1 HW3 - Vlad Pavlovich

2 Data Set Understanding / Task

The main task for this assignment was to implement a machine learning model to perform slot tagging on sentences. The training dataset consists of three columns: an ID, a sentence, and the corresponding slot tags. There are three main types of tags used for classification: **O** (outside a named entity), **B_** (beginning of a named entity), and **I_** (inside a named entity). The training set contains 2,298 sentences describing and providing information about movies, with a total of 27 different slot tags.

For example:

```
who plays luke on star wars new hope
O O B_char O B_movie I_movie I_movie I_movie
```

Here, "Star Wars a New Hope" is tagged with `B_movie` for the first token and `I_movie` for the remaining tokens.

3 Data Preparation

To prepare the dataset for sequence labeling, I converted each CSV row into numerical tensors suitable for machine learning models. Each sentence is tokenized into individual words, and the corresponding IOB tags are parsed and validated to ensure correct alignment.

Two vocabularies are constructed: (1) a word-to-ID mapping that includes `PAD` and `UNK` tokens, and (2) a tag-to-ID mapping for the 27 slot labels.

Because sentences vary in length, both the token sequences and their tag sequences are padded to a uniform length. However, the original sequence lengths are preserved for use during training. The final preprocessing step outputs two tensors, X and Y, containing padded word ID sequences and padded tag ID sequences, respectively. These serve as the core training inputs for all evaluated models.

4 Model Descriptions

For the sequence labeling task, I experimented with several neural architectures designed to capture contextual dependencies across sentences: RNN, LSTM, GRU, and a hybrid LSTM+CNN model.

RNN: Serves as the simplest baseline, capturing sequential patterns through hidden state propagation.

LSTM / GRU: Improve upon RNNs by introducing gating mechanisms, mitigating exploding/vanishing gradients and allowing long-range dependencies to be learned more effectively.

LSTM+CNN Hybrid: Combines a bidirectional LSTM for global context with a 1D CNN applied to LSTM outputs to extract local n-gram features.

All models were implemented with both unidirectional and bidirectional variants.

5 Training Setup and Hyperparameter Exploration

All models were trained using the AdamW optimizer with Cross Entropy Loss. A fixed learning rate of 0.001 consistently produced the best results across architectures. Training used a batch size of 32, and I experimented with both 80/20 and 90/10 train-validation splits.

Experiments were run for 15–40 epochs depending on convergence rate.

Hyperparameters explored included:

- Embedding dimensions: 100–256
- Hidden dimensions: 128–256
- Number of layers: 1–2
- Bidirectional vs. unidirectional models

Model performance was evaluated using the F1 score.

6 Figure 1: Loss Curves

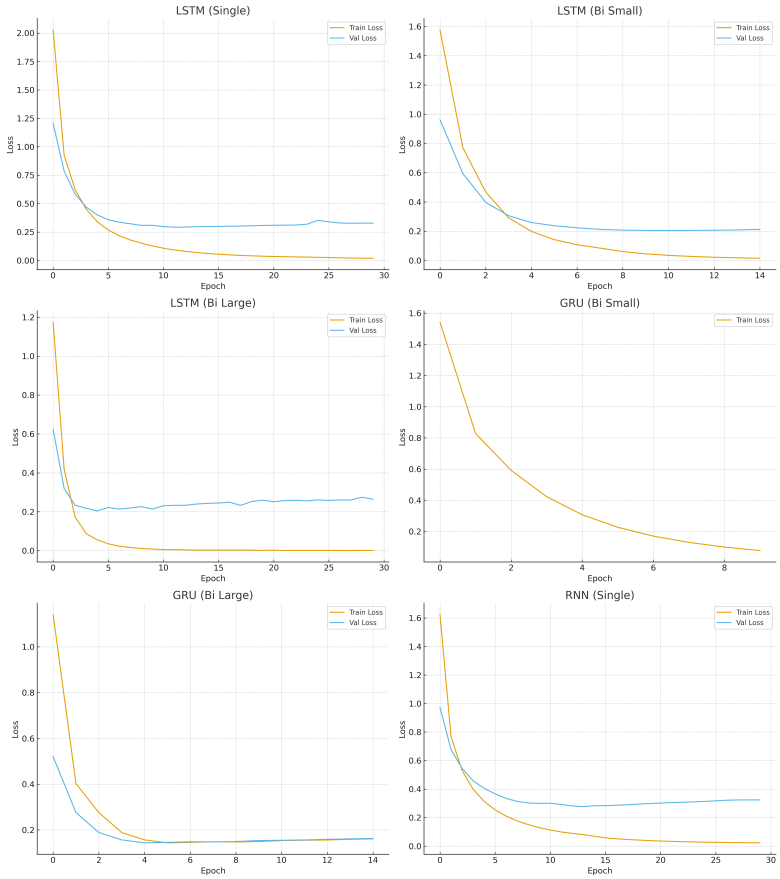

Figure 1: Training and validation loss curves for different model configurations.

7 Architectures and Experiments

I evaluated four main architectures: RNN, LSTM, GRU, and a hybrid LSTM+CNN model. Initial experiments used single-direction recurrent layers with 100-dimensional embeddings and 128 hidden units. These produced baseline F1 scores in the 0.60–0.64 range. Introducing bidirectionality significantly improved performance. Bidirectional LSTMs increased the F1 score to 0.68. Scaling the embedding dimension to 256, hidden dimension to 256, and using 2 recurrent layers raised the F1 to 0.73. GRU models showed similar improvement trends. The best-performing model overall was a bidirectional GRU with large embeddings, achieving an F1 score of **0.75**, outperforming all LSTM variants. The LSTM+CNN hybrid improved over simple single-direction models but plateaued

at an F1 of 0.66, underperforming compared to the stronger bidirectional recurrent models. Overall, the experiments show that bidirectionality, larger embedding sizes, and larger hidden dimensions consistently improve performance. Models with more than two layers tended to overfit.

8 Results

Across all experiments, the GRU architecture demonstrated the best overall performance, with the highest-scoring model achieving an F1 of **0.75**. These results highlight the effectiveness of recurrent architectures—especially bidirectional GRUs—for slot tagging tasks.

9 Figure 2: Model Results

Model	Direction	Embedding	Hidden	Layers	Split	F1 Score
RNN	Single	100	128	1	80/20	0.60
LSTM	Single	100	128	1	80/20	0.64
LSTM	Bi	100	128	1	80/20	0.68
LSTM	Bi	256	256	2	80/20	0.73
GRU	Bi	100	128	1	90/10	0.74
GRU	Bi	256	128	1	90/10	0.75
LSTM+CNN	Bi	256	256	2	80/20	0.66

Figure 2: Comparative F1 scores across model architectures and hyperparameter settings.