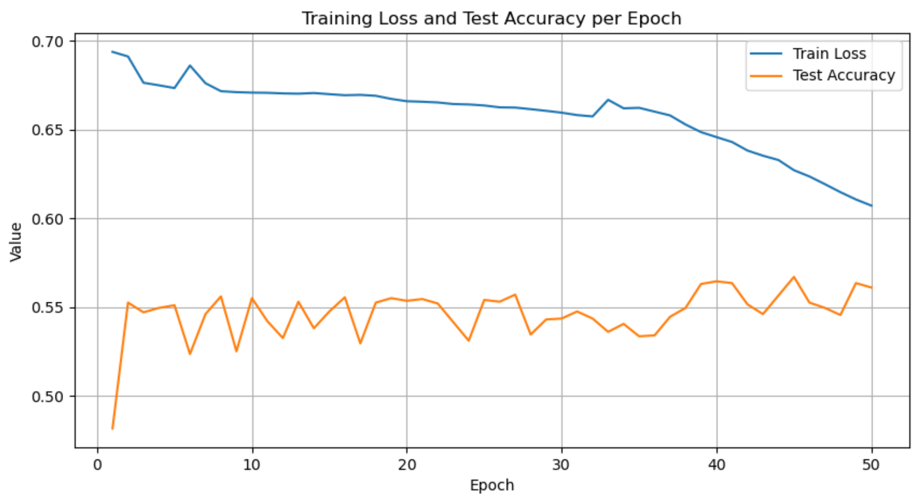
This report describes three languages (8–10) that challenge the RNN acceptor's ability to distinguish between valid and invalid sequences. Each case defines a language and its complement, aiming to find weaknesses in the LSTM-based model.

2. Language 8

Language Description:
Each string begins with an integer value, followed by a random sequence of lowercase letters. The prefix integer equals the ASCII value of the most frequent character in the string plus its frequency count.

Challenge Rationale:
This task requires counting occurrences and computing a non-linear transformation. It involves non-local dependencies and basic arithmetic, which RNNs struggle with.

Experiment Outcome:
The model failed to learn this mapping correctly, showing low accuracy on both train and test sets. The model was trained on 5,000 positive and 5,000 negative examples, split 80/20 into training and test sets, over 50 epochs with a batch size of 64. Unlike languages 9 and 10, the training loss for Language 8 remained relatively high throughout the training process, indicating difficulty in fitting the training data. Correspondingly, the test accuracy stayed near random chance (~0.5), showing that the model neither learned nor generalized effectively. This poor performance is due to the complexity of Language 8, which requires the model to count character frequencies and perform arithmetic operations involving ASCII values—tasks that involve non-local, numeric reasoning beyond the typical capabilities of standard LSTMs.

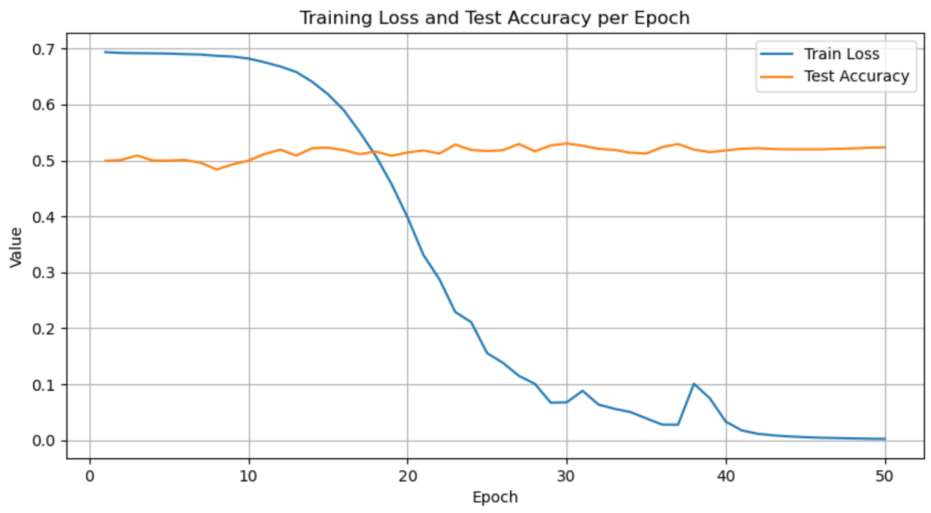


3. Language 9

Language Description:
Strings that begin and end with the same character (e.g., 'abca', 'zxxxz').

Challenge Rationale:
The dependency is between distant characters (first and last), which LSTMs find difficult to capture.

Experiment Outcome:
The model did not generalize well, performing close to chance level. Accuracy plateaued early. The model was trained on a dataset consisting of 5,000 positive and 5,000 negative examples, split with an 80/20 ratio into training and test sets. Training ran for 50 epochs with a batch size of 64. During training, the loss quickly dropped to nearly zero, indicating that the model was able to perfectly fit the training data. However, the test accuracy remained consistently around 0.5, which is equivalent to random guessing. This indicates that the model overfitted the training set and failed to generalize to new data. The failure likely stems from the challenge posed by Language 9, which requires identifying if the first and last characters in a sequence are the same—a long-range dependency that standard LSTMs struggle to capture effectively.

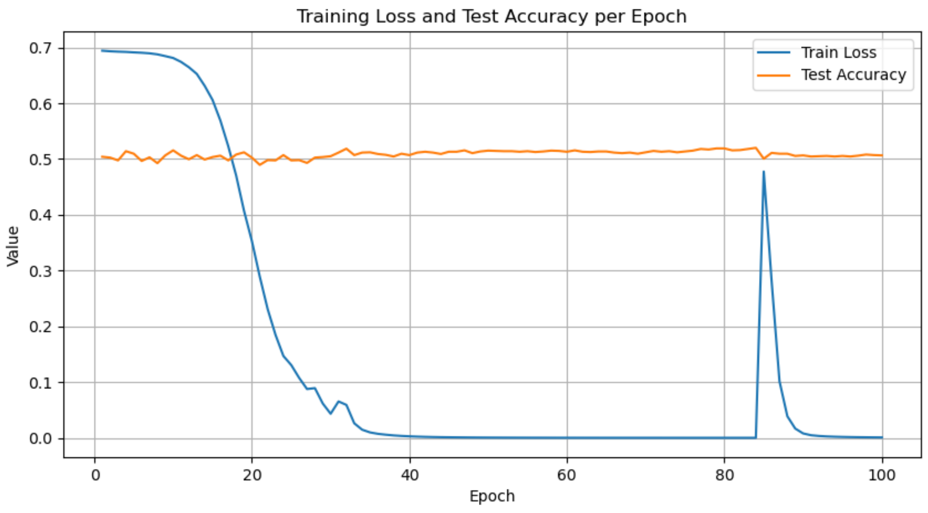


4. Language 10

Language Description:
Strings with even length are considered valid, while strings with odd length are invalid.

Challenge Rationale:
This requires a global property of the sequence (its length), which cannot be easily inferred from local patterns.

Experiment Outcome:
The model failed to detect this property consistently, likely because the decision is not based on token content. The model was trained similarly on 5,000 positive and 5,000 negative samples with an 80/20 training-test split, over 100 epochs with batch size 64. The training loss rapidly decreased to near zero, demonstrating perfect memorization of the training data. Yet, the test accuracy remained stuck at approximately 0.5 throughout training, showing no generalization capability. This result is attributed to the nature of the language, where the classification depends on a global property—the parity of the sequence length (even vs. odd). Such global features are not easily captured by local recurrent architectures like LSTMs, causing the model to fail both in learning a generalizable rule and in distinguishing valid from invalid sequences in unseen data.



5. Summary

Languages 8–10 were designed to probe limitations of sequence-based models like LSTMs. All three languages resulted in model failure, reinforcing known difficulties in modeling counting, distant dependencies, and global properties using standard recurrent architectures.