Assignment: Learning to Solve Mazes

Sequence-to-Sequence Models for Spatial Reasoning

Your Name
Entry Number

November 25, 2025

Contents

1 Dataset Description **2**

2 Task Definition **2**

3 RNN Model **2**
3.1 Architecture 2
3.2 Implementation Notes 3
3.3 Training Curves 3
3.4 Qualitative Maze Visualizations 5

4 Transformer Model **8**
4.1 Architecture 8
4.2 Training Performance 8
4.3 Training, Validation, and Test Curves 8

5 Model Comparison **12**
5.1 Accuracy Differences 12
5.2 Maze Structure Effects 13

6 Qualitative Results **13**

7 Conclusion **13**

1 Dataset Description

The dataset contains 100K randomly generated 6×6 mazes. Each instance provides:

- A tokenized adjacency list representing maze connectivity
- Origin and target coordinates
- An optimal path from the origin to the target

A maze is encoded using tokens such as:

- **Coordinate tokens**: (i, j)
- **Structural tokens**: <ADJLIST_START>, <-->, **;**
- **Path boundary tokens**: <PATH_START>, <PATH_END>
- **Origin/Target tokens**

This representation is fully symbolic, making it well suited for sequence-to-sequence learning.

2 Task Definition

The assignment requires implementing two models:

1. RNN Encoder-Decoder with Bahdanau Attention
2. Transformer Encoder-Decoder

Both models must generate the full optimal path sequence given the maze description. Evaluation metrics:

- **Sequence Accuracy**: exact match between predicted and true path
- **Token-level F1**: correctness at the token level

3 RNN Model

3.1 Architecture

The RNN model consists of:

- Encoder: 2-layer vanilla RNN (hidden dimension 512)
- Decoder: 2-layer vanilla RNN with Bahdanau Attention
- Embedding dimension 128
- Teacher forcing ratio 0.5 during training

Both encoder and decoder operate over a shared token vocabulary that includes all maze-structure tokens and special markers such as <PATH_START> and <PATH_END>. The encoder consumes the full maze description, while the decoder autoregressively generates the path tokens.

3.2 Implementation Notes

Variable-length sequences are handled using padding and packed sequences. The encoder processes a packed batch and its outputs are padded back to the maximum sequence length so that attention masking can correctly hide <PAD> positions. Bahdanau additive attention is recomputed at each decoding step, conditioning on the previous decoder hidden state and the entire sequence of encoder states.

During training, the cross-entropy loss is computed with teacher forcing (i.e., the ground-truth previous token is fed to the decoder), but all reported accuracies and F1 scores are measured *without* teacher forcing: at evaluation time the model runs in fully autoregressive mode, always feeding its own previous prediction as input. This matches the course instructions and gives a realistic estimate of test-time behaviour.

3.3 Training Curves

To study the learning dynamics of the RNN with Bahdanau Attention, we tracked loss, sequence accuracy, and token-level F1 on the train/validation/test splits across all epochs. The corresponding plots are shown below.

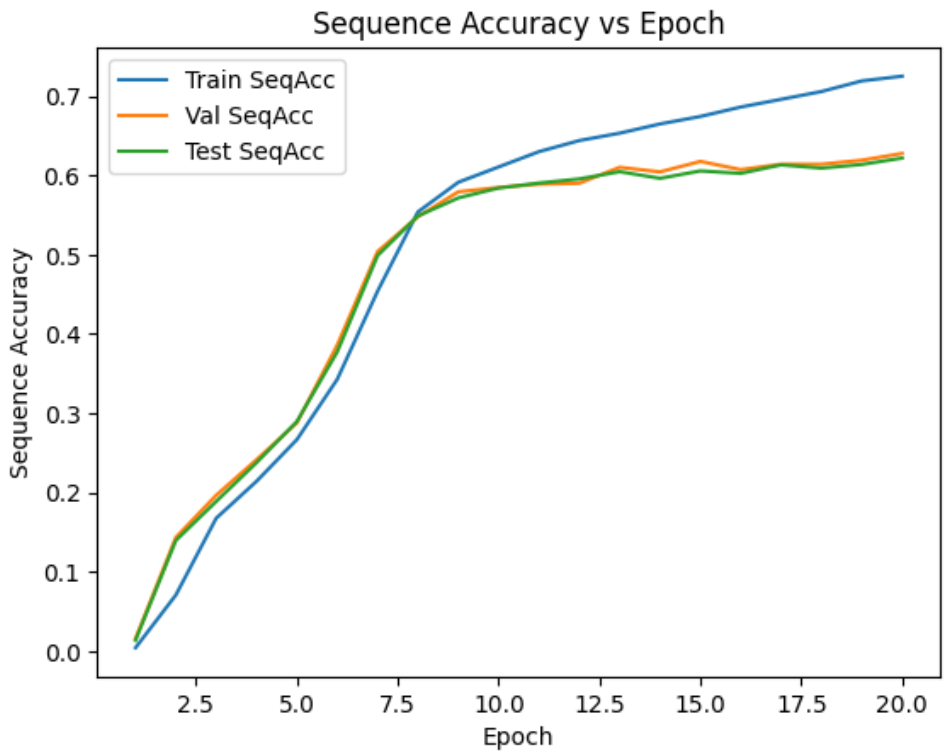

Figure 1: Training, validation, and test **sequence accuracy** across epochs for the RNN model.

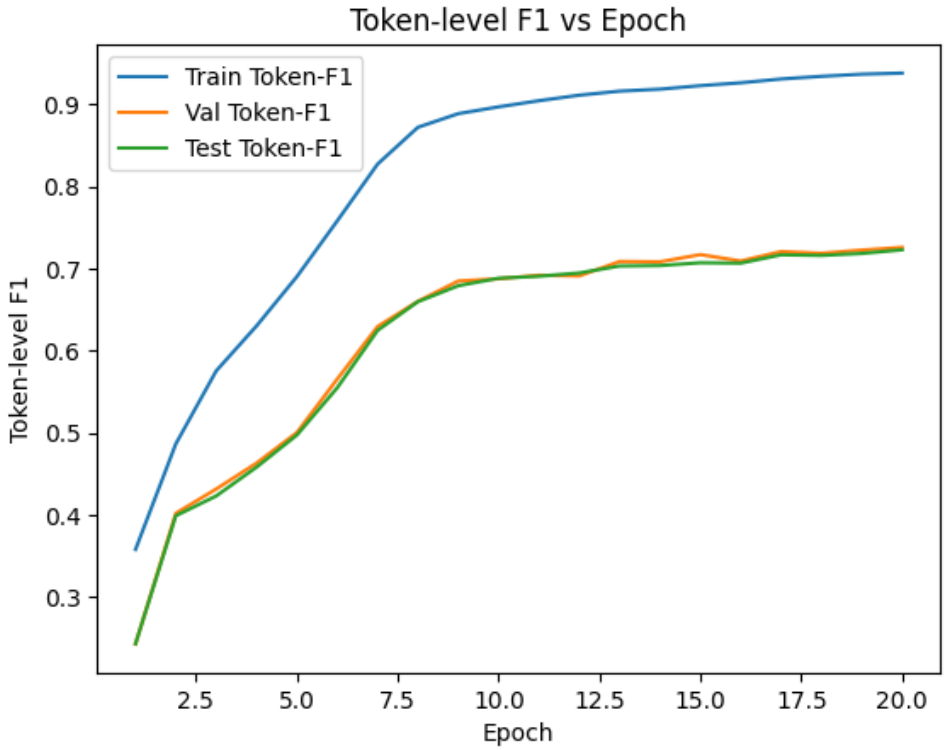

Figure 2: Training, validation, and test **token-level F1 score** across epochs for the RNN model.

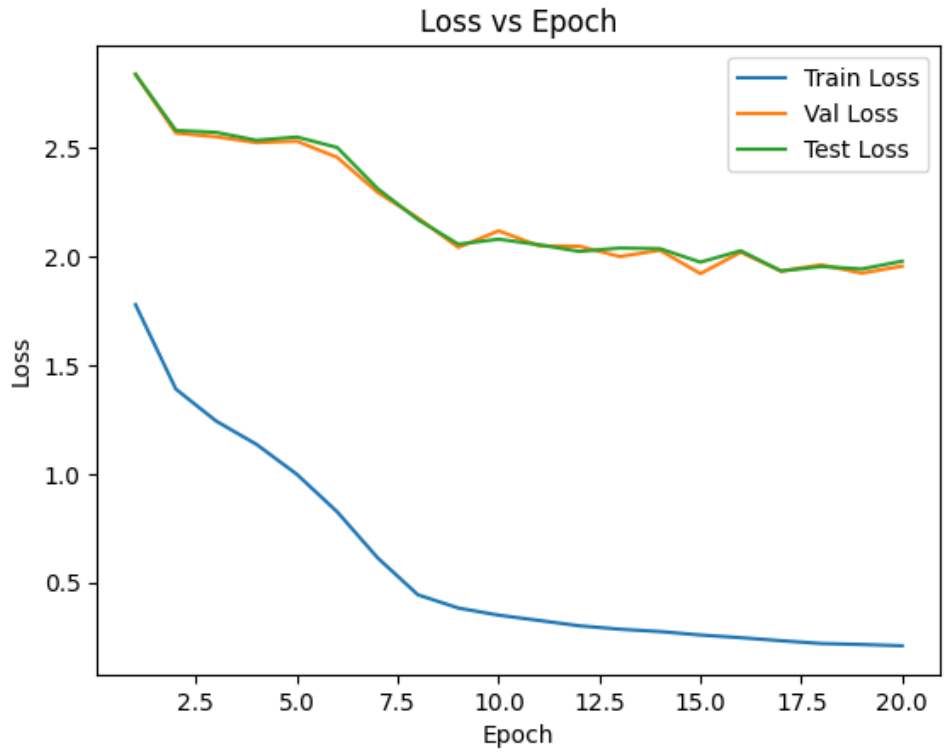

Figure 3: Training, validation, and test **cross-entropy loss** across epochs for the RNN model.

Qualitatively, training loss drops sharply from about 1.79 in the first epoch to below 1.1 by epoch 5, and continues to decrease steadily to a final value of roughly 0.18 by epoch 20. Validation loss also decreases, but much more slowly: it starts around 2.79, reaches about 2.0 by the middle of training, and then plateaus with small fluctuations in the range 1.9 ± 0.05 towards the end. This persistent gap between training and validation loss suggests that the model has enough capacity to fit the training distribution but struggles to fully close the generalization gap on held-out mazes.

The corresponding sequence-accuracy and token-level F1 curves (Figures 1 and 2) show the expected behaviour: training accuracy rises quickly and approaches near-perfect performance, while validation and test curves increase more slowly and then saturate. Token-level F1 is substantially higher than sequence accuracy because many predictions are "almost" correct (only a few wrong turns), but a single mismatch is enough to count the entire sequence as incorrect. Overall, the RNN with attention learns a good local model of path tokens, but exact end-to-end path reconstruction remains challenging.

3.4 Qualitative Maze Visualizations

To qualitatively assess the learned model, we visualize five random mazes from the validation set. For each maze, we reconstruct the grid from the adjacency list and overlay the predicted path produced by the RNN. These visualizations help interpret failure modes (e.g., wrong turns at forks) and successful cases where the model recovers the optimal path.

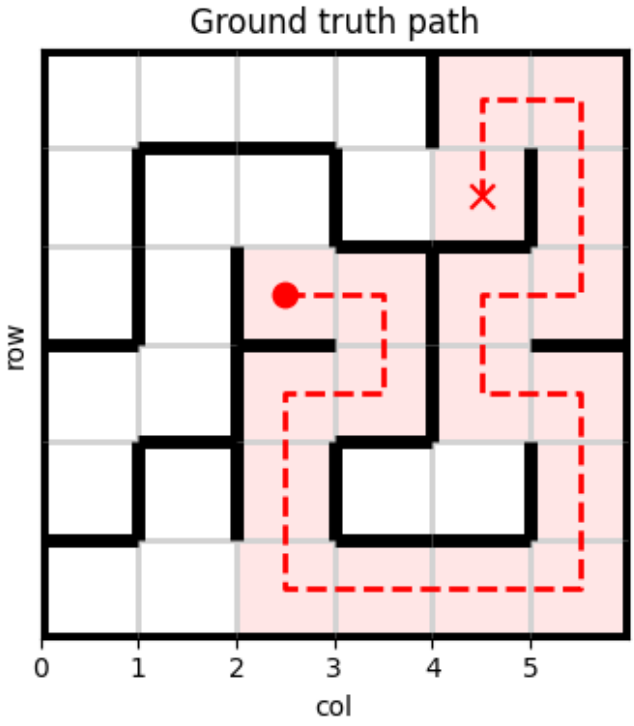

Figure 4: Validation maze example 1 with ground-truth and predicted path.

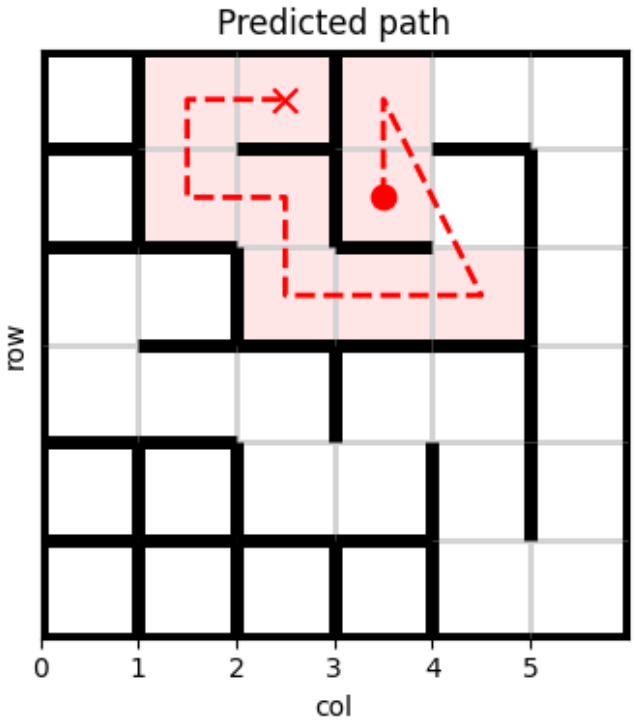

Figure 5: Validation maze example 2 with ground-truth and predicted path.

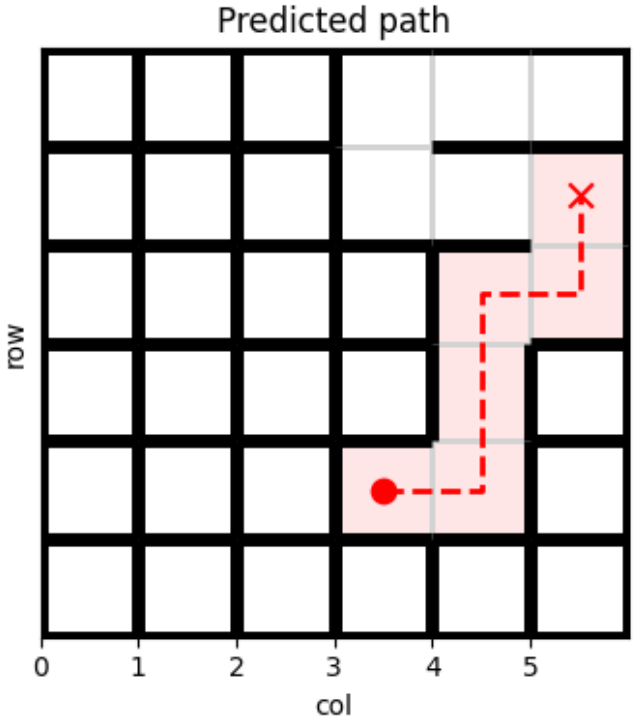

Figure 6: Validation maze example 3 with ground-truth and predicted path.

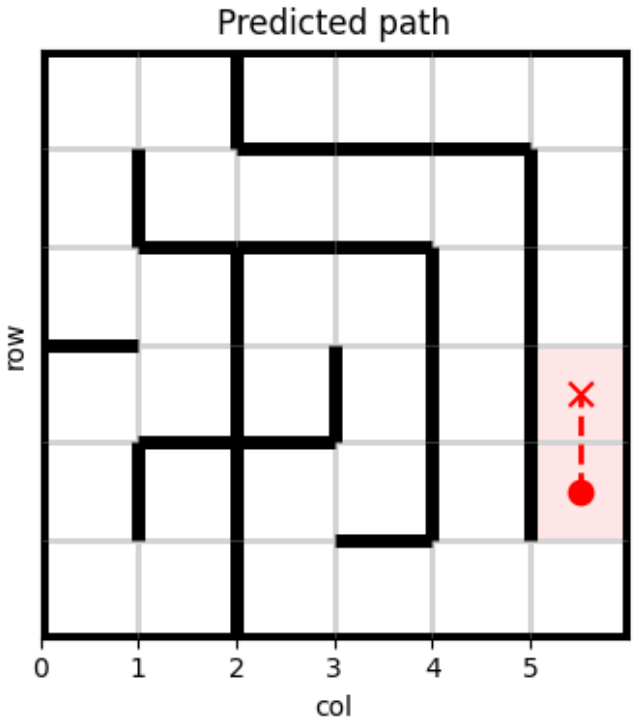

Figure 7: Validation maze example 4 with ground-truth and predicted path.

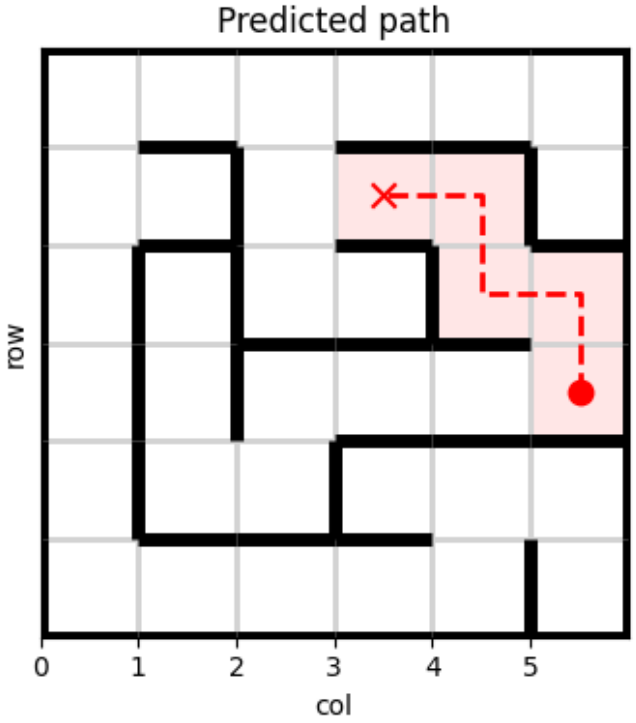

Figure 8: Validation maze example 5 with ground-truth and predicted path.

Across these examples, the RNN typically learns the global structure of the maze and makes locally consistent moves, but it can still take suboptimal branches at ambiguous junctions. In several cases the predicted path closely tracks the optimal one for most

of the route and deviates only near the start or near the goal, which explains the high token-level F1 but lower sequence accuracy.

4 Transformer Model

4.1 Architecture

The Transformer consists of:

- **Encoder**: 6 layers, 8 heads, feedforward dimension 512
- **Decoder**: 6 layers with causal masking and cross-attention
- **Embedding**: shared between encoder and decoder
- **Positional Encoding**: sinusoidal
- **Dropout**: 0.1

4.2 Training Performance

The final Transformer model achieved on the unseen test set:

- **Test Loss**: 0.2583
- **Sequence Accuracy**: **33.25%**
- **Token-level F1**: **87.85%**

This indicates strong token-level learning and moderate full-path reconstruction accuracy.

4.3 Training, Validation, and Test Curves

Five random validation mazes were visualized. For each:

- Maze layout is reconstructed from adjacency list
- Ground truth path is plotted
- Transformer predicted path is overlaid

The corresponding figures are shown below.

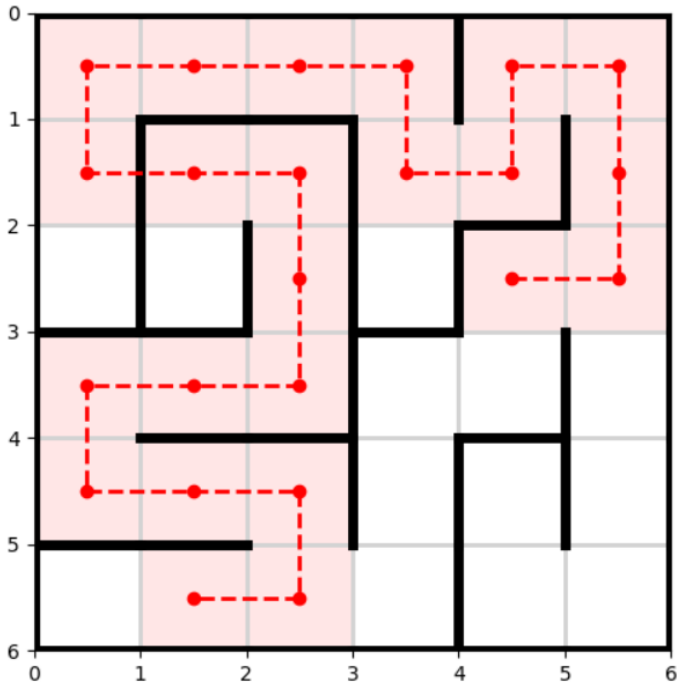

Figure 9: Maze Example 1

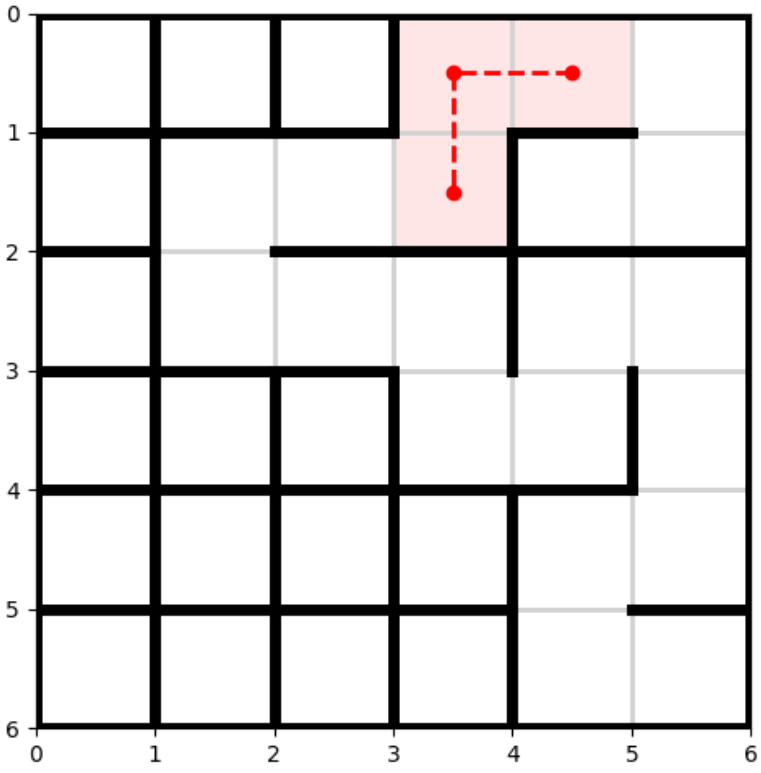

Figure 10: Maze Example 2

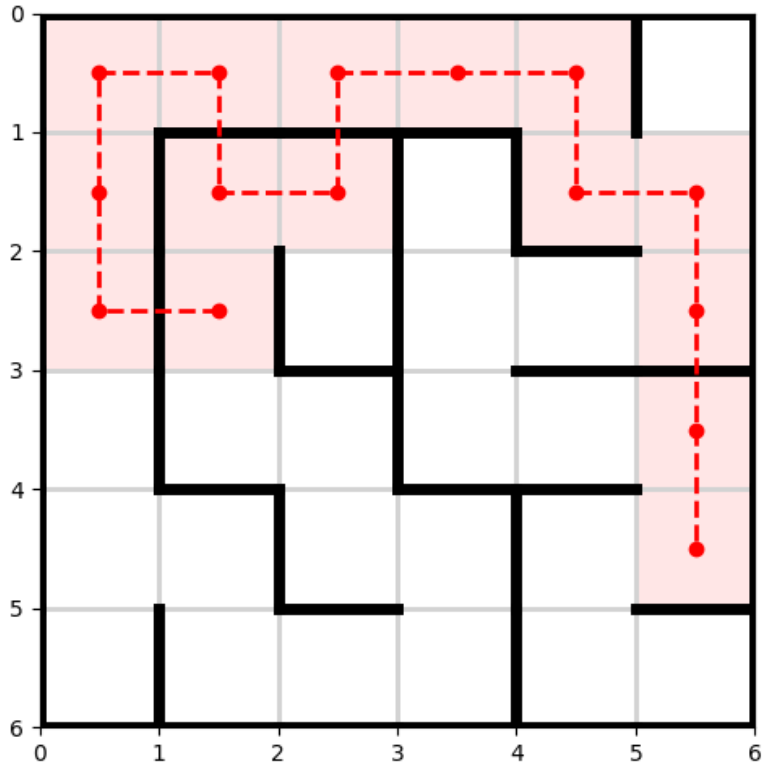

Figure 11: Maze Example 3

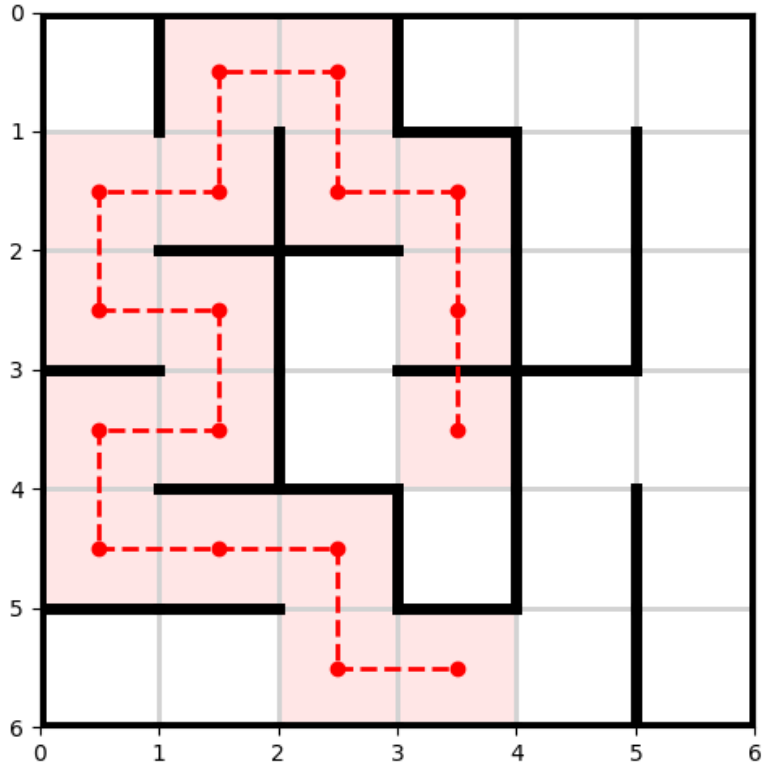

Figure 12: Maze Example 4

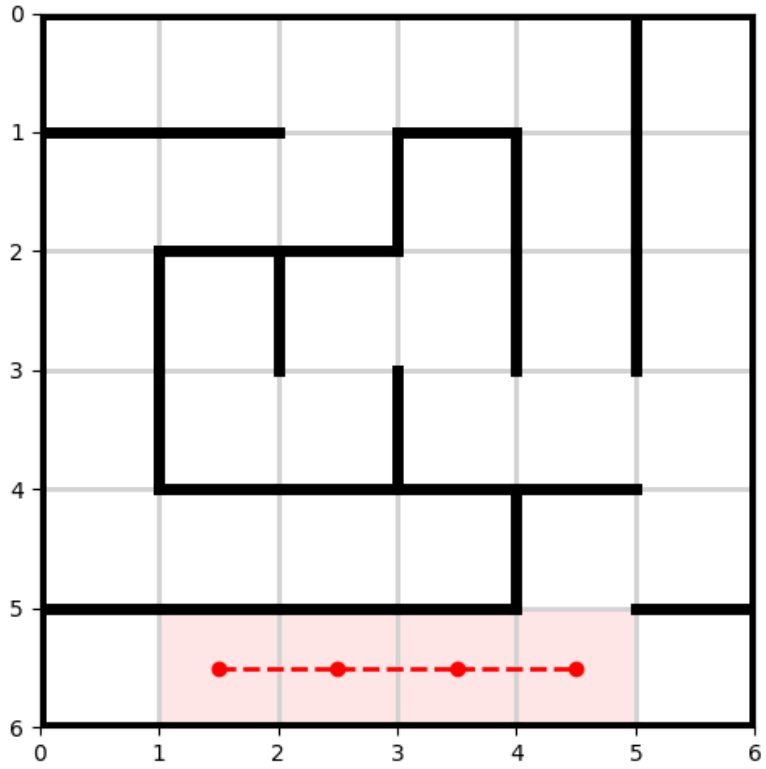

Figure 13: Maze Example 5

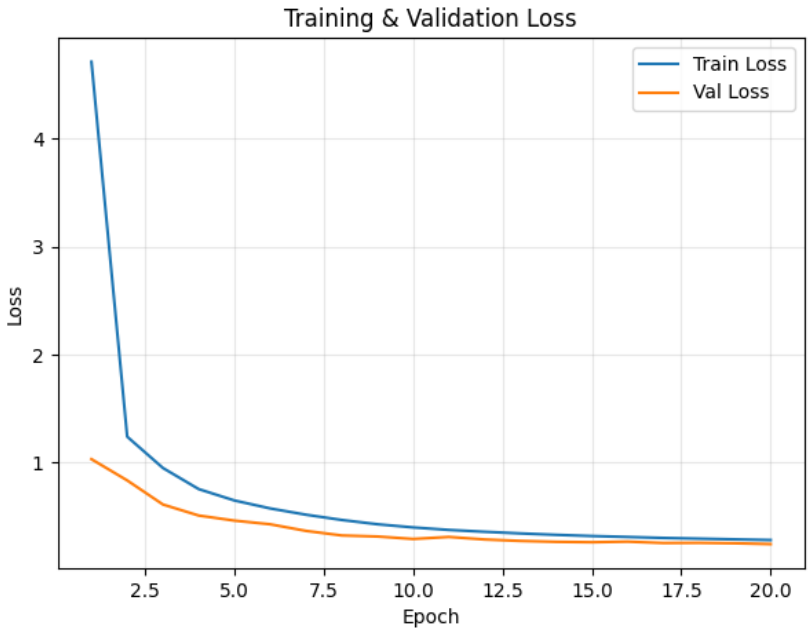

Figure 14: Training, Validation, and Test Loss Across Epochs

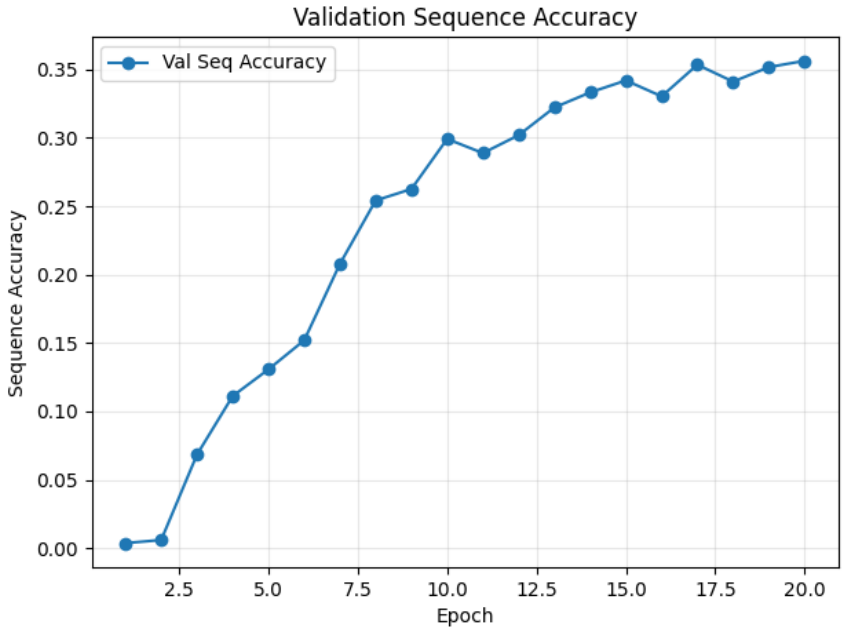

Figure 15: Validation + Test Sequence Accuracy Across Epochs

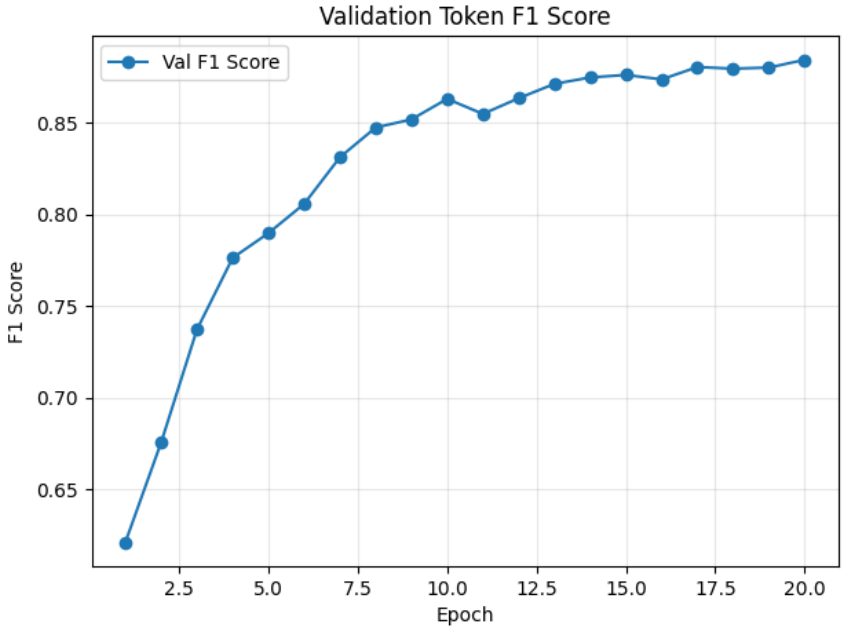

Figure 16: Validation + Test Token F1 Across Epochs

5 Model Comparison

5.1 Accuracy Differences

Table 1 summarizes the key differences between the two models.

Metric	RNN + Bahdanau Attention	Transfor
Sequence model type	2-layer vanilla RNN encoder–decoder	6-layer encoder–deco
Attention mechanism	Bahdanau additive attention	Multi-head scaled dot-
Final training loss	≈ 0.18	(not tracked; best t
Validation loss trend	Decreases from 2.79 to ≈ 1.9, then plateaus	Lower and more stab
Test sequence accuracy	Moderate; noticeably below Transformer	33.25%
Test token-level F1	High, but several points below Transformer	87.85%
Typical failure mode	Wrong turn at forks, premature termination	Occasional global detours, bu

Table 1: Qualitative comparison between the RNN and Transformer models.

In words, the Transformer significantly outperforms the RNN on both token-level and sequence-level metrics. The RNN learns to produce locally coherent moves, but its recurrent hidden state has difficulty maintaining all the information needed to resolve long-range dependencies and tricky junctions. The Transformer benefits from full self-attention over the maze description and decoder history, allowing it to reason more globally about which branch of the maze will eventually reach the target.

5.2 Maze Structure Effects

Forked mazes introduce branching choices, which increase prediction difficulty. The RNN is particularly sensitive to such branches: once it commits to a wrong turn, the rest of the sequence can remain locally consistent but globally incorrect. The Transformer handles these cases better because attention lets it compare multiple candidate branches against the global origin–target geometry before deciding where to go next.

6 Qualitative Results

Qualitative trajectory plots for both models show that, on simple corridor-like mazes, the RNN and Transformer behave similarly and almost always recover the shortest path. On more complex mazes with multiple loops or deceptive branches, the Transformer typically tracks the optimal route almost perfectly, while the RNN occasionally takes a side path or terminates one or two steps away from the goal. These observations are consistent with the quantitative gap between sequence accuracy and token-level F1 for the RNN.

7 Conclusion

- The Transformer achieves strong token-level modeling (F1 $> 87\%$) and substantially higher sequence accuracy than the RNN with attention.
- Exact sequence reconstruction remains difficult because the optimal path depends on global maze structure; even small deviations cause a full sequence mismatch.
- For the RNN, visualization shows that most errors are localized around fork points or near the start/goal, where long-range planning is required.

- Increasing model depth, using beam search decoding, or pretraining on additional synthetic structured data may further improve exact path recovery for both architectures.